

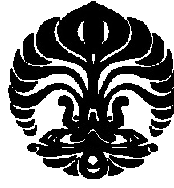
UNIVERSITAS INDONESIA

**PENCOCOKAN HAMPIRAN UNTAI
DENGAN PROGRAM DINAMIK**

TESIS

**RURUH WURYANI
0906577394**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MAGISTER MATEMATIKA
DEPOK
JULI 2011**



UNIVERSITAS INDONESIA

**PENCOCOKAN HAMPIRAN UNTAI
DENGAN PROGRAM DINAMIK**

TESIS

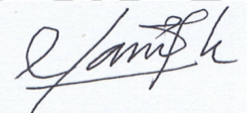
diajukan sebagai salah satu syarat
untuk memperoleh gelar Magister Sains

**RURUH WURYANI
0906577394**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MAGISTER MATEMATIKA
DEPOK
JULI 2011**

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah hasil karya sendiri,
Dan semua sumber baik yang dikutip maupun dirujuk
Telah saya nyatakan dengan benar

Nama : RURUH WURYANI
NPM : 0906577394
Tanda tangan : 
Tanggal : Juni 2011

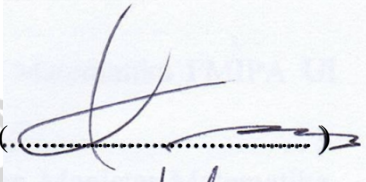
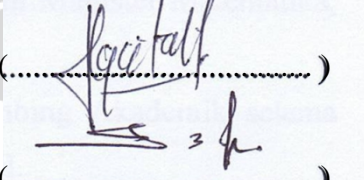
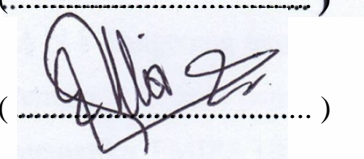

HALAMAN PENGESAHAN

Tesis ini diajukan oleh;

Nama : Ruruh Wuryani
NPM : 0906577394
Program Studi : Magister Matematika
Judul Tesis : Pencocokkan Hampiran Untai dengan Program Dinamik.

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Sains pada Program Studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Prof. Dr. Djati Kerami ()
Penguji : Gatot F. Hertono, P.Hd. ()
Penguji : Dr. rer. nat. Hendri Murfi, M. Kom. ()
Penguji : Alhadi Bustamam, P.Hd. ()

Ditetapkan di : Depok
Tanggal : 14 Juli 2011

KATA PENGANTAR

Tiada kata yang patut terucap selain puji syukur kehadiran Allah SWT. yang telah melimpahkan rahmat-Nya yang begitu besar hingga pada akhirnya penulis dapat menyelesaikan tesis ini dengan baik. Penulisan tesis ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Magister Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia.

Dalam prosesnya, tentulah tidak mudah bagi penulis untuk menyelesaikan tesis tanpa adanya bimbingan serta bantuan dari berbagai pihak. Oleh sebab itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. Djati Kerami, selaku dosen pembimbing yang penulis sangat hormati. Sungguh besar jasa beliau yang dengan sabar membimbing penulis dalam menyelesaikan penyusunan tesis ini. Semoga Allah membalas segala kebaikan beliau. Amin.
2. Bapak Dr. Yudi Satria selaku Kepala Departemen Matematika FMIPA UI beserta staf-stafnya.
3. Bapak Prof. Dr. Djati Kerami, selaku Ketua Program Magister Matematika, FMIPA UI beserta staf-stafnya.
4. Ibu Dr. Dian Lestari, M.Si. selaku dosen Pembimbing Akademik selama penulis belajar di Departemen Matematika FMIPA UI..
5. Bapak H. Hikmat, S.Pd.,MM selaku Pimpinan di SMA N 8 Tangerang tempat penulis bekerja, yang telah memberikan ijin dan kesempatan kepada penulis untuk melanjutkan belajar pada Program Magister Matematika, FMIPA UI.
6. Ibu dan Bapak(alm), yang telah membesarkan, mendidik, dan mengayomi penulis tiada henti, sehingga menjadi pendorong bagi penulis untuk menyelesaikan tesis ini dengan sebaik-baiknya. Semoga Allah selalu melimpahkan rahmat-Nya kepada mereka. Amin.
7. Suamiku Nur Wijaya dan anak-anakku Lala, Sasa dan Rara yang tercinta dan tersayang telah menjadi pendorong dan pengertiannya atas kesibukan penulis.

8. Muhamad Rafly Fadillah , selaku teman diskusi yang selalu siap membantu penulis, mengajarkan dan mengenakan pemrograman C, yang sama-sama berjuang dalam menyelesaikan tugas akhir.
9. Bu Bella, Bu Sri, Bu Kiki, Pak Gatot, Pak Hengky, Bu Bevina, Pak Hendri, Pak Al Hadi serta seluruh Bapak dan Ibu dosen Program Magister Matematika FMIPA UI yang telah memberikan motivasi kepada penulis hingga terselesaikannya tesis ini.
10. Mbak Santi dan Mbak Rusmi yang selalu bersedia menjawab pertanyaan penulis seputar birokrasi akademik, terima kasih pula kepada Pak Salman, Pak Saliman, serta seluruh staf tata usaha Departemen Matematika FMIPA UI.
11. Teman-teman seperjuangan mahasiswa Program Magister Matematika angkatan 2009.
12. Para sahabat yang telah sekian banyak membantu dalam menyelesaikan tesis ini, terima kasih semuanya.

Penulis berharap dan berdo'a kepada Allah SWT. agar senantiasa berkenan membalas segala kebaikan dari semua pihak yang telah membantu. Amin.

Penulis hanyalah makhluk yang jauh dari sempurna sehingga penulis menyadari tidak luput dari kesalahan. Untuk itu, penulis mohon saran dan kritik untuk penyempurnaan tesis ini. Semoga tesis ini bisa memberikan manfaat yang berguna dalam pengembangan ilmu pengetahuan.

Depok, 14 Juli 2011

Penulis

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Ruruh Wuryani
NPM : 0906577394
Program Studi : Magister Matematika
Departemen : Matematika
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jenis karya : Tesis

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Pencocokkan Hampiran Untai dengan Program Dinamik

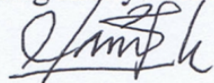
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : Juli 2011

Yang menyatakan



(Ruruh Wuryani)

ABSTRAK

Nama : Ruruh Wuryani
Program Studi : Matematika
Judul : Pencocokan Hampiran Untai dengan Program Dinamik

Dalam tesis dibahas pencocokan hampiran untaai (*approximate string matching*) dari dua untaai berbeda. Dalam meninjau tingkat kedekatan hampiran dua untaai atau tingkat kemiripan dua untaai digunakan ukuran Jarak Levenshtein, Dalam penentuan jarak tersebut digunakan metode program dinamik. Diperoleh beberapa sifat-sifat yang berhubungan dengan susunan kedua untaai yang dicocokkan. Pada akhir tesis diberikan juga program komputer sederhana dalam penentuan jarak Levenshtein.

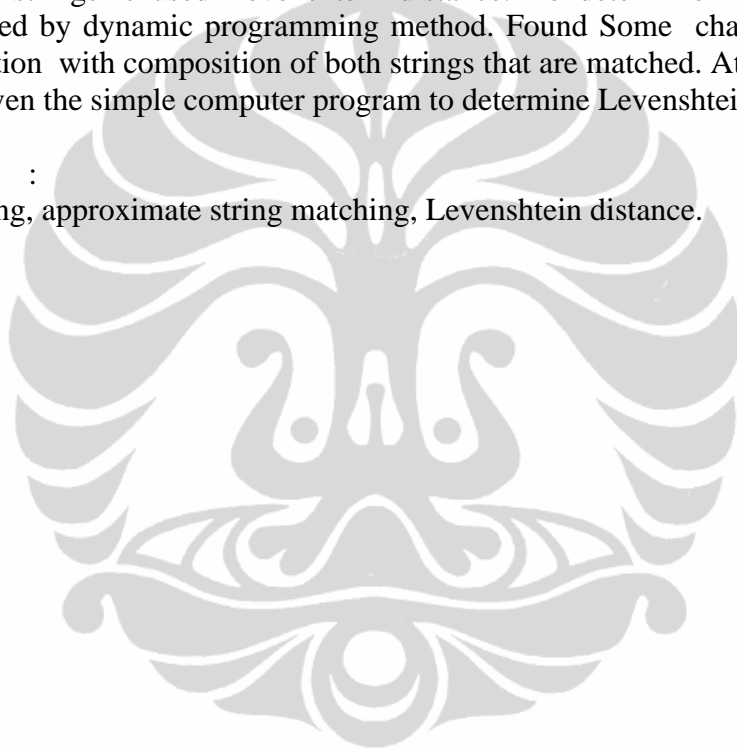
Kata Kunci :
Pencocokan untaai, pencocokan hampiran untaai, jarak Levenshtein.

ABSTRACT

Name : Ruruh Wuryani
Program Study : Mathematics
Title : Approximate String Matching by Dinamic Programming

In this thesis described approximate string matching problem between two different strings. To show the approximate level of both strings or the similarity level of both strings is used Levenshtein distance. To determine Levenshtein distance is used by dynamic programming method. Found Some characteristics that have relation with composition of both strings that are matched. At the end of the thesis, given the simple computer program to determine Levenshtein distance.

Key Words :
String matching, approximate string matching, Levenshtein distance.



DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PERNYATAAN ORISINALITAS	ii
HALAMAN PENGESAHAN	iii
KATA PENGANTAR	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI	vi
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR LAMPIRAN	xiii
1. PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Perumusan Masalah dan Ruang Lingkup	3
1.3. Jenis Penelitian dan Metode yang Digunakan.....	3
1.4. Tujuan Penelitian	4
2. UNTAI DAN OPERASI UNTAI	5
2.1. Untai.....	5
2.1.1. Definisi Alfabet	5
2.1.2. Definisi Untai	5
2.1.3. Himpunan Untai	6
2.1.4. Panjang Untai dan Posisi Simbol pada Untai	7
2.1.5. Operasi pada Untai	8
2.1.6. Eksponensiasi pada Untai	9
2.1.7. Subuntai, Prefiks, dan Sufiks	9
2.1.8. Operasi antar Dua Himpunan Untai	10
2.2. Pencocokan Untai Eksak	11
2.3. Jarak	11
2.4. Jarak Hamming.....	12
3. PENCOCOKAN HAMPIRAN UNTAI DENGAN UKURAN JARAK LEVENSHTEIN	13
3.1. Jarak Levenshtein.....	13
3.2. Pencocokan Hampiran Untai dengan Jarak Levenshtein	15
3.3. Pencocokan Hampiran Untai dengan Ukuran Jarak Levenshtein Menggunakan Program Dinamik	20
3.3.1. Program Dinamik	20
3.3.2. Pencocokan Hampiran Untai dengan Menggunakan Program Dinamik	21
3.4. Algoritma Pencocokan Untai terdekat dengan Program Dinamik ...	31
3.5. Sifat-Sifat pada pencocokan Untai Terdekat dengan Ukuran Jarak Levenshtein	32

4. IMPLEMENTASI DAN SIMULASI	42
4.1. Implementasi	42
4.2. Simulasi	42
5. PENUTUP	47

DAFTAR PUSTAKA
LAMPIRAN



DAFTAR GAMBAR

Gambar 3.1	Graf pencocokkan a pada $x(1)$ dengan c pada $y(1)$	16
Gambar 3.2	Graf pencocokkan b pada $x(1)$ dengan c pada $y(1)$	16
Gambar 3.3	Graf proses penghapusan b pada $x(2)$ atau pada $x'(1)$...	17
Gambar 3.4	Graf proses penyisipan c pada $y(1)$ atau pada $x''(1)$	17
Gambar 3.5	Graf proses penggantian b pada $x'(1)$ atau pada $x(2)$ dengan c pada $y(1)$	17
Gambar 3.6	Graf proses penyisipan c pada $x'(1)$ menjadi $x''(1)$	18
Gambar 3.7	Graf proses penghapusan b pada $x''(2)$	18
Gambar 3.8	Graf proses penggantian a pada $x(1)$ dengan c pada $y(1)$	18
Gambar 3.9	Graf proses penghapusan b pada $x'(2)$	19
Gambar 3.10	Graf proses penyisipan c pada $x(1)$ menjadi c pada $x'(1)$	19
Gambar 3.11	Graf proses penghapusan a pada $x'(2)$ dan b pada $x'(3)$	19
Gambar 3.12	Graf keseluruhan proses pecocokkan antara $x = ab$ dan $y = c$	20
Gambar 3.13	Graf penggambaran keseluruhan kemungkinan proses pecocokkan antara $x = graph$ dan $y = graf$	22
Gambar 3.14	Graf pengisian titik-titik $D[j, 0]$ dan $D[0, i]$	23
Gambar 3.15	Graf pengisian titik-titik $D[j, 1]$	24
Gambar 3.16	Graf pengisian titik-titik $D[j, 2]$	26
Gambar 3.17	Graf pengisian titik-titik $D[j, 3]$	27
Gambar 3.18	Graf pengisian titik-titik $D[j, 4]$	28
Gambar 3.19	Graf pengisian titik-titik $D[j, 5]$	29
Gambar 3.20	Graf hasil proses pecocokkan antara $x = graph$ dan $y = graf$	30
Gambar 3.21	Graf hasil proses pecocokkan antara $x = gr$ dan $y = gr..$	33

Gambar 3.22	Graf hasil proses pecocokkan antara $x = time$ dan $y = like$	34
Gambar 3.23	Graf hasil proses pecocokkan antara $x = tim$ dan $y = say$	34
Gambar 3.24	Graf hasil proses pecocokkan antara $x = ^$ dan $y = cat$	35
Gambar 3.25	Graf hasil proses pecocokkan antara $x = tang$ dan $y = ^$	35
Gambar 3.26	Graf hasil proses pecocokkan antara $x = tuang$ dan $y = tang$	36
Gambar 3.27	Graf hasil proses pecocokkan antara $x = tag$ dan $y = tang$	37
Gambar 3.28	Graf hasil proses pecocokkan antara $x = tim$ dan $y = atm$	37
Gambar 3.29	Graf hasil proses pecocokkan antara $x = tuang$ dan $y = tugu$	38
Gambar 3.30	Graf hasil proses pecocokkan antara $x = tua$ dan $y = tiga$	39
Gambar 3.31	Graf hasil proses pecocokkan antara $x = tua$ dan $y = situ$	40
Gambar 4.1.	Tampilan awal pada <i>Command Window</i>	42
Gambar 4.2.	Tampilan input untai berikutnya pada <i>Command Window</i> ..	43
Gambar 4.3.	Tampilan tabel hasil perhitungan jarak Levenshtein pada <i>Command Window</i>	43
Gambar 4.4.	Tampilan kemungkinan pertama operasi-operasi yang dilakukan pada untai x	44
Gambar 4.5.	Tampilan kemungkinan kedua operasi-operasi yang dilakukan pada untai x	45
Gambar 4.6.	Tampilan kemungkinan ketiga operasi-operasi yang dilakukan pada untai x	45
Gambar 4.7.	Tampilan Hasil Akhir Pada <i>Command Window</i>	46

DAFTAR TABEL

Tabel 3.1. Tabel hasil proses pecocokkan antara $x = graph$ dan $y = graf$ 31



DAFTAR LAMPIRAN

Lampiran 1 Listing Program Interaktif Perhitungan Jarak Levenshtein.



BAB 1

PENDAHULUAN

1.1. Latar Belakang

Permasalahan pencocokan untai (*string matching*) merupakan permasalahan dalam dunia komputer. Contoh implementasi dari permasalahan pencocokan untai ini adalah pencarian sebuah untai (*string*) kata pada *Microsoft Word* pada menu *replace* ataupun teks editor yang lain. Hal tersebut adalah termasuk pencocokkan untai eksak. Dalam dunia bioinformatika, dikenal adanya pencocokan suatu untai DNA. Dalam pencocokkan untai DNA ini dua buah untai DNA dikatakan cocok atau mirip setelah tingkat kecocokkannya memenuhi tingkat kecocokkan yang ditentukan. Atau dengan kata lain tingkat perbedaan antara kedua untai DNA tersebut tidak melebihi dari suatu angka tertentu yang diberikan. Pencocokkan untai DNA ini termasuk pencocokkan hampiran untai (*approximate string matching*).

Dalam teori komputasi, sebuah untai didefinisikan sebagai suatu barisan berhingga dari simbol-simbol atau karakter-karakter anggota himpunan alfabet V . Misal diberikan sebuah alfabet $V = \{a, b, c, \dots, z\}$, maka untai-untai yang dapat dibentuk sepanjang alfabet V tersebut antara lain adalah *aabbcd*, *semangka*, *matematika* dan sebagainya. Jika $V = \{0, 1\}$, maka untai-untai yang dapat dibentuk antara lain ialah *0111011*, *00001*, *01111*, dan lain sebagainya. Sebuah untai bisa saja tidak mengandung satupun karakter, dalam hal ini disebut dengan untai hampa yang dinotasikan dengan ϵ atau e .

Masalah utama dari pencocokkan untai adalah mencari sebuah untai yang terdiri dari beberapa karakter dalam sejumlah besar teks. Misalkan diberikan suatu teks $T = t_1 t_2 t_3 \dots t_n$ dan suatu untai $P = p_1 p_2 p_3 \dots p_m$. Selanjutnya akan dicari apakah P ada di dalam T dan jika ada dimana posisinya (*exact string matching*). Atau manakah untai yang paling mendekati P di dalam T dan di mana posisinya (*approximate string matching*) (Tretyakov, 2003). Proses yang

dilakukan adalah mencocokkan setiap dua untai kemudian menentukan yang paling mirip.

Untuk itu diperlukan suatu ukuran untuk menentukan tingkat kemiripan antara dua untai. Ukuran kemiripan antara dua untai ini disebut jarak antara dua buah untai. Jarak antara dua buah elemen a dan b yang merupakan anggota dari suatu himpunan tak kosong H yang dinotasikan dengan $D(a, b)$ didefinisikan sebagai suatu fungsi D yang memetakan (a, b) ke suatu bilangan real nonnegatif, atau $D: H \times H \rightarrow \mathbb{R}^+ \cup \{0\}$, yang memenuhi sifat-sifat nonnegatifitas, simetris, serta memenuhi ketaksamaan segitiga. Sedangkan jarak antara dua buah untai didefinisikan sebagai sebuah ukuran dari banyaknya operasi pada untai yang dilakukan untuk menyamakan dua untai. Operasi penyamaan untai itu sendiri bisa berupa penghapusan (*deletion*), penyisipan (*insertion*), maupun penggantian (*substitution*) satu atau lebih karakter pada suatu untai. Salah satu ukuran jarak antara dua buah untai ialah Jarak Levenshtein (*Levenshtein Distance*) yaitu banyaknya operasi minimum dalam menyamakan dua untai. Di suatu referensi lainnya jarak Levenshtein dinyatakan sebagai jarak Edit Jika diberikan dua buah untai, x dan y , maka jarak antara untai x dan untai y adalah banyaknya operasi-operasi penghapusan, penyisipan dan penggantian yang harus dilakukan untuk menyamakan karakter pada untai x menjadi sama dengan karakter pada untai y pada posisi yang bersesuaian.

Untuk menentukan untai yang paling mirip, maka dilakukan pencocokkan tiap dua untai dengan menghitung jarak Levenshtein antara dua untai yang dicocokkan. Karena begitu banyaknya cara yang dapat dilakukan untuk mencocokkan antara dua untai, maka menentukan banyaknya operasi minimal dalam pencocokkan bukanlah masalah yang mudah. Semakin panjang untai yang dicocokkan semakin banyak kemungkinan bentuk pencocokannya dan semakin sulit dalam menentukan jarak Levenshtein antara kedua untai. Sehingga diperlukan suatu cara atau metode yang tepat untuk menentukan jarak Levenshtein ini.

1.2. Perumusan Masalah dan Ruang Lingkup

Dari uraian latar belakang di atas, penulis merumuskan suatu rumusan masalah sebagai berikut:

- a. Bagaimanakah sifat-sifat pencocokan hampiran untai ini dilihat dari metode yang digunakan untuk menentukan jarak Levenshtein?
- b. Bagaimanakah membangun suatu algoritma dari metode pencocokan hampiran untai dengan menggunakan jarak Levenshtein sebagai ukurannya?

1.3. Jenis Penelitian dan Metode yang Digunakan

Jenis penelitian yang digunakan dalam penyusunan tugas akhir ini adalah studi literatur serta melakukan beberapa percobaan dan simulasi dari segala macam kemungkinan dari proses pencocokkan hampiran untai dan mengumpulkan sifat-sifatnya.

1.4. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

- a. Menginventarisir sifat-sifat yang muncul pada proses pencocokkan hampiran untai sehubungan metode yang digunakan dalam menentukan jarak Levenshtein. Membuat suatu algoritma dari metode pencocokkan hampiran untai dengan menggunakan jarak Levenshtein sebagai ukurannya.

BAB 2

UNTAI DAN OPERASI PADA UNTAI

Pada bab ini akan dipaparkan beberapa teori dasar tentang unta, pencocokan unta eksak, dan jarak yang akan digunakan dalam penelitian ini.

2.1. Untai

Berikut ini adalah beberapa teori dasar tentang unta secara matematis.

2.1.1. Definisi Alfabet

Misal diberikan suatu himpunan berhingga V yang terdiri dari simbol-simbol. Maka himpunan inilah yang kemudian disebut dengan abjad atau alfabet (*alphabet*) yang terkadang dinotasikan juga dengan Σ . Contohnya, Abjad Latin $\{a, b, c, \dots, z\}$ ataupun Abjad Yunani $\{\alpha, \beta, \gamma, \dots, \omega\}$. Abjad yang terutama sekali berhubungan dengan teori komputasi adalah Abjad Biner $\{0, 1\}$ (Lewis & Papadimitriou, 1998).

Dalam kehidupan nyata, terdapat banyak sekali simbol-simbol dengan bentuk yang beragam. Maka untuk penyederhanaan, yang kita gunakan sebagai simbol di sini adalah huruf, angka, dan simbol khusus lain seperti \$ ataupun #.

2.1.2. Definisi Untai

Untai (*string*) sepanjang alfabet V adalah sebuah barisan berhingga simbol-simbol anggota alfabet V tersebut. Contohnya, *aabbccdd*, *cgtcttgc*, dan *matematika* adalah unta sepanjang alfabet $\{a, b, c, \dots, z\}$ sedangkan 0111011, 00001, dan 01111 adalah unta sepanjang $\{0, 1\}$. Dalam bahasa sehari-hari, unta yang mempunyai makna dalam bahasa alami disebut juga sebagai suatu kata.

Terkadang sebuah unta hanya mengandung satu simbol saja, maka unta seperti ini disebut dengan unta tunggal. Contohnya, *a* adalah unta yang hanya

mengandung satu simbol saja yaitu a . Dalam hal ini, simbol a bisa disebut sebagai untai a .

Sebuah untai juga bisa saja tidak mengandung satupun simbol, untai seperti ini disebut dengan untai hampa yang dinotasikan dengan \wedge atau e .

2.1.3. Himpunan Untai

Himpunan dari semua untai, termasuk untai hampa, sepanjang alfabet V dinotasikan dengan V^* . Dalam hal ini, V^* disebut sebagai penutup (*closure*) dari himpunan V . Contohnya, jika $V = \{a, b\}$, maka penutup dari himpunan V adalah $V^* = \{\wedge, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, \dots\}$. Apabila untai hampa \wedge tidak dimasukkan dalam himpunan untai tersebut, maka himpunan untai tersebut menjadi $V^* - \{\wedge\}$ dan ditulis dengan V^+ . Di sini, himpunan hampa \emptyset atau $\{\}$ merupakan himpunan yang tidak mengandung satupun untai.

Teorema:

“Jika V adalah himpunan alfabet yang berhingga, maka V^* merupakan himpunan yang tak berhingga tetapi terhingga.”

Untuk membuktikannya, harus ditunjukkan bahwa terdapat pemetaan bijektif $f: \mathbb{N} \rightarrow V^*$.

Pertama, ambil sebuah himpunan alfabet $V = \{a_1, a_2, \dots, a_n\}$, dengan a_1, a_2, \dots, a_n yang berbeda dan telah terurut (diatur urutannya). Maka, setiap anggota dari V^* dapat dicacah dengan cara sebagai berikut:

- Untuk setiap $k \geq 0$, semua untai dengan panjang k dicacah sebelum semua untai yang panjangnya $k + 1$.
- Untai-untai yang panjangnya tepat k dicacah secara leksikografi, yaitu $a_{i_1} \dots a_{i_k}$ mendahului $a_{j_1} \dots a_{j_k}$, dengan aturan, untuk sembarang m dengan $0 \leq m \leq k - 1$, $i_p = j_p$ untuk $p = 1, \dots, m$, dan $i_{m+1} < j_{m+1}$.

Contohnya, jika diberikan $V = \{0,1\}$, maka urutan setiap anggota dari V^* adalah $\wedge, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \dots$

Dengan demikian, jika diberikan $V = \{a_1, a_2\}$, maka urutannya adalah sebagai berikut:

$$\begin{aligned} & \wedge \\ & a_1 \\ & a_2 \\ & a_1 a_1 \\ & a_1 a_2 \\ & a_2 a_1 \\ & a_2 a_2 \\ & a_1 a_1 a_1 \\ & a_1 a_1 a_2 \\ & a_1 a_2 a_1 \\ & a_1 a_2 a_2 \\ & \dots \dots \dots \end{aligned}$$

Jika V adalah Abjad Latin dan urutan dari $V = \{a_1, a_2, \dots, a_{26}\}$ adalah seperti biasa yaitu $\{a, b, \dots, z\}$, maka urutan leksikografi untuk untai-untai yang panjangnya sama diurutkan seperti dalam kamus. Meskipun demikian, aturan yang tercantum pada butir a dan b di atas yang mendaftar untai terpendek sebelum untai yang lebih panjang untuk semua untai anggota V^* berbeda dengan urutan pada kamus.

2.1.4. Panjang Untai dan Posisi Simbol pada Untai

Panjang dari sebuah untai x merupakan banyaknya simbol yang membentuk untai x tersebut, dinotasikan dengan $|x|$. Contohnya, jika $x = 101$, maka $|x| = |101| = 3$, jika $x = \wedge$, maka $|x| = |\wedge| = 0$.

Sebuah untai x anggota dari V^* juga dapat direpresentasikan sebagai suatu fungsi, yaitu $x: \{1, \dots, |x|\} \rightarrow V$. Nilai fungsi dari $x(i)$ untuk $1 \leq i \leq |x|$, i bilangan bulat, adalah simbol pada posisi ke- i dari x . Contohnya, jika $x = telepon$, maka $x(1) = t$ dan $x(4) = x(2) = e$. Pada dasarnya, simbol e pada posisi ke-4 identik dengan simbol e pada posisi ke-2. Akan tetapi, untuk membedakan simbol yang identik tersebut pada posisi yang berbeda dalam untai,

kita sebut mereka sebagai kejadian yang berbeda dari suatu simbol. Yaitu, simbol σ anggota dari V muncul pada posisi ke- i dari untai x anggota V^* jika $x(i) = \sigma$.

Untuk selanjutnya, simbol pada posisi ke- i dari x akan ditulis dengan x_i untuk $1 \leq i \leq |x|$, i bilangan bulat.

2.1.5. Operasi pada Untai

Dua buah untai sepanjang alfabet yang sama dapat dirangkai menjadi bentuk ketiga dengan operasi perangkaian (*concatenation*). Perangkaian dari untai x dan y yang ditulis dengan $x \circ y$ atau disingkat menjadi xy , adalah untai x yang diikuti oleh untai y . Secara formal, $w = x \circ y = xy$ jika dan hanya jika:

- $|w| = |x| + |y|$,
- $w(j) = x(j)$ untuk $j = 1, \dots, |x|$, dan
- $w(|x| + j) = y(j)$ untuk $j = 1, \dots, |y|$.

Contohnya, jika $x = 01$ dan $y = 001$, maka $x \circ y = 01 \circ 001 = 01001$, jika $x = \text{mata}$ dan $y = \text{hari}$, maka $x \circ y = \text{mata} \circ \text{hari} = \text{matahari}$. Berdasarkan definisi operasi perangkaian tersebut, didapat $x \circ \wedge = \wedge \circ x = x$ untuk sembarang untai x .

Himpunan V^* dengan operasi perangkaian merupakan monoid karena memenuhi sifat-sifat berikut:

- Tertutup terhadap operasi perangkaian.

Bukti: Ambil x dan y anggota V^* dengan panjang masing-masing m dan n .

Maka, $x = x_1x_2 \dots x_m$ dan $y = y_1y_2 \dots y_n$

dengan $x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n$ adalah simbol-simbol anggota V .

Sehingga $xy = x_1x_2 \dots x_my_1y_2 \dots y_n$ merupakan untai yang terdiri dari simbol-simbol anggota V juga.

Akibatnya, xy juga anggota V^* .

- Asosiatif, yaitu: $(xy)z = x(yz)$ untuk sembarang untai x , y , dan z .

Bukti: Ambil x , y , dan z anggota V^* dengan panjang masing-masing p , q , dan r .

Maka, $x = x_1 \dots x_p$, $y = y_1 \dots y_q$, dan $z = z_1 \dots z_r$,

dengan $x_1 \dots x_p, y_1 \dots y_q, z_1 \dots z_r$ adalah simbol-simbol anggota V .

$$\begin{aligned}
\text{Sehingga : } (xy)z &= (x_1 \dots x_p y_1 \dots y_q)z_1 \dots z_r \\
&= x_1 \dots x_p y_1 \dots y_q z_1 \dots z_r \\
&= x_1 \dots x_p (y_1 \dots y_q z_1 \dots z_r) \\
&= x(yz)
\end{aligned}$$

c. Memiliki elemen identitas, yaitu untai hampa \wedge .

Sedemikian sehingga $x \wedge = \wedge x = x$, untuk setiap untai x anggota V^* .

2.1.6. Eksponensiasi pada Untai

Untuk setiap untai x dan setiap bilangan bulat nonnegatif i , untai x^i didefinisikan secara induktif sebagai:

$$\begin{aligned}
x^0 &= \wedge, && \text{yaitu untai hampa} \\
x^{i+1} &= x^i \circ x, && \text{untuk setiap } i = 0, 1, 2, \dots
\end{aligned}$$

Sehingga, untuk $i = 0$, didapat $x^1 = x$. Demikian pula dengan $(do)^2$, berdasarkan definisi di atas (dengan $i = 1$), didapat:

$$\begin{aligned}
(do)^2 &= (do)^1 \circ do && \text{dengan } i = 1 \\
&= ((do)^0 \circ do) \circ do && \text{dengan } i = 0 \\
&= (\wedge \circ do) \circ do \\
&= do \circ do \\
&= dodo
\end{aligned}$$

2.1.7. Subuntai, Prefiks, dan Sufiks

Suatu untai v dikatakan subuntai dari untai w jika dan hanya jika terdapat untai x dan y sedemikian sehingga $w = xzy$. Contohnya, jika diberikan untai $z =$ masuk dan untai $w =$ pemasukan maka terdapat untai $x =$ pe dan untai $y =$ an sedemikian sehingga $w =$ pemasukan $w = xzy$. Dengan kata lain, untai z adalah subuntai dari untai w .

Untai x dan y dapat pula berupa untai hampa \wedge baik sendiri-sendiri maupun secara bersamaan, sehingga setiap untai adalah subuntai dari dirinya sendiri. Dengan mengambil $x = w$ dan $z = y = \wedge$, terlihat bahwa $w = w \wedge \wedge =$

xzy , sehingga untai hampa \wedge adalah subuntai dari setiap untai. Suatu untai bisa saja memiliki subuntai yang sama. Contohnya, untai $ababab$ memiliki tiga kali kemunculan untai ab dan dua kali kemunculan untai $abab$.

Jika $w = zy$ untuk suatu y , maka z disebut awalan atau prefiks dari w . Sedangkan, jika $w = xz$ untuk suatu x , maka z disebut akhiran atau sufiks dari w . Contohnya, jika diberikan $z = \text{mata}$ dan $w = \text{matahari}$, maka terdapat $y = \text{hari}$ sedemikian sehingga $w = \text{matahari} = zy$, maka z adalah awalan dari w , sedangkan jika diberikan $w = \text{kacamata}$, maka terdapat $x = \text{kaca}$ sedemikian sehingga $w = \text{kacamata} = xz$, maka z adalah akhiran dari w .

2.1.8. Operasi antar Dua Himpunan Untai

Misalkan A dan B adalah sub himpunan dari V^* , dengan V adalah suatu alfabet. Perangkaian dari A dan B , dinotasikan dengan AB , adalah himpunan dari semua untai yang berbentuk xy dengan x adalah untai anggota A dan y adalah untai anggota B . Contohnya, jika diberikan himpunan $A = \{0, 11\}$ dan $B = \{1, 10, 110\}$, maka $AB = \{01, 010, 0110, 111, 1110, 11110\}$ dan $BA = \{10, 111, 100, 1011, 1100, 11011\}$. Dari contoh tersebut terlihat bahwa $AB \neq BA$.

Berdasarkan definisi perangkaian dua himpunan untai di atas, dapat didefinisikan A^n untuk $n = 0, 1, 2, \dots$ secara rekursif sebagai berikut:

$$A^0 = \{\wedge\}$$

$$A^{n+1} = A^n A, \quad \text{untuk } n = 0, 1, 2, \dots$$

Contohnya, jika diberikan $A = \{1, 00\}$, maka:

$$A^0 = \{\wedge\}$$

$$A^1 = A^0 A = \{\wedge\} A = A = \{1, 00\}$$

$$A^2 = A^1 A = \{1, 00\} A = \{11, 100, 001, 0000\}$$

$$A^3 = A^2 A = \{11, 100, 001, 0000\} A$$

$$= \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$$

2.2. Pencocokan Untai

Diberikan sebuah untaian $y = y_1y_2 \dots y_n$ dengan panjang n dan sebuah untaian $x = x_1x_2 \dots x_m$ dengan panjang m , $m \leq n$, masing-masing sepanjang alfabet V yang berukuran r . Maka, masalah pencocokan untaian eksak adalah menemukan apakah untaian x ada di dalam untaian y dan jika ada akan dicari semua posisi kemunculan untaian x pada untaian y tersebut. Contohnya, jika diberikan $y = atcgtcacgtc$ dan $x = cgt$ maka untaian x ada di dalam untaian y yaitu:

$$y = \underline{atcgtcacgtc}$$

Dalam hal ini, untaian x disebut juga sebagai subuntaian dari y . Sedangkan jika diberikan untaian lain yaitu $z = agt$ maka untaian z tidak ada di dalam untaian y . Atau manakah untaian y yang paling mendekati x dan di mana posisinya. Hal inilah yang disebut pencocokan hampiran untaian (*approximate string matching*) (Tretyakov, 2003).

2.3. Jarak

Definisi jarak secara umum, diberikan suatu himpunan tak kosong H dan dua buah elemen a dan b anggota himpunan H tersebut. Jarak antara a dan b yang dinotasikan dengan $D(a, b)$ didefinisikan sebagai fungsi d yang memetakan (a, b) ke suatu bilangan real nonnegatif, atau $D: H \times H \rightarrow \mathbb{R}^+ \cup \{0\}$, yang memenuhi sifat-sifat berikut:

- a. $D(a, b) \geq 0$
- b. $D(a, b) = 0$ jika $a = b$
- c. $D(a, b) = D(b, a)$
- d. $D(a, c) \leq D(a, b) + D(b, c)$ untuk c anggota H

(Bartle & Sherbert, 2000)

2.4. Jarak Hamming

Definisi 2.4.1

Jika diberikan dua buah untai $x = x_1x_2 \dots x_n$ dan $y = y_1y_2 \dots y_n$ dengan panjang yang sama, misalkan $|x| = |y| = n$, sepanjang alfabet V berukuran r sedemikian sehingga x dan y anggota dari V^* , maka jarak Hamming antara untai x dan untai y tersebut adalah fungsi HD yang memetakan (x, y) ke suatu bilangan real nonnegatif, atau $HD: V^* \times V^* \rightarrow \mathbb{R} + \{0\}$ yang didefinisikan sebagai berikut:

$$\begin{aligned} HD(x, y) &= d(x_1, y_1) + d(x_2, y_2) + \dots + d(x_n, y_n) \\ &= \sum_{i=1}^n d(x_i, y_i) \end{aligned} \quad (3.1)$$

dengan : $x_i, y_i \in V$ untuk $i = 1, 2, \dots, n$

$$d(x_i, y_i) = 0 \quad \text{jika } x_i = y_i \text{ dan}$$

$$d(x_i, y_i) = 1 \quad \text{jika } x_i \neq y_i$$

Berdasarkan definisi di atas, dapat diartikan bahwa jarak Hamming antara untai x dan untai y yang memiliki panjang yang sama merupakan banyaknya karakter pada untai x yang berbeda dengan karakter pada untai y yang berada pada posisi yang bersesuaian. Dengan kata lain, jumlah minimum penggantian karakter yang diperlukan untuk merubah untai x menjadi untai y ataupun sebaliknya.

Contoh 2.1.

Jika diberikan untai $x = \text{gelas}$ dan $y = \text{beras}$, terlihat bahwa $|x| = |y| = 5$, maka jarak Hamming antara kedua untai tersebut adalah

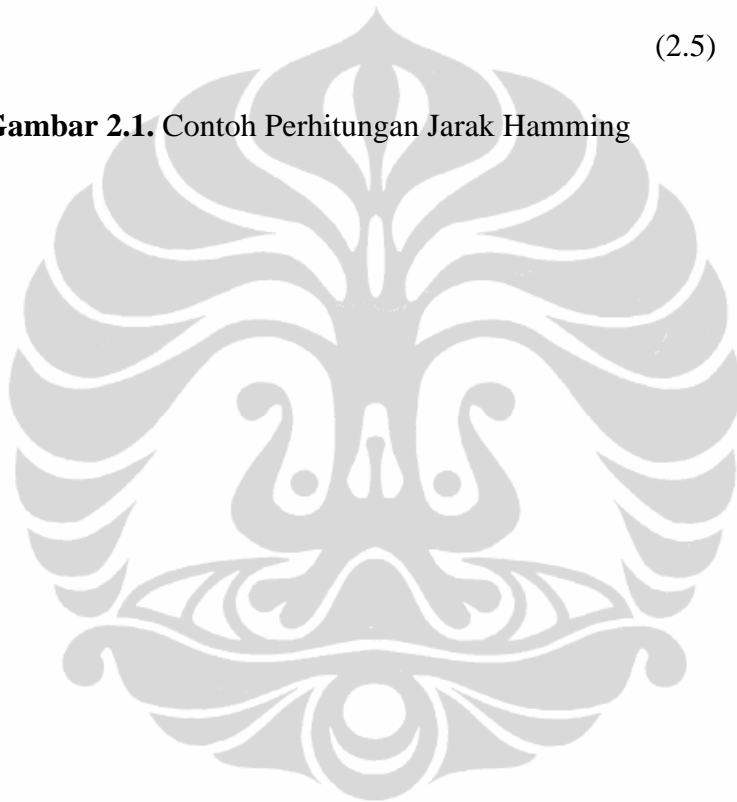
$$\begin{aligned} HD(x, y) &= \sum_{i=1}^5 d(x_i, y_i) \\ &= d(x_1, y_1) + d(x_2, y_2) + d(x_3, y_3) + d(x_4, y_4) + d(x_5, y_5) \\ &= d(g, b) + d(e, e) + d(l, r) + d(a, a) + d(s, s) \\ &= 1 + 0 + 1 + 0 + 0 \\ &= 2 \end{aligned}$$

Jadi, jarak Hamming antara untai $x = \text{gelas}$ dan $y = \text{beras}$ adalah $HD(x, y) = 2$ karena $x_1 = g \neq b = y_1$ dan $x_3 = l \neq r = y_3$.

$i :$	1	2	3	4	5
$x =$	g	e	l	a	s
$y =$	b	e	r	a	s

(2.5)

Gambar 2.1. Contoh Perhitungan Jarak Hamming



BAB 3

PENCOCOKAN HAMPIRAN UNTAI DENGAN UKURAN JARAK LEVENSZTEIN

Jarak Levenshtein dikenal setelah seorang ilmuwan Rusia Vladimir Levenshtein membangun algoritma mengenai pencocokkan untai pada tahun 1965. Jarak Levenshtein $D(x, y)$ antara dua buah untai x dan y ($m = |x|, n = |y|$) adalah banyaknya operasi minimum dari penyisipan, penghapusan atau penggantian yang dilakukan untuk menyamakan x dan y . Penyisipan (*insertion*) adalah menyisipkan suatu karakter baru pada x . Misal $x = vw$ setelah disisipkan suatu karakter a maka menjadi $x' = vaw$. Penghapusan (*deletion*) adalah menghapus suatu karakter pada x . Misalkan $x = vaw$, setelah dilakukan penghapusan satu karakter a maka menjadi $x' = vw$. Penggantian (*Replacement*) adalah mengganti suatu karakter di x sama dengan di y . Misalkan $x = vaw$, setelah dilakukan penggantian a dengan suatu karakter b maka menjadi $x' = vbw$ (Pinzõn, 2006).

3.1. Jarak Levenshtein

Diberikan dua buah untai $x = x(1)x(2)x(3) \dots x(m)$ dan $y = y(1)y(2)y(3) \dots y(n)$ dengan $|x| = m$ dan $|y| = n$ sepanjang alfabet V berukuran r sedemikian sehingga x dan y anggota dari V^* . $x(j)$ adalah karakter pada posisi ke- j pada untai x dan $y(i)$ adalah karakter pada posisi ke- i pada untai y .

Definisi 3.1.

Jarak Levenshtein antara untai x dan untai y tersebut adalah fungsi D yang memetakan (x, y) ke suatu bilangan real nonnegatif, atau $: V^* \times V^* \rightarrow \mathbb{R} + \{0\}$. Dengan cara menyusun karakter yang cocok antara untai x dan y dan menyisipkan untai-untai hampa (\wedge) sedemikian sehingga panjang x' dan y' menjadi sama, misalkan $|x'| = |y'| = l$, maka jarak Levenshtein didefinisikan sebagai berikut:

$$\begin{aligned}
 D(x, y) &= d(x_1, y_1) + d(x_2, y_2) + \dots + d(x_l, y_l) \\
 &= \sum_{i=1}^l d(x_i, y_i)
 \end{aligned} \tag{3.1}$$

dengan : $x_i, y_i \in V$ untuk $i = 1, 2, \dots, l$

$$d(x_i, y_i) = 0 \quad \text{jika } x_i = y_i \text{ dan}$$

$$d(x_i, y_i) = 1 \quad \text{jika } x_i \neq y_i$$

Berdasarkan definisi di atas, dapat diartikan bahwa jarak Levensthein adalah seperti jarak Hamming antara untai x' dan untai y' yang memiliki panjang yang sama merupakan banyaknya karakter pada untai x' yang berbeda dengan karakter pada untai y' yang berada pada posisi yang bersesuaian. Dengan kata lain, jumlah minimum penggantian karakter yang diperlukan untuk merubah untai x' menjadi untai y' ataupun sebaliknya. Dalam hal ini jika merubah karakter \wedge pada untai x' menjadi suatu karakter yang bukan karakter hampa (\wedge) pada untai y' maka berarti proses penyisipan. Jika merubah suatu karakter bukan \wedge pada untai x' dengan karakter \wedge maka berarti proses penghapusan. Jika merubah suatu karakter bukan \wedge pada untai x' dengan suatu karakter yang bukan \wedge dari untai y' maka berarti proses penggantian.

Contoh 3.1.

Jika diberikan untai $x = \text{tabel}$ dan $y = \text{able}$, dengan $|x| = 5, |y| = 4$,

$$x' = \text{tab}^{\wedge}\text{el}$$

$$y' = ^{\wedge}\text{able}^{\wedge}$$

Terlihat bahwa $|x'| = |y'| = 6$,

maka jarak Levensthein antara x dan y adalah sama dengan jarak Hamming antara x' dan y' tersebut adalah

$$\begin{aligned}
 DH(x, y) &= \sum_{i=1}^6 d(x_i, y_i) \\
 &= d(x_1, y_1) + d(x_2, y_2) + d(x_3, y_3) + d(x_4, y_4) + d(x_5, y_5) \\
 &\quad + d(x_6, y_6) \\
 &= d(t, ^{\wedge}) + d(a, a) + d(b, b) + d(^{\wedge}, l) + d(e, e) + d(l, ^{\wedge}) \\
 &= 1 + 0 + 0 + 1 + 0 + 1 \\
 &= 3
 \end{aligned}$$

Jadi, jarak Levenshtein antara untai $x = \text{tabel}$ dan $y = \text{able}$ adalah $D(x, y) = 3$ karena penghapusan t pada $x'(1)$, penyisipan l pada $x'(4)$, dan penghapusan l pada $x'(6)$.

Selanjutnya, karena jarak Levenshtein dapat dimodifikasi menjadi jarak Hamming maka secara otomatis jarak Levenshtein memenuhi sifat-sifat sebagai jarak sebagaimana jarak Hamming yang telah dibuktikan di bab 2. Jadi berlaku sifat-sifat :

- a. $D(x, y) \geq 0$
- b. $D(x, y) = 0$ jika $x = y$
- c. $D(x, y) = D(y, x)$
- d. $D(x, z) \leq D(x, y) + D(y, z)$

3.2. Pencocokkan Hampiran Untai dengan Ukuran Jarak Levenshtein

Pada proses pencocokkan dua buah untai, untuk setiap dua karakter yang dicocokkan apabila ternyata berbeda maka akan ada tiga kemungkinan operasi selanjutnya yaitu penyisipan, penghapusan atau penggantian (Pinzõn,2006).

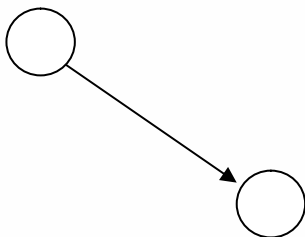
Misalkan operasi penghapusan dilambangkan dengan garis horizontal:



Operasi penyisipan dilambangkan dengan garis vertikal:



Serta operasi penggantian dilambangkan dengan garis diagonal:



Selanjutnya proses pencocokkan hampiran untai dapat digambarkan sebagai berikut :

Misalkan terdapat 2 buah untai untai yang akan dicocokkan:

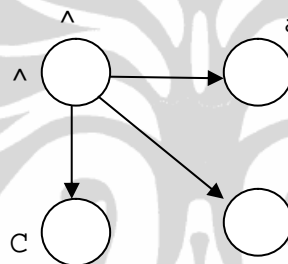
$$x = ab \quad \text{dimana } |x| = 2$$

$$y = c \quad \text{dimana } |y| = 1$$

$x(i)$ adalah posisi ke- i pada x dan $y(i)$ adalah posisi ke- i pada y .

Mula-mula dicocokkan karakter pada $x(1)$ dengan karakter pada $y(1)$, karakter pada $x(1) \neq y(1)$, maka ada 3 kemungkinan operasi yang dapat dilakukan yaitu menghapus a pada $x(1)$, menyisipkan c pada $y(1)$ menjadi $x(1)$ atau mengganti

a pada $x(1)$ dengan c pada $y(1)$.



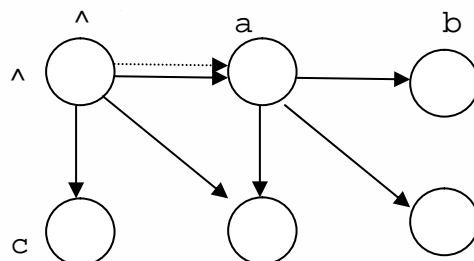
Gambar 3.1 Graf pencocokkan a pada $x(1)$ dengan c pada $y(1)$

Jika pilihannya adalah menghapus a maka

$$x' = b$$

$$y = c$$

selanjutnya membandingkan b pada $x'(1)$ dan c pada $y(1)$ ada tiga pilihan yaitu menghapus b pada $x'(1)$, menyisipkan c pada $y(1)$ menjadi $x'(1)$ atau mengganti b pada $x'(1)$ dengan c pada $y(1)$.

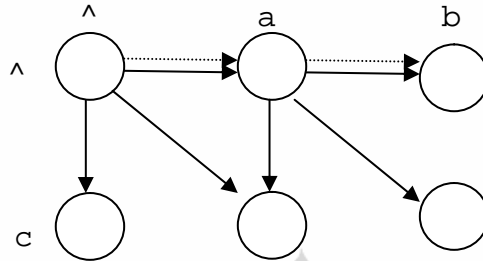


Gambar 3.2 Graf pencocokkan b pada $x(1)$ dengan c pada $y(1)$

Jika pilihannya adalah menghapus b maka

$$x'' =$$

$$y = c$$

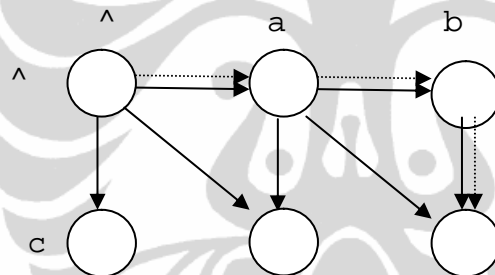


Gambar 3.3 Graf proses penghapusan b pada $x(2)$ atau pada $x'(1)$

Selanjutnya harus dilakukan penyisipan c pada $y(1)$ di $x''(1)$ sehingga menjadi

$$x''' = c$$

$$y = c$$

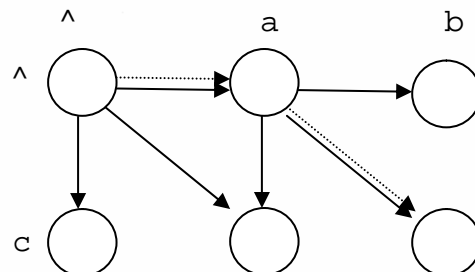


Gambar 3.4 Graf proses penyisipan c pada $y(1)$ atau pada $x''(1)$

Jika pilihannya adalah mengganti b pada $x'(1)$ dengan c pada $y(1)$, maka

$$x'' = c$$

$$y = c$$

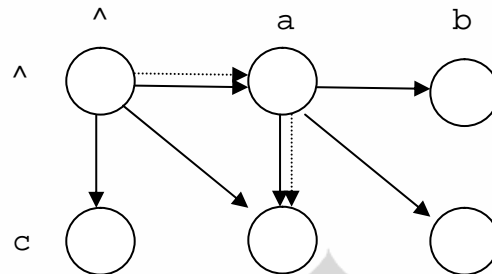


Gambar 3.5 Graf proses penggantian b pada $x'(1)$ atau pada $x(2)$ dengan c pada $y(1)$

Jika pilihannya adalah menyisipkan c pada $x'(1)$ maka

$$x'' = cb$$

$$y = c$$

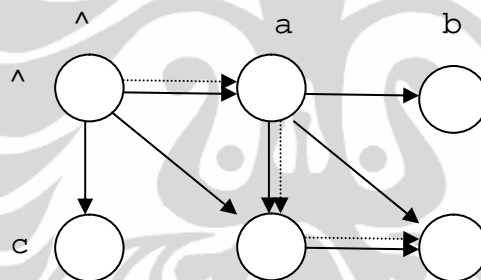


Gambar 3.6 Graf proses penyisipan c pada $x'(1)$ menjadi $x''(1)$

Selanjutnya harus dilakukan penghapusan b pada $x''(2)$ sehingga

$$x''' = c$$

$$y = c$$

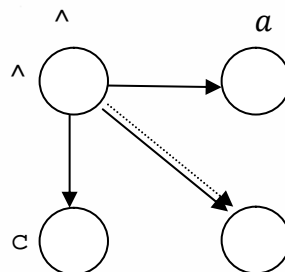


Gambar 3.7 Graf proses penghapusan b pada $x''(2)$

Jika pilihan sebelumnya adalah mengganti a pada $x(1)$ dengan c pada $y(1)$ maka

$$x'' = cb$$

$$y = c$$

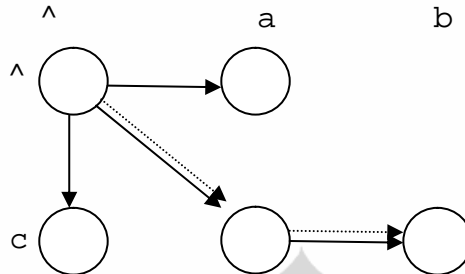


Gambar 3.8 Graf proses penggantian a pada $x(1)$ dengan c pada $y(1)$

Langkah selanjutnya haruslah menghapus b pada $x'(2)$ maka

$$x'' = c$$

$$y = c$$

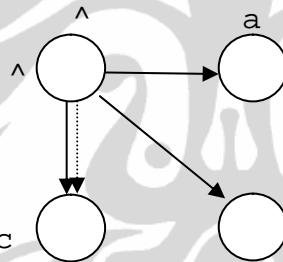


Gambar 3.9 Graf proses penghapusan b pada $x'(2)$

Jika pilihan sebelumnya adalah menyisipkan c pada $x(1)$ maka

$$x' = cab$$

$$y = c$$

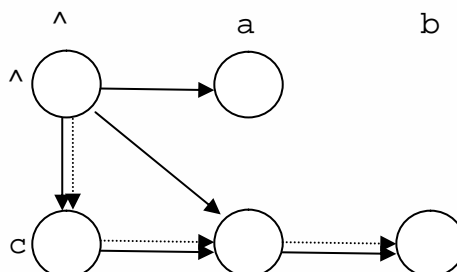


Gambar 3.10 Graf proses penyisipan c pada $y(1)$ menjadi c pada $x'(1)$

Langkah selanjutnya haruslah menghapus a pada $x'(2)$ dan b pada $x'(3)$ sehingga

$$x'' = c$$

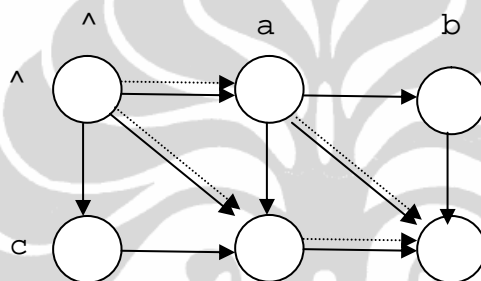
$$y = c$$



Gambar 3.11 Graf proses penghapusan a pada $x'(2)$ dan b pada $x'(3)$

Jika dimisalkan setiap operasi penyisipan, penghapusan dan penggantian karakter yang berbeda diberikan biaya 1 dan 0 jika karakter yang dicocokkan sama. Sehingga proses pencocokkan untai pada gambar 3.4 mempunyai total biaya 3. Pada gambar 3.5 mempunyai total biaya 2. Pada gambar 3.7 mempunyai total biaya 3. Pada gambar 3.9 mempunyai total biaya 2. Dan pada gambar 3.11 mempunyai total biaya 3. Total biaya terkecilnya adalah 2 sehingga jarak Levenshteinnya adalah 2.

Secara keseluruhan proses pencocokkan kedua untai $x = ab$ dan $y = c$ tersebut berupa lintasan-lintasan antara dua simpul dan dapat digambarkan sebagai berikut:



Gambar 3.12 Graf keseluruhan proses pencocokkan antara $x = ab$ dan $y = c$

Pada gambar 3.12 proses pencocokkan hampiran untai antara $x = ab$ dan $y = c$ menghasilkan jarak Levenshtein adalah 2. Proses pencocokkannya ada dua alternatif terlihat sebagai garis putus-putus. Proses pencocokkannya adalah menghapus a pada $x(1)$ dan mengganti b pada $x(2)$ dengan c pada $y(2)$, atau mengganti a pada $x(1)$ dengan c pada $y(1)$ dan menghapus b pada $x(2)$.

3.3. Pencocokkan Hampiran Untai dengan Ukuran Jarak Levenshtein Menggunakan Program Dinamik

3.3.1. Program Dinamik

Program dinamik merupakan sebuah algoritma pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan. Sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian

keputusan yang saling berkaitan. Dan keputusan dibuat secara berurutan. Program Dinamik memperlakukan persoalan di mana keputusan pada satu tahap mempengaruhi keputusan tahap berikutnya. Sehingga, setiap submasalah, yang dapat berubah dari tahap ke tahap adalah suatu fungsi dari keputusan tahap sebelumnya (Mulyono,2007). Ciri-ciri pokok masalah program dinamik adalah :

- a. Dalam masalah program dinamik, keputusan tentang suatu masalah ditandai dengan optimasi pada tahap berikutnya, bukan keserentakan. Berarti setiap masalah yang akan diselesaikan dengan program dinamik, harus dipecah menjadi n submasalah.
- b. Program dinamik berkaitan dengan masalah-masalah di mana pilihan atau keputusan dibuat pada masing-masing tahap. Seluruh kemungkinan pilihan dicerminkan, diatur atau keduanya, oleh system *state* pada setiap tahap.
- c. Berkaitan dengan setiap keputusan pada setiap tahap adalah mengevaluasi pilihan yang dibuat dalam arti sumbangan yang diberikan kepada tujuan keseluruhan (maksimisasi atau minimisasi).
- d. Pada setiap tahap proses keputusan dihubungkan dengan tahap yang berdekatan melalui fungsi transisi. Fungsi ini dapat berupa kuantitas yang diskrit maupun kontinu tergantung pada sifat-sifat masalah.
- e. Suatu hubungan rekursif digunakan untuk menghubungkan kebijakan optimum pada tahap n dengan $n-1$. Ada dua macam prosedur rekursif yaitu *foreward* dan *backward*.
- f. Dengan menggunakan hubungan rekursif tersebut, maka prosedur penyelesaian bergerak dari tahap ke tahap sampai kebijakan optimum tahap terakhir ditemukan. Sekali kebijakan optimum tahap n ditemukan, n komponen keputusan dapat ditemukan kembali dengan melacak balik melalui fungsi transisi tahap n .

3.3.2 Pencocokkan Hampiran Untai dengan Program Dinamik

Dari pembahasan sebelumnya dan dari uraian mengenai program dinamik maka masalah pencocokkan hampiran unta ini dapat dikategorikan sebagai permasalahan program dinamik, khususnya serupa dengan masalah mencari

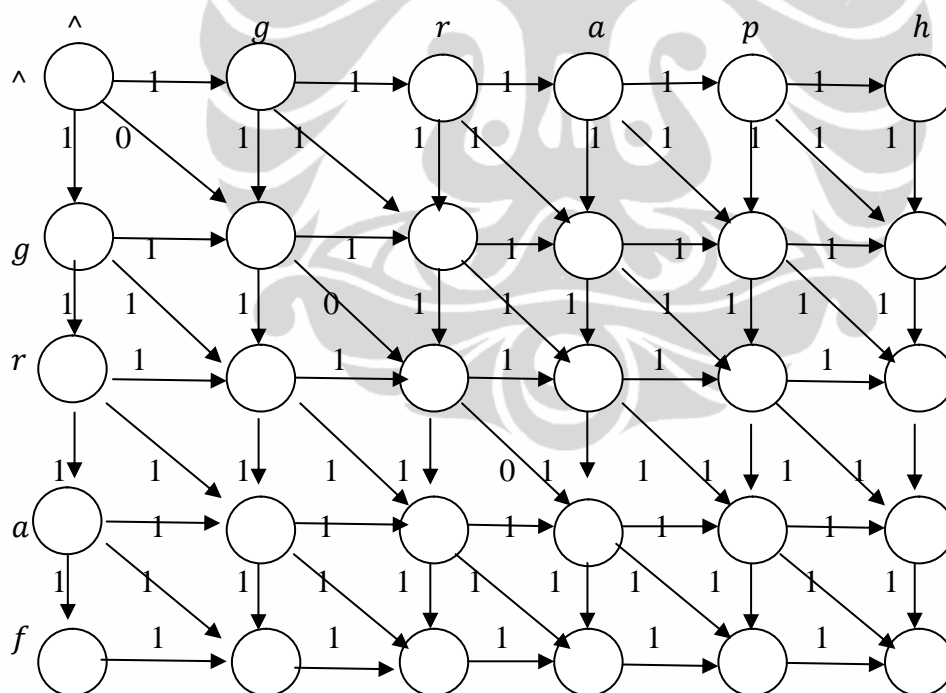
lintasan terpendek dari dua buah simpul pada graf. Penggambaran masalah pencocokkan hampiran unta ini dalam bentuk graf terlihat mempunyai struktur yang sangat kas. Selanjutnya dicari suatu lintasan terpendek dari simpul pada pojok kiri atas dengan simpul pada pojok kanan bawah. Lintasan terpendek adalah suatu lintasan yang mempunyai total bobot minimal. Dengan melalui prosedur perhitungan rekursif tahap maju kemudian dilanjutkan dengan tahap mundur maka akan diperoleh suatu lintasan yang mempunyai bobot minimal. Lintasan tersebut menggambarkan proses pencocokkan kedua unta. Dan akan terlihat pula ada atau tidaknya solusi alternatif.

Misalkan akan dilakukan pencocokkan dua unta :

$x = graph$ di mana $|x| = 5$

$y = graf$ di mana $|y| = 4$

Graf yang menggambarkan secara keseluruhan kemungkinan proses pencocokkan unta dan biaya setiap operasinya adalah sebagai berikut :



Gambar 3.13 Graf penggambaran keseluruhan kemungkinan proses pencocokkan antara $x = graph$ dan $y = graf$

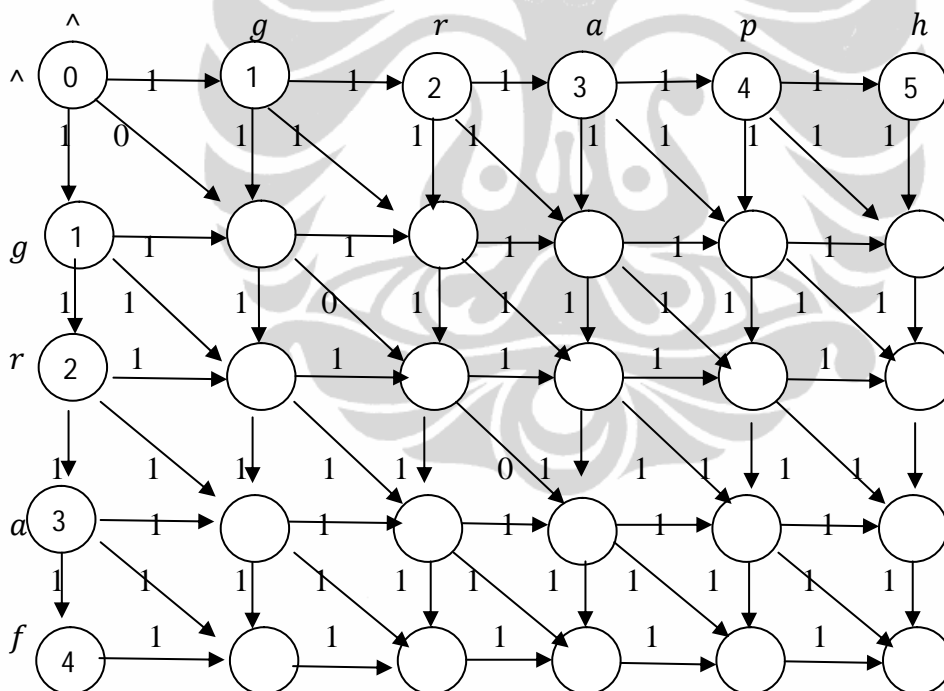
Nilai yang ada di dalam simpul adalah menunjukkan jarak terdekat (bobot minimal) simpul tersebut dengan simpul awal. Adapun cara untuk menentukannya adalah sebagai berikut:

$x = graph$ dimana $|x| = 5$ dengan $x(0) = \wedge, x(1) = g, x(2) = r, x(3) = a, x(4) = p, x(5) = h$, $x(i)$ adalah karakter pada untai x pada posisi ke- i , di mana $i = 0, 1, 2, 3, 4, 5$. Dalam graf di bawah i merupakan indeks untuk kolom.

$y = graf$ dimana $|y| = 4$, dengan $y(0) = \wedge, y(1) = g, y(2) = r, y(3) = a, y(4) = f$, $y(i)$ adalah karakter pada untai y pada posisi ke- j , di mana $j = 0, 1, 2, 3, 4$. Dalam graf di bawah j merupakan indeks untuk baris.

$D[j, i]$ adalah bobot minimum dari lintasan terpendek antara simpul $[0, 0]$ dan simpul $[j, i]$.

$D[0, 0] = 0, D[0, 1] = 1, D[0, 2] = 2, D[0, 3] = 3, D[0, 4] = 4, D[0, 5] = 5$
 $D[1, 0] = 1, D[2, 0] = 2, D[3, 0] = 3, D[4, 0] = 4$



Gambar 3.14 Graf pengisian titik-titik $D[j, 0]$ dan $D[0, i]$

Selanjutnya pengisian nilai pada tiap simpulnya adalah sebagai berikut:

$i = 1$

$j = 1$

apakah $x(1) = y(1)$? ya, karena $g = g$, maka

$$\begin{aligned}
 D[1,1] &= \min(D[0,0], D[1,0], D[0,1]) + 0 \\
 &= \min(0, 1, 1) + 0 \\
 &= 0
 \end{aligned}$$

$j = 2$

apakah $x(1) = y(2)$? tidak, karena $g \neq r$, maka

$$\begin{aligned}
 D[2,1] &= \min(D[1,1], D[2,0], D[1,0]) + 1 \\
 &= \min(0, 2, 1) + 1 \\
 &= 1
 \end{aligned}$$

$j = 3$

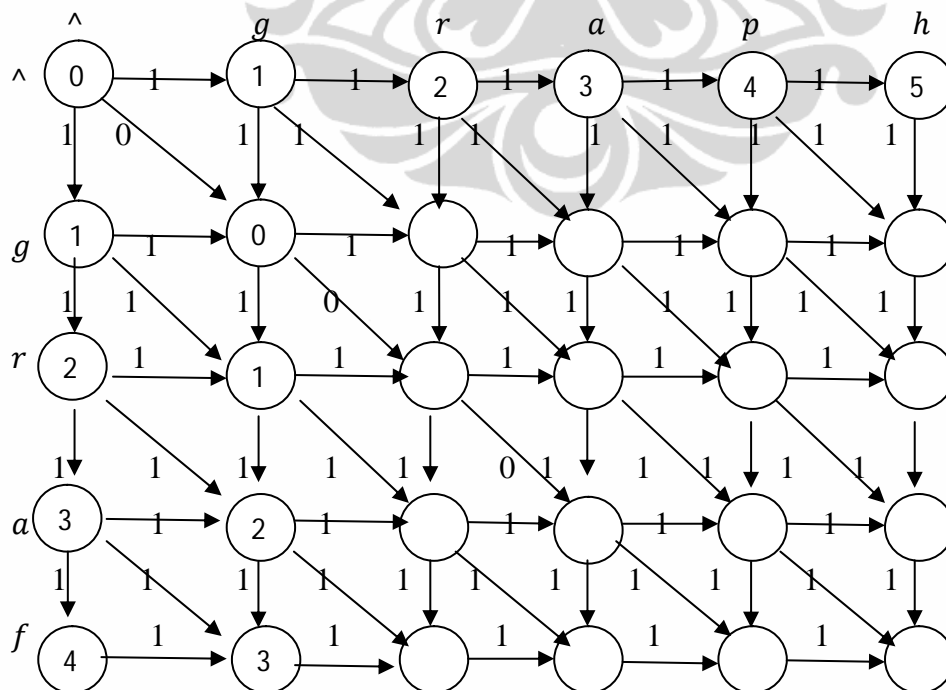
apakah $x(1) = y(3)$? tidak, karena $g \neq a$, maka

$$\begin{aligned}
 D[3,1] &= \min(D[2,1], D[3,0], D[2,0]) + 1 \\
 &= \min(1, 3, 2) + 1 \\
 &= 2
 \end{aligned}$$

$j = 4$

apakah $x(1) = y(4)$? tidak, karena $g \neq f$, maka

$$\begin{aligned}
 D[4,1] &= \min(D[3,1], D[4,0], D[3,0]) + 1 \\
 &= \min(2, 4, 3) + 1 \\
 &= 3
 \end{aligned}$$



Gambar 3.15 Graf pengisian titik-titik $D[j, 1]$

$$i = 2$$

$$j = 1$$

apakah $x(2) = y(1)$? tidak, karena $r \neq g$, maka

$$\begin{aligned} D[1,2] &= \min(D[1,1], D[0,2], D[0,1]) + 1 \\ &= \min(0, 2, 1) + 1 \\ &= 1 \end{aligned}$$

$$j = 2$$

apakah $x(2) = y(2)$? ya, karena $r = r$, maka

$$\begin{aligned} D[2,2] &= \min(D[1,1], D[1,2], D[2,1]) + 0 \\ &= \min(0, 1, 1) + 0 \\ &= 0 \end{aligned}$$

$$j = 3$$

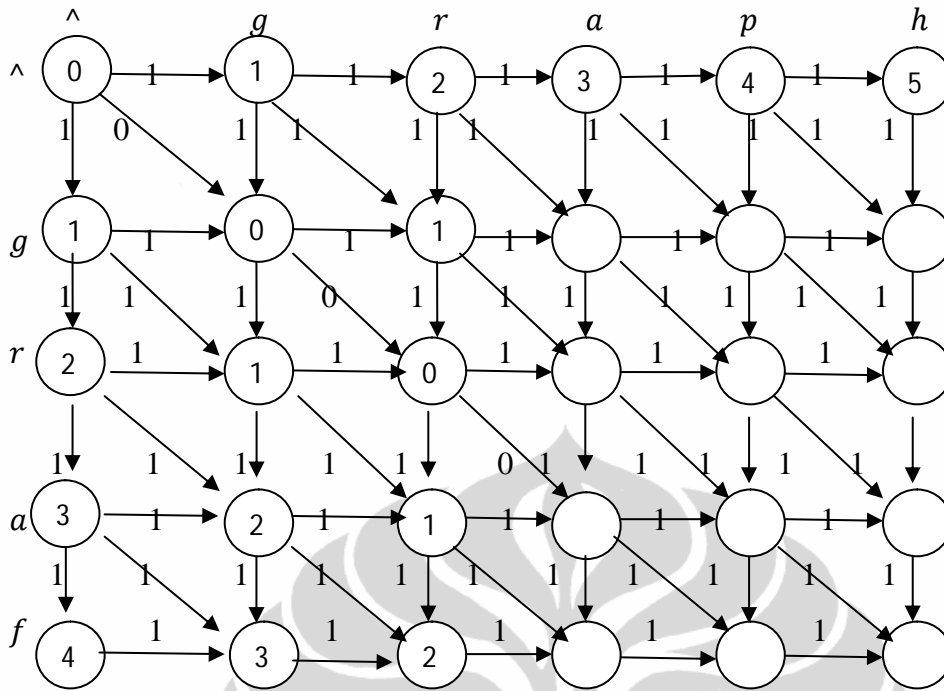
apakah $x(2) = y(3)$? tidak, karena $r \neq a$, maka

$$\begin{aligned} D[3,2] &= \min(D[2,2], D[3,1], D[2,1]) + 1 \\ &= \min(0, 2, 1) + 1 \\ &= 1 \end{aligned}$$

$$j = 4$$

apakah $x(2) = y(4)$? tidak, karena $r \neq f$, maka

$$\begin{aligned} D[4,2] &= \min(D[3,2], D[4,1], D[3,1]) + 1 \\ &= \min(1, 3, 2) + 1 \\ &= 2 \end{aligned}$$



Gambar 3.16 Graf pengisian titik-titik $D[j, 2]$

$i = 3$

$j = 1$

apakah $x(3) = y(1)$? tidak, karena $a \neq g$, maka

$$\begin{aligned} D[1,3] &= \min(D[0,3], D[1,2], D[0,2]) + 1 \\ &= \min(3, 1, 2) + 1 \\ &= 2 \end{aligned}$$

$j = 2$

apakah $x(3) = y(2)$? tidak, karena $a \neq r$, maka

$$\begin{aligned} D[2,3] &= \min(D[2,2], D[1,3], D[1,2]) + 1 \\ &= \min(0, 2, 1) + 1 \\ &= 1 \end{aligned}$$

$j = 3$

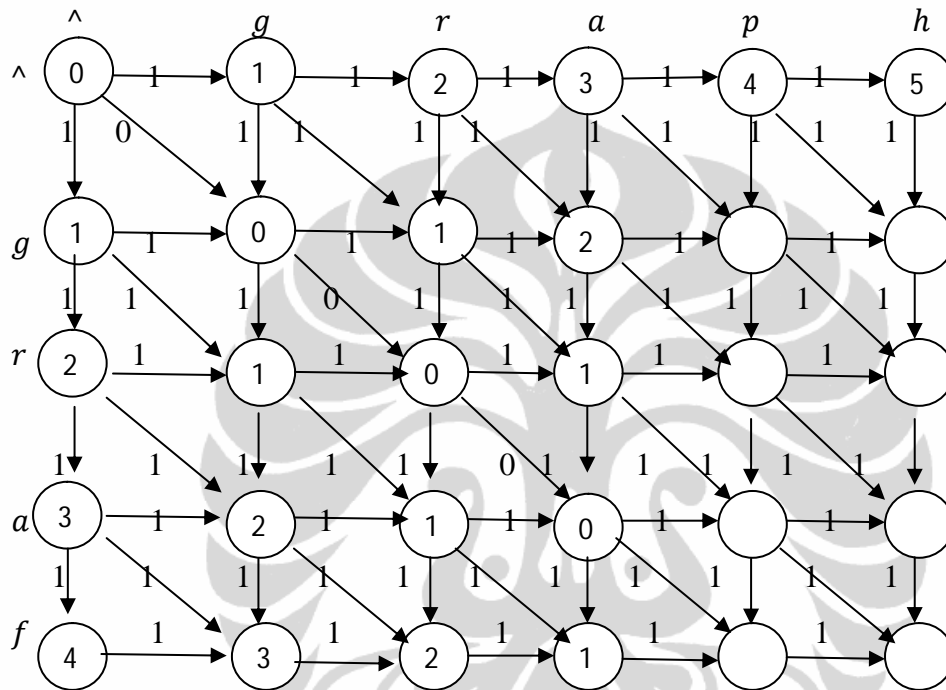
apakah $x(3) = y(3)$? ya, karena $a = a$, maka

$$\begin{aligned} D[3,3] &= \min(D[2,2], D[2,3], D[3,2]) + 1 \\ &= \min(0, 1, 1) + 0 \\ &= 0 \end{aligned}$$

$$j = 4$$

apakah $x(3) = y(4)$? tidak, karena $a \neq f$, maka

$$\begin{aligned} D[4,3] &= \min(D[3,3], D[2,4], D[2,3]) + 1 \\ &= \min(0, 2, 1) + 1 \\ &= 1 \end{aligned}$$



Gambar 3.17 Graf pengisian titik-titik $D[j, 3]$

$$i = 4$$

$$j = 1$$

apakah $x(4) = y(1)$? tidak, karena $p \neq g$, maka

$$\begin{aligned} D[1,4] &= \min(D[0,4], D[1,3], D[0,3]) + 1 \\ &= \min(4, 2, 3) + 1 \\ &= 3 \end{aligned}$$

$$j = 2$$

apakah $x(4) = y(2)$? tidak, karena $p \neq r$, maka

$$\begin{aligned} D[2,4] &= \min(D[2,3], D[1,4], D[1,3]) + 1 \\ &= \min(3, 1, 2) + 1 \\ &= 2 \end{aligned}$$

$$j = 3$$

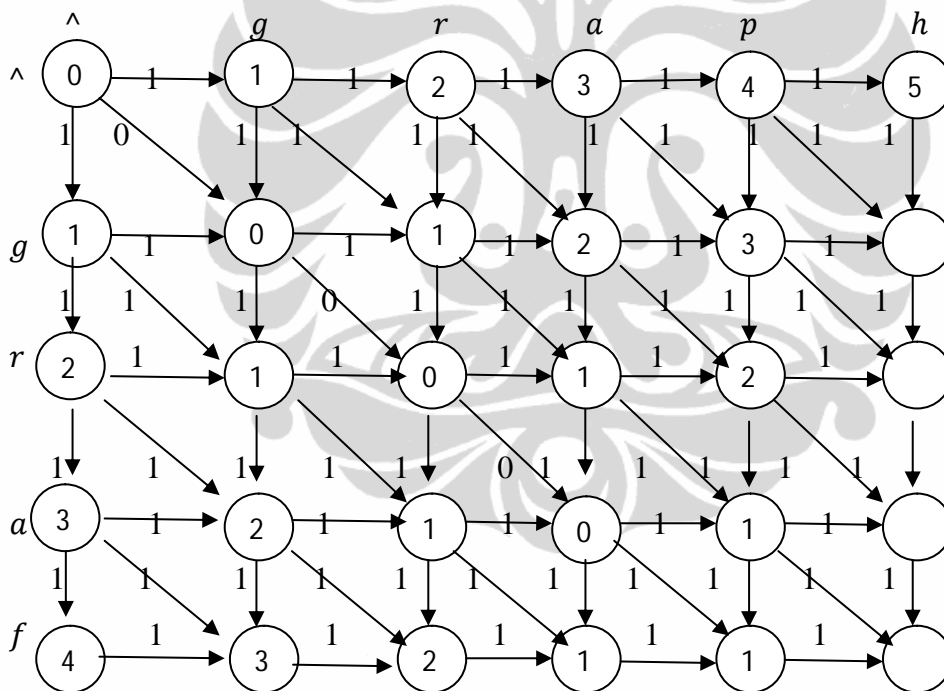
apakah $x(4) = y(3)$? tidak, karena $p \neq a$, maka

$$\begin{aligned} D[3,4] &= \min(D[4,2], D[3,3], D[3,2]) + 1 \\ &= \min(2, 0, 1) + 1 \\ &= 1 \end{aligned}$$

$$j = 4$$

apakah $x(4) = y(4)$? tidak, karena $p \neq f$, maka

$$\begin{aligned} D[4,4] &= \min(D[3,4], D[4,3], D[3,3]) + 1 \\ &= \min(1, 1, 0) + 1 \\ &= 1 \end{aligned}$$



Gambar 3.18 Graf pengisian titik-titik $D[j, 4]$

$$i = 5$$

$$j = 1$$

apakah $x(5) = y(1)$? tidak, karena $h \neq g$, maka

$$\begin{aligned} D[1,5] &= \min(D[0,5], D[1,4], D[0,4]) + 1 \\ &= \min(5, 3, 4) + 1 \\ &= 4 \end{aligned}$$

$$j = 2$$

apakah $x(5) = y(2)$? tidak, karena $h \neq r$, maka

$$\begin{aligned} D[2,5] &= \min(D[2,4], D[1,5], D[1,4]) + 1 \\ &= \min(2, 4, 3) + 1 \\ &= 3 \end{aligned}$$

$$j = 3$$

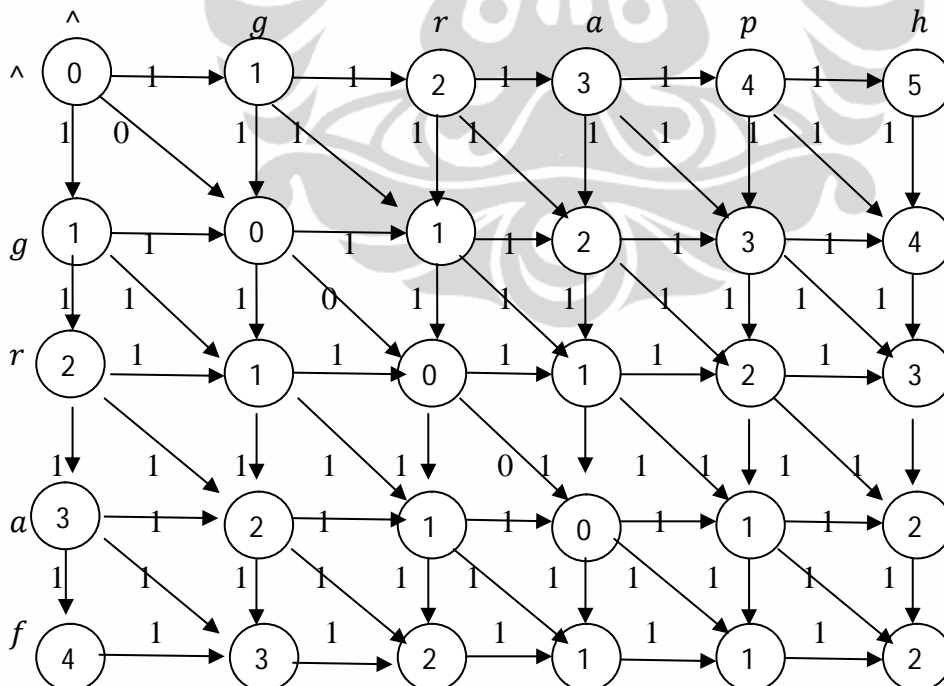
apakah $x(5) = y(3)$? tidak, karena $h \neq a$, maka

$$\begin{aligned} D[3,5] &= \min(D[2,5], D[3,4], D[2,4]) + 1 \\ &= \min(3, 1, 2) + 1 \\ &= 2 \end{aligned}$$

$$j = 4$$

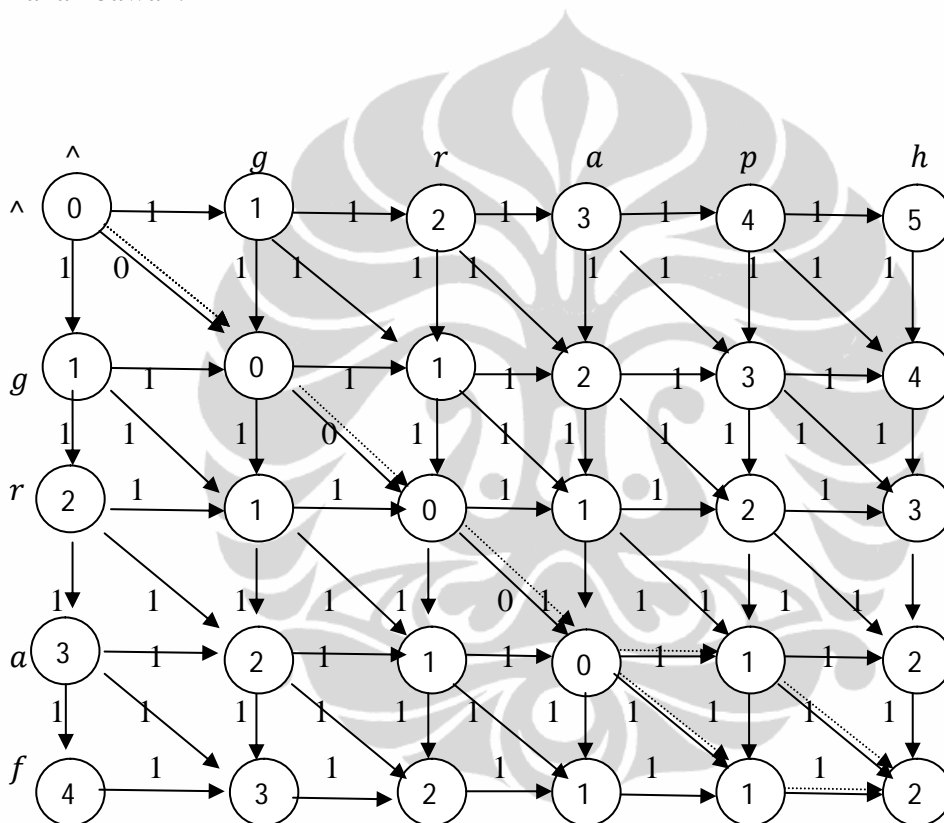
apakah $x(5) = y(4)$? tidak, karena $h \neq f$, maka

$$\begin{aligned} D[5,4] &= \min(D[3,5], D[4,4], D[3,4]) + 1 \\ &= \min(2, 1, 1) + 1 \\ &= 2 \end{aligned}$$



Gambar 3.19 Graf pengisian titik-titik $D[j, 5]$

Selanjutnya dilakukan proses peruntukan balik untuk mendapatkan lintasan terpendek (atau lintasan dengan bobot minimum). Hasilnya adalah berupa lintasan dengan garis putus-putus seperti pada gambar 3.20. Terlihat bahwa solusi dari pencocokkan untai ini ada 2 macam, terlihat dari lintasan yang berupa garis putus-putus terdapat percabangan. Keduanya menunjukkan jarak Levenshtein yang sama yaitu 2 yang dapat dilihat dari simpul di ujung diagonal pada pojok kanan bawah.



Gambar 3.20 Graf hasil proses pencocokkan antara $x = graph$ dan $y = graf$

Karena graf untuk pencocokkan hampiran untai ini adalah mempunyai struktur yang kas, maka bentuk graf dapat disederhanakan menjadi tabel atau matriks berukuran $(5+1)(4+1)$, apabila diambil simpul-simpulnya. Misalkan dari graf pada gambar 3.20 dapat dibuat dalam bentuk tabel sebagai berikut:

	\wedge	g	r	a	p	h
\wedge	0	1	2	3	4	5
g	1	0	1	2	3	4
r	2	1	0	1	2	3
a	3	2	1	0	1	2
f	4	3	2	1	1	2

Tabel 3.1 Tabel hasil proses pencocokkan antara $x = graph$ dan $y = graf$

Persegi yang berwarna menggambarkan lintasan terpendeknya. Jika persegi berwarna berikutnya adalah disamping kanan yaitu persegi $D[j, i + 1]$, artinya dilakukan penghapusan karakter ke- i pada untai $x = graph$. Jika persegi berwarna berikutnya adalah di bawahnya yaitu persegi $D[j + 1, i]$, artinya dilakukan penyisipan karakter ke- j dari untai $y = graf$ pada untai $x = graph$. Jika persegi berwarna berikutnya adalah dibawahnya secara diagonal yaitu persegi $D[j + 1, i + 1]$, artinya dilakukan penggantian karakter ke- j dari $y = graf$ kepada karakter ke- i pada $x = graph$ jika nilai pada persegi adalah 1. Jika nilai pada persegi adalah 0 maka berarti tidak perlu ada penggantian karakter karena keduanya sudah sama.

3.4. Algoritma Pencocokkan Untai Terdekat dengan Program Dinamik

Dari pembahasan sebelumnya maka dapat dibangun suatu algoritma perhitungan bobot minimum tiap simpul atau jika di dalam tabel adalah nilai tiap-tiap persegi sampai akhirnya diperoleh nilai dari jarak Levenshtein pada ujung diagonal yaitu persegi pojok kanan bawah yaitu persegi $D[n, m]$. Berikut ini adalah algoritma pencocokkan unta terdekat dengan program dinamik.

Algoritma program dinamik untuk jarak Levenshtein**Input** : Untai x dengan panjang m dan untaian y dengan panjang n .**Output** : Nilai jarak Levenshtein

```

1  procedure  $D[x, y]$           { $m = |x|, n = |y|$ }
2  begin
3      for  $i := 0$  to  $m$  do  $D[0, i] = i$ 
4      for  $j := 0$  to  $n$  do  $D[j, 0] = j$ 
5      for  $i := 1$  to  $m$  do
6          for  $j := 1$  to  $n$  do
7              if  $x(i) = y(j)$ , then
8                   $D[j, i] := \min\{D[j, i - 1], D[j - 1, i], D[j - 1, i - 1]\} + 0$ 
9              else
10                  $D[j, i] := \min\{D[j, i - 1], D[j - 1, i], D[j - 1, i - 1]\} + 1$ 
11             next
12         next
13     return  $D[n, m]$ 
14 end

```

Dari algoritma di atas baris ke-3 adalah pengisian baris pertama dari tabel Jarak Levenshtein. Baris ke-4 adalah pengisian kolom pertama. Dari baris ke-5 sampai dengan baris ke-12 dilakukan pengisian kolom ke-2 sampai kolom ke- $(m+1)$ dimana setiap kolom terdapat n elemen. Pengisian setiap elemen adalah dari baris ke-6 sampai baris ke-11. Dimana setiap pengisian elemen ini terdapat operasi perbandingan. Sehingga secara keseluruhan terdapat $n \times m$ operasi perbandingan. Dengan demikian kompleksitas algoritma program dinamik untuk jarak Levenshtein adalah $O(m \cdot n)$ atau (n^2) .

3.5. Sifat-sifat Pada Pencocokkan Untai Terdekat dengan Ukuran Jarak Levenshtein

Berdasarkan pembahasan sebelumnya maka akan dilakukan pengamatan pada berbagai contoh kasus. Dari pengamatan akan diperoleh sifat-sifat yang terdapat pada pencocokkan hampiran untaian. Tentu saja pada tulisan ini pencocokkan untaian x dan y menggunakan ukuran jarak Levenshtein dengan program dinamik.

Contoh 3.2

Misal $x = \hat{\ }^{\wedge}$ dan $y = \hat{\ }^{\wedge}$

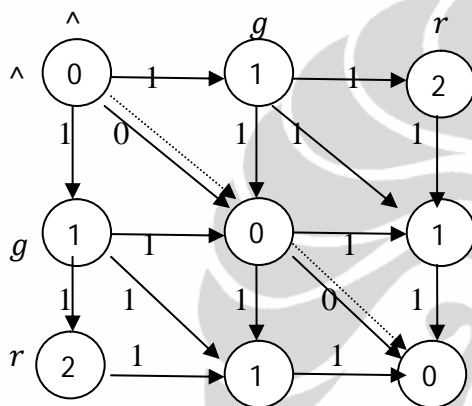
$$|x| = |y| = 2$$

Sudah dapat dipastikan tidak perlu ada pencocokkan karena kedua untai identik jadi jarak levenshtein adalah 0.

Contoh 3.3

Misal $x = gr$

$$y = gr$$



Gambar 3.21 Graf hasil proses pencocokkan antara $x = gr$ dan $y = gr$

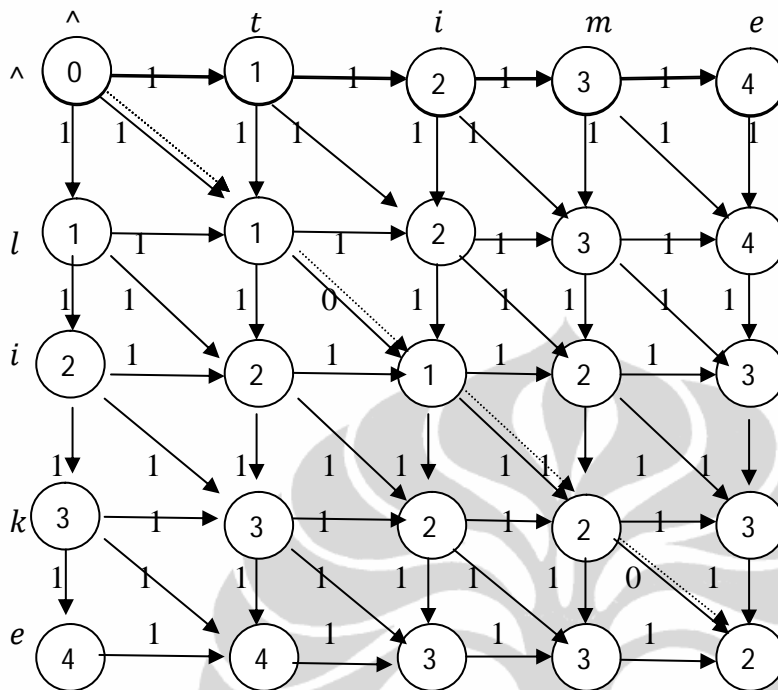
Jika $|x| = |y| = 2$ dan $x = y$, dengan kata lain untai x identik dengan untai y maka tidak perlu ada operasi untuk menyamakan x dan y , dengan demikian $D(x, y) = 0$

Contoh 3.4.

Misal $x = time$

$$y = like$$

Pada contoh 3.4. dari gambar 3.22 karakter yang sama pada posisi yang sama $x(2) = y(2)$, yaitu $i = i$ dan $i \neq l, k, e$ dan karakter pada $x(4) = y(4)$ yaitu $e = e$ dan $m \neq l, i, e$. Operasi yang dilakukan untuk menyamakan x dan y adalah mengganti t pada $x(1)$ menjadi y dan m pada $x(3)$ menjadi k . Dengan demikian ada 2 operasi penggantian jadi $D[x, y] = 4 - 2 = 2$.

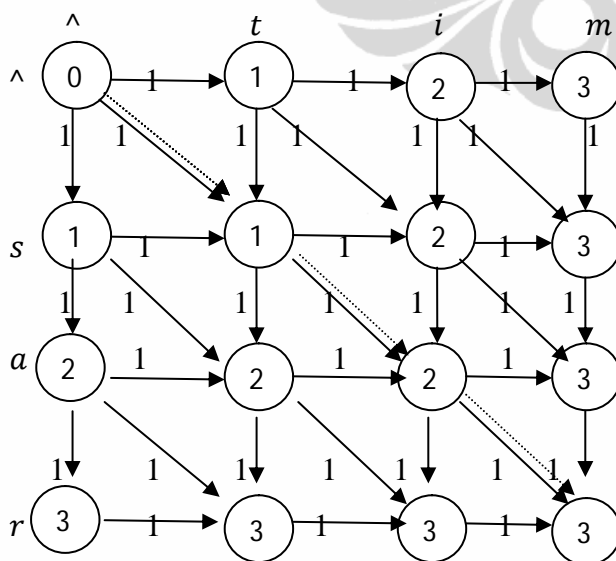


Gambar 3.22 Graf hasil proses pecocokkan antara $x = \text{time}$ dan $y = \text{like}$

Contoh 3.5

Misal $x = \text{tim}$

$y = \text{sar}$



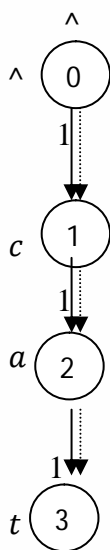
Gambar 3.23 Graf hasil proses pecocokkan antara $x = \text{tim}$ dan $y = \text{sar}$

Pada contoh 3.5. dan gambar 3.23 tidak ada karakter yang sama pada posisi yang sama dan $\forall x(i) \neq \forall y(j)$, dimana $i = 1,2,3$ dan $j = 1,2,3$. Operasi yang dilakukan untuk menyamakan x dan y adalah mengganti t pada $x(1)$ menjadi s , mengganti i pada $x(2)$ dengan a dan m pada $x(3)$ menjadi r . Dengan demikian ada 3 operasi penggantian jadi $D[x, y] = 3 - 0 = 3$.

Contoh 3.6

Misal $x = \wedge$

$y = cat$



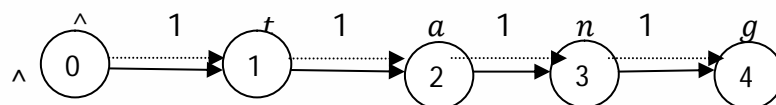
Gambar 3.24 Graf hasil proses pencocokkan antara $x = \wedge$ dan $y = cat$

Ada tiga penyisipan karakter dari untai y ke untai x , karakter $y(1)$ yaitu c disisipkan di $x(1)$, karakter $y(2)$ yaitu a disisipkan di $x(2)$ dan karakter $y(3)$ yaitu t disisipkan di $x(3)$. maka $D[x, y] = 3$.

Contoh 3.7

Misal $x = tang$

$y = \wedge$



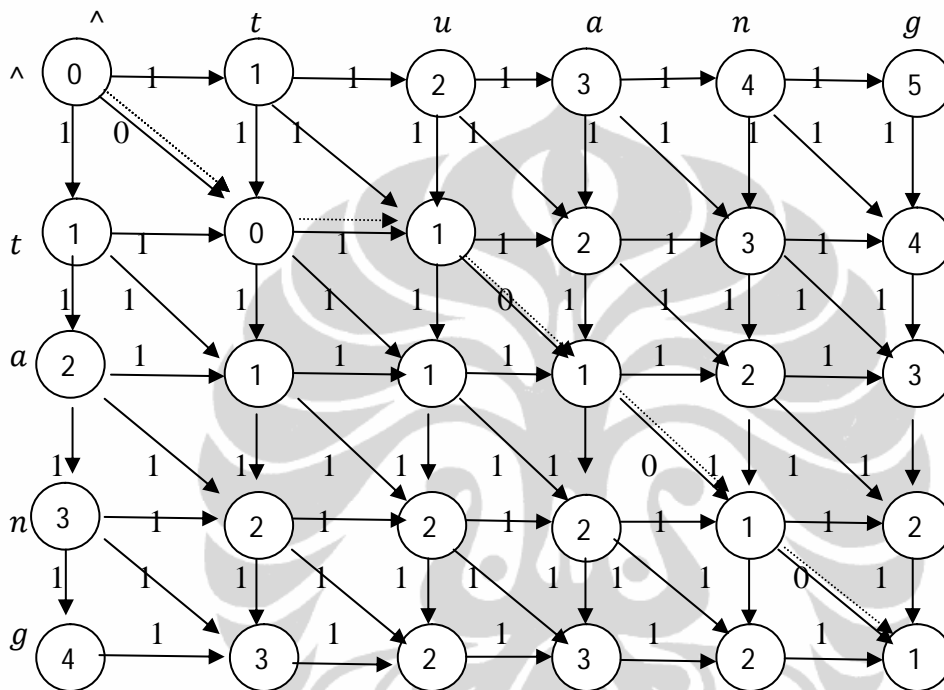
Gambar 3.25 Graf hasil proses pencocokkan antara $x = tang$ dan $y = \wedge$

Ada empat penghapusan karakter pada untai x , sehingga $x' = y = \hat{\ }.$
maka $D[x, y] = 4.$

Contoh 3.8

Misal $x = tuang$

$y = tang$



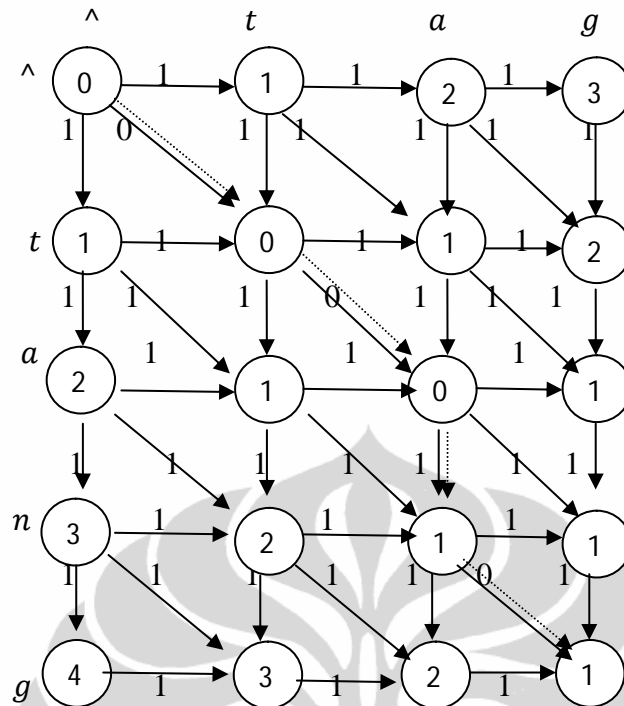
Gambar 3.26 Graf hasil proses pencocokan antara $x = tuang$ dan $y = tang$

Untai $y = tang$ merupakan sub barisan dari $x = tuang$, terdapat satu karakter yaitu u pada $x(2)$ yang bukan karakter pada y . Operasi yang dilakukan adalah menghapus u pada $x(2)$, dengan demikian $D[x, y] = 1.$

Contoh 3.9

Misal $x = tag$

$y = tang$



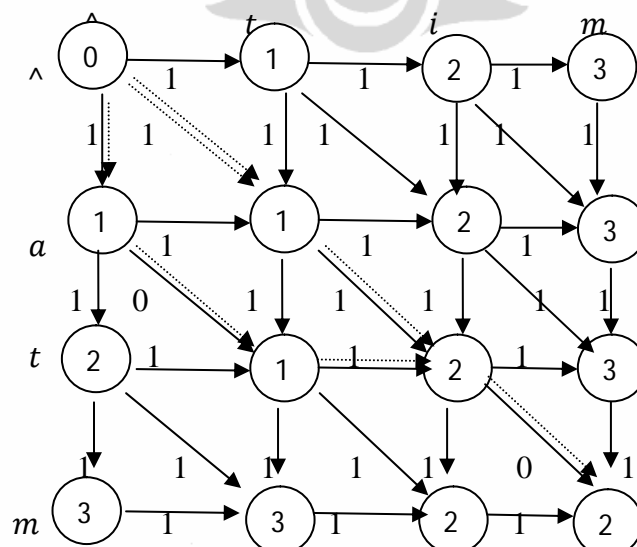
Gambar 3.27 Graf hasil proses pencocokan antara $x = tag$ dan $y = tang$

Untai $x = tag$ merupakan sub barisan dari $y = tang$, terdapat satu karakter yaitu n pada $y(3)$ yang bukan karakter pada y . Operasi yang dilakukan adalah menyisipkan n pada $x(3)$, dengan demikian $D[x, y] = 1$.

Contoh 3.10

Misal $x = tim$

$y = atm$



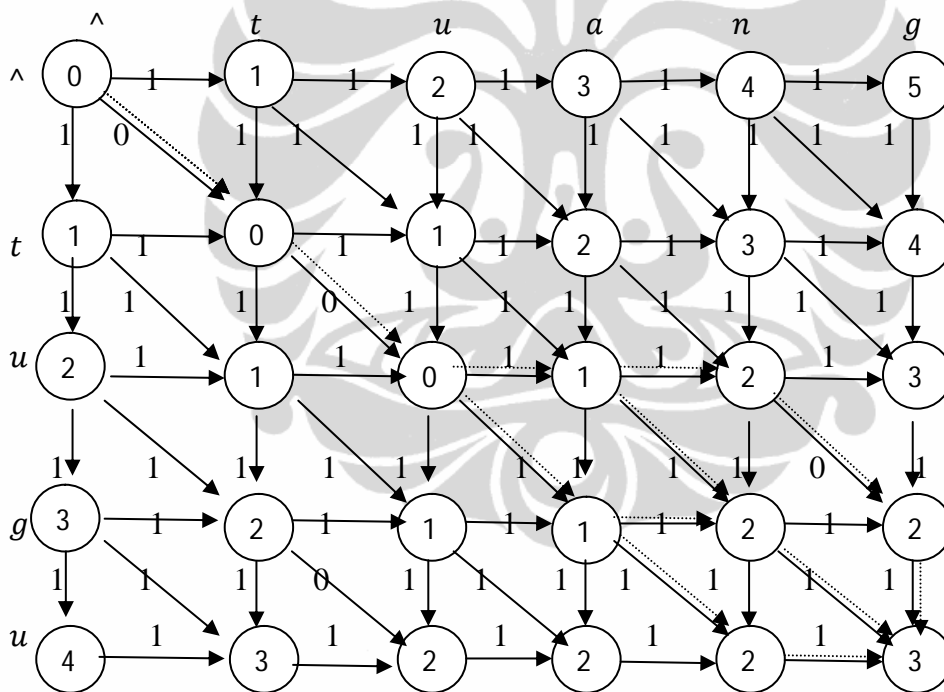
Gambar 3.28 Graf hasil proses pencocokan antara $x = tim$ dan $y = atm$

Misalkan untai $z = tm$ maka sub barisan persekutuan terpanjang antara untai $x = tim$ dan untai $y = atm$, terdapat 1 karakter pada x yang bukan karakter z yaitu i pada $x(2)$ dimana posisi i menyisip di antara t dan m pada untai x dan terdapat 1 karakter pada y yang bukan karakter z yaitu a pada $y(1)$ dimana posisinya sebagai awalan dari $y = atm$. Operasi yang dilakukan adalah mengganti t pada $x(1)$ dengan a dan i pada $x(2)$ dengan t atau alternatif kedua adalah menyisipkan a pada $x(1)$ kemudian menghapus i pada $x(3)$, dengan demikian $D[x, y] = 1 + 1 = 2$.

Contoh 3.11

Misal $x = tuang$

$y = tugu$



Gambar 3.29 Graf hasil proses pencocokan antara $x = tuang$ dan $y = tugu$

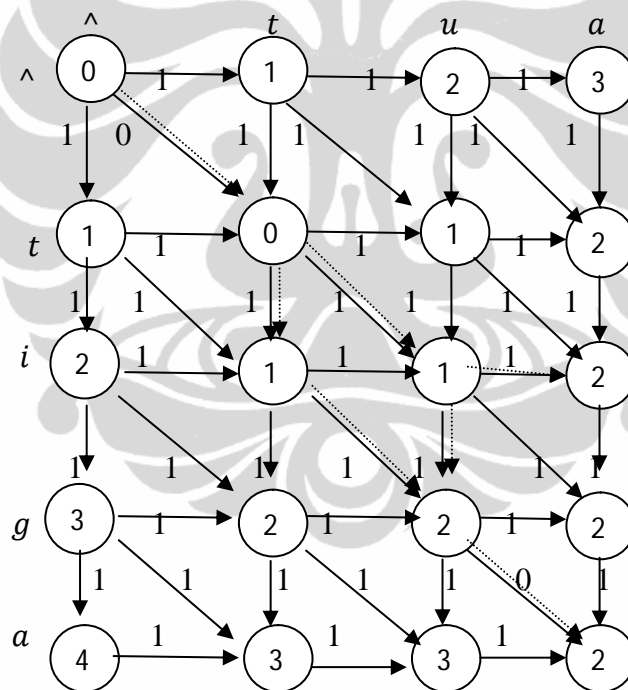
Misalkan untai $z = tug$ maka sub barisan persekutuan terpanjang antara untai $x = tuang$ dan untai $y = tugu$, terdapat 2 karakter pada x yang bukan karakter z yaitu a pada $x(3)$ dan n pada $x(4)$ dengan posisi diantara tu dan g , serta terdapat 1 karakter pada y yang bukan karakter z yaitu u pada $y(4)$ dimana

posisinya sebagai akhiran dari $z = tugu$. Ada empat kemungkinan operasi yang dilakukan untuk menyamakan x dan y . Kemungkinan pertama mengganti a dan n pada x masing-masing dengan g dan u kemudian menghapus g pada x (5). Kemungkinan kedua menghapus a dan n pada x kemudian menyisipkan u pada x (4). Kemungkinan ketiga menghapus a pada x kemudian mengganti n dan g masing-masing dengan g dan u . Dan kemungkinan keempat mengganti a dan g pada x masing-masing dengan g dan u serta menghapus n . Dan semua kemungkinan tersebut mempunyai jarak Levenshtein $D[x, y] = 2 + 1 = 3$.

Contoh 3.12

Misal $x = tua$

$y = tiga$



Gambar 3.30 Graf hasil proses pencocokan antara $x = tua$ dan $y = tiga$

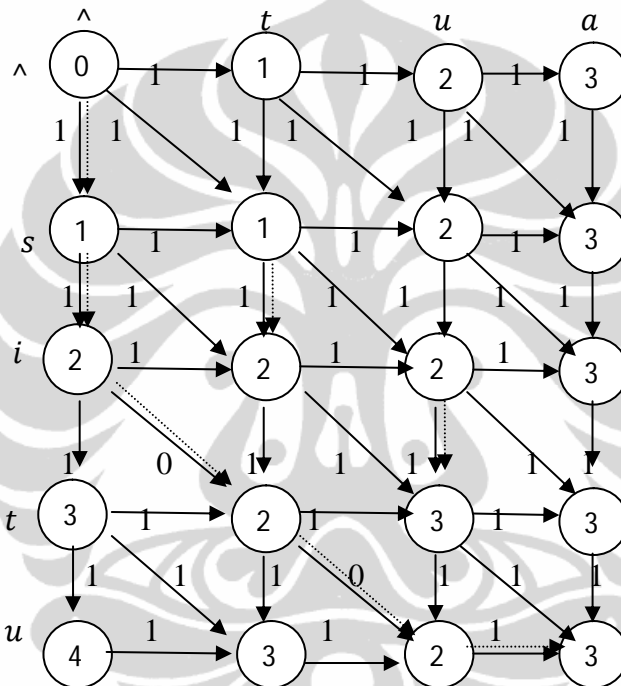
Misalkan untai $z = ta$ maka sub barisan persekutuan terpanjang antara untai $x = tua$ dan untai $y = tiga$, terdapat 1 karakter pada x yang bukan karakter z yaitu u pada $x(2)$ dengan posisi sebagai sisipan dari $x = tua$ dan terdapat 2 karakter pada y yang bukan karakter z yaitu i pada $y(2)$ dan g pada $y(3)$ dimana posisinya sebagai sisipan dari $y = tiga$. Ada dua kemungkinan operasi yang

dilakukan untuk menyamakan x dan y . Kemungkinan pertama menyisipkan i pada $x(2)$ kemudian mengganti u dengan g . Kemungkinan kedua mengganti u dengan i pada $x(2)$ kemudian menyisipkan g . Dan semua kemungkinan tersebut mempunyai jarak Levenshtein $D[x, y] = 2 + 0 = 2$.

Contoh 3.13

Misal $x = tua$

$y = tiga$



Gambar 3.31 Graf hasil proses pencocokan antara $x = tua$ dan $y = situ$

Misalkan untai $z = tu$ maka sub barisan persekutuan terpanjang antara untai $x = tua$ dan untai $y = situ$, terdapat 1 karakter pada x yang bukan karakter z yaitu a pada $x(3)$ dengan posisi sebagai akhiran dari $x = tua$ dan terdapat 2 karakter pada y yang bukan karakter z yaitu s pada $y(1)$ dan i pada $y(2)$ dimana posisinya sebagai awalan dari $y = satu$. Operasi yang dilakukan untuk menyamakan x dan y adalah menyisipkan s dan i masing-masing pada $x(2)$ dan $x(2)$ kemudian menghapus a pada $y(3)$. Dengan demikian jarak Levenshtein antara dua untai tersebut adalah $D[x, y] = 1 + 2 = 3$.

Berikut ini adalah ringkasan dari beberapa kasus yang telah dipaparkan di atas.

- a. Untai x dan y adalah dua buah untai yang sama jika dan hanya jika jarak Levenshtein antara x dan y sama dengan nol atau $D(x, y) = 0$.
- b. Jika panjang x sama dengan 0 dan panjang y sama dengan n , maka jarak Levenshtein antara x dan y sama dengan panjang y yaitu n atau $D(x, y) = n$.
- c. Jika panjang x sama dengan n dan panjang y sama dengan 0, maka jarak Levenshtein antara x dan y sama dengan panjang x yaitu n atau $D(x, y) = n$.
- d. Jika panjang x sama dengan n , panjang y sama dengan m dengan $n < m$ dan x merupakan sub barisan dari y maka jarak Levenshtein antara x dan y sama dengan panjang y dikurangi panjang x atau $D(x, y) = m - n$.
- e. Jika panjang x sama dengan n , panjang y sama dengan m dengan $n > m$ dan y merupakan sub barisan dari x maka jarak Levenshtein antara x dan y sama dengan panjang x dikurangi panjang y atau $D(x, y) = n - m$.
- f. Jika diberikan dua untai x dan y dengan $|x| = n$ dan $|y| = m$. Misalkan z adalah subbarisan irisan terpanjang antara x dan y , p_x, q_x, r_x adalah awalan, sisipan dan akhiran dari x . Sedangkan p_y, q_y, r_y adalah awalan, sisipan dan akhiran dari y . $|p_x| + |q_x| + |r_x| + |z| = n$ dan $|p_y| + |q_y| + |r_y| + |z| = m$. Maka jarak Levenshtein antara x dan y adalah jumlahan awalan terpanjang, sisipan terpanjang dan akhiran terpanjang pada untai x atau y . Atau $D(x, y) = \max[|p_x|, |p_y|] + \max[|q_x|, |q_y|] + \max[|r_x|, |r_y|]$.

BAB 4

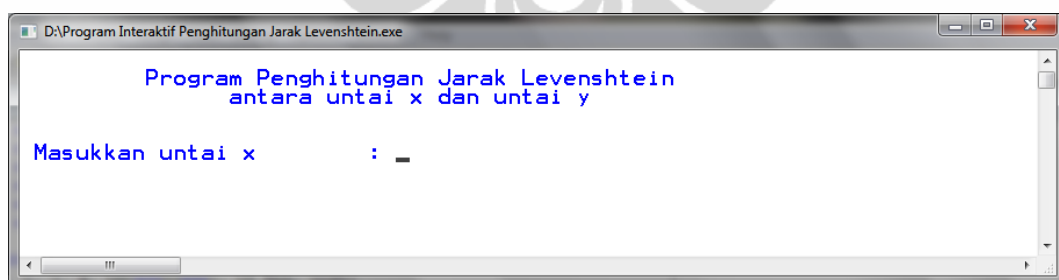
IMPLEMENTASI DAN SIMULASI

4.1. Implementasi

Dari algoritma yang didapat, maka dibuatlah suatu program sederhana untuk menghitung jarak Levenshtein dengan menggunakan pemrograman dinamik. Program ini dibuat dengan menggunakan bahasa C pada aplikasi *Borland C++*. Listing program perhitungan jarak Levenshtein tersebut dapat dilihat pada Lampiran. Dengan program ini dapat ditentukan nilai jarak Levenshtein antara dua untai sebagai inputnya. Panjang maksimum untai yang dapat dijadikan input adalah 500 karakter. Dan karakter yang digunakan adalah seluruh karakter yang terdapat pada *keyboard* komputer pada umumnya.

4.2. Simulasi Program

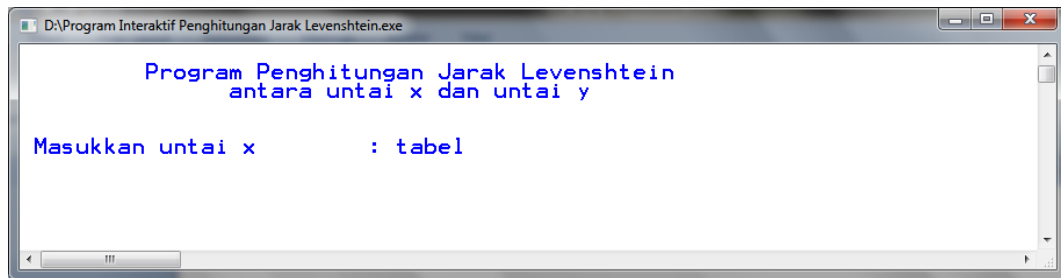
Tampilan awal dari program perhitungan jarak Levenshtein pada *Command Window* terlihat seperti pada Gambar 4.1.



Gambar 4.1. Tampilan awal pada *Command Window*

Selanjutnya dapat memulai menggunakan program ini. Langkah pertama adalah memasukkan dua buah untai. Untai pertama dengan panjang m dan untai kedua dengan panjang n yang akan di hitung jarak levenshteinya. Untai ini dapat berupa barisan huruf, angka., simbol yang ada pada *keyboard* computer.

Misalkan kita masukkan untai $x = \text{tabel}$.



Gambar 4.2. Tampilan input untai berikutnya pada *Command Window*

Kemudian masukkan untai $y = \text{able}$. Setelah *enter* maka akan muncul suatu tabel dari program dinamik pencocokkan dua untai x dan y yang telah diketikkan. Angka pada ujung diagonal (pojok kanan bawah) adalah jarak levenshtein antara x dan y . Pada gambar 4.3. memperlihatkan bahwa jarak levenshtein antara antara untai $x = \text{tabel}$ dan $y = \text{able}$ adalah 3.

Selanjutnya dengan menekan *enter* maka akan diperoleh lintasan terpendek yang menggambarkan proses pencocokkan untai x dan y . Seperti pada pencocokkan antara untai $x = \text{tabel}$ dan $y = \text{able}$, lintasan terpendeknya dapat dilihat pada gambar 4.4. yang opsinya adalah menghapus t pada $x(1)$ dan e pada $x(4)$ serta menyisipkan e pada posisi $x(5)$. Kadangkala lintasan terpendek dari program dinamik pencocokkan dua untai ini dapat lebih dari satu. Artinya ada alternatif lain. Perhatikan gambar 4.4., gambar 4.5., dan gambar 4.6.

Program Penghitungan Jarak Levenshtein
antara untai x dan untai y

Masukkan untai x : tabel
Masukkan untai y : able

Tabel Hasil Penghitungan Jarak Levenshtein.

	t	a	b	e	l	
0	0	1	2	3	4	5
a	1	1	1	2	3	4
b	2	2	2	1	2	3
l	3	3	3	2	2	2
e	4	4	4	3	2	3

Gambar 4.3. Tampilan tabel hasil perhitungan jarak Levenshtein pada *Command Window*

Program Penghitungan Jarak Levenshtein
antara untai x dan untai y

Masukkan untai x : tabel
Masukkan untai y : able

Tabel Hasil Penghitungan Jarak Levenshtein.

	t	a	b	e	l	
0	0	1	2	3	4	5
a	1	1	1	2	3	4
b	2	2	2	1	2	3
l	3	3	3	2	2	2
e	4	4	4	3	2	3

Gambar 4.4. Tampilan kemungkinan pertama operasi-operasi yang dilakukan pada untai *x*

Program Penghitungan Jarak Levenshtein
antara untai x dan untai y

Masukkan untai x : tabel
Masukkan untai y : able

Tabel Hasil Penghitungan Jarak Levenshtein.

		t	a	b	e	l	
0	-1	0	1	2	3	4	5
a	1	1	1	2	3	4	
b	2	2	2	1	2	3	
l	3	3	3	2	2	2	
e	4	4	4	3	2	3	

Gambar 4.5. Tampilan kemungkinan kedua operasi-operasi yang dilakukan pada untai x

Program Penghitungan Jarak Levenshtein
antara untai x dan untai y

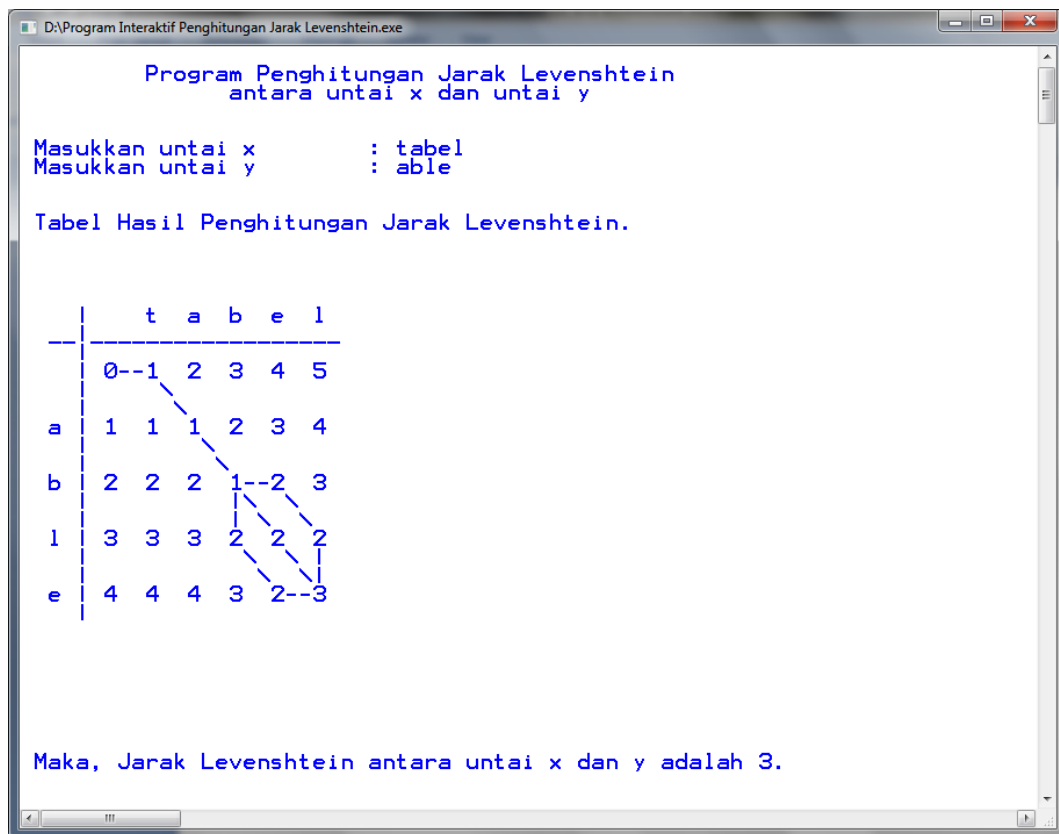
Masukkan untai x : tabel
Masukkan untai y : able

Tabel Hasil Penghitungan Jarak Levenshtein.

		t	a	b	e	l	
0	-1	0	1	2	3	4	5
a	1	1	1	2	3	4	
b	2	2	2	1	2	3	
l	3	3	3	2	2	2	
e	4	4	4	3	2	3	

Gambar 4.6. Tampilan kemungkinan ketiga operasi-operasi yang dilakukan pada untai x

Pada gambar 4.5. memperlihatkan adanya lintasan terpendek ke-2 yang menunjukkan adanya operasi pencocokkan alternatif yaitu menghapus t pada $x(1)$ mengganti e pada $x(4)$ dengan l dan mengganti l pada $x(5)$ dengan e . Pada gambar 4.6. lintasan terpendek ke-3 menunjukkan adanya operasi pencocokkan alternatif yaitu menghapus t pada $x(1)$ menyisipkan l pada $x(4)$ dan menghapus l pada $x(5)$. Dan selanjutnya akan dinyatakan jarak Levenshteinnya.



Gambar 4.7. Tampilan Hasil Akhir Pada *Command Window*

Dengan menggunakan program ini maka dapat dilakukan pengamatan lebih lanjut untuk berbagai kasus dan akan diperoleh sifat-sifat pencocokkan antara dua untai terkait dengan nilai jarak Levenshtein. Dan akhirnya dapat diambil kesimpulan terhadap sifat-sifat tersebut.

BAB 5 PENUTUP

Pada bab ini, akan diberikan beberapa kesimpulan yang dihasilkan dari pembahasan pada bab-bab sebelumnya sehingga menjawab semua tujuan dari penulisan tesis ini.

- a. Untuk menentukan tingkat kemiripan antara dua buah untai x dan y dengan panjang yang tidak harus sama, misalkan $|x| = m$, $|y| = n$, sepanjang alfabet V berukuran r sedemikian sehingga x dan y anggota dari V^* , maka diperlukan suatu ukuran yaitu jarak Levenshtein. Jarak Levenshtein adalah banyaknya operasi penghapusan, penyisipan atau penggantian yang minimal untuk menyamakan untai x dan y . Suatu metode yang dapat digunakan untuk mencari solusi untuk mencari jarak Levenshtein dalam pencocokkan hampiran untai adalah program dinamik.
- b. Jika diberikan dua untai x dan y dengan $|x| = n$ dan $|y| = m$. Misalkan z adalah subbarisan irisan terpanjang antara x dan y , p_x, q_x, r_x adalah awalan, sisipan dan akhiran dari x . Sedangkan, p_y, q_y, r_y adalah awalan, sisipan dan akhiran dari y . $|p_x| + |q_x| + |r_x| + |z| = n$ dan $|p_y| + |q_y| + |r_y| + |z| = m$. Maka jarak Levenshtein antara x dan y adalah jumlahan awalan terpanjang, sisipan terpanjang dan akhiran terpanjang pada untai x atau y . Atau $D(x, y) = \max[|p_x|, |p_y|] + \max[|q_x|, |q_y|] + \max[|r_x|, |r_y|]$.
- c. Dengan menggunakan algoritma program dinamik yang telah dibahas sebelumnya dengan kompleksitas waktu $O(n^2)$, didapatkan suatu program sederhana untuk pencocokkan hampiran untai menggunakan ukuran jarak Levenshtein dalam bahasa C pada aplikasi *Borland C++*.

DAFTAR PUSTAKA

- Bartle, Robert G., & Sherbert, Donald R. (2000). *Introduction to real analysis* (3rd ed.). New York: John Wiley & Sons.
- Lewis, Harry R., & Papadimitriou, Cristos H. (1998). *Elements of the theory of computation* (2nd ed.). New Jersey: Prentice Hall.
- Mulyono, Sri(2007). *Riset Operasi*(Edisi revisi).Jakarta: Lembaga Penerbit Fakultas Ekonomi Universitas Indonesia.
- Navarro, Gonzalo(2001), *A Gaided Tour to Approximate String Matching*. Chile: Dept. of Computer Science, University of Chile, Blanco Encalada 2120, Santiago,
- Pinzón, Yoan.(2006). *Algorithms for Approximate String Matching*. Colombia: Universidad Nacional de Colombia Departamento de Ingenieria de Sistemas e Industrial.
- Tretyakov, Konstantin(2003). *An Overview of Two Fast Bit-Vector Approximate String Matching Algorithm*. Estonia: Bioinformatics Group, Institute of Computer Science, Software Technology and Competence Centre.

LAMPIRAN

Listing Program Penghitungan Jarak Levenshtein

```
#include <dos.h> //sleep();
#include <stdio.h> //gets();
#include <conio.h> //clrscr(); gotoxy();
#include <stdlib.h>
#include <string.h> //strlen();
#include <ctype.h> //tolower();

char x[500], y[500], x2[500], y2[500], opsi[500];
int m, n, i, j, k, LD[500][500], LD_Min, cost[500][500];

void brute_force_LD(void);
void tabel(void);
void runut_balik(void);

main()
{
    clrscr();
    opsi[0] = 's';
    while (opsi[0] == 's' || opsi[0] == 'x' || opsi[0] == 'y' )
    {
        clrscr();
        printf(" \n");
        printf(" \t Program Penghitungan Jarak Levenshtein \n");
        printf(" \t \t antara untai x dan untai y \n");
        printf(" \n\n");

        if (opsi[0] == 's')
        {
            printf(" Masukkan untai x \t : ");
            gets(x);
            m = strlen(x);
            printf(" Masukkan untai y \t : ");
            gets(y);
            n = strlen(y);
        }

        if (opsi[0] == 'x')
        {
            printf(" Masukkan untai x \t : \n");
            printf(" Masukkan untai y \t : %s ", y);
            gotoxy(28,6); gets(x);
            m = strlen(x);
            printf(" \n");
        }

        if (opsi[0] == 'y')
        {
            printf(" Masukkan untai x \t : %s ", x);
            printf(" \n");
            printf(" Masukkan untai y \t : ");
            gets(y);
            n = strlen(y);
        }
    }
}
```



```

    }

    strcpy(x2,x);
    strcpy(y2,y);
    strlwr(x2);
    strlwr(y2);

    brute_force_LD();
    printf(" \n\n");
    printf(" Tabel Hasil Penghitungan Jarak Levenshtein.\n\n");
    tabel();
    getchar();
    runut_balik();
    gotoxy(1,27+3*(n-1));
    printf(" Maka, Jarak Levenshtein antara untai x dan y adalah
    %d. \n", LD_Min);
    getchar();

    k = 0;
    while (x2[k] != NULL)
    {
        x2[k] = NULL;
        k++;
    }

    k = 0;
    while (y2[k] != NULL)
    {
        y2[k] = NULL;
        k++;
    }

    printf(" \n\n");
    printf(" Anda Ingin Coba Lagi ? \n\n");
    printf(" \t Jika ingin merubah x, \t\t pilih 'x' \n\n");
    printf(" \t Jika ingin merubah y, \t\t pilih 'y' \n\n");
    printf(" \t Jika ingin merubah x dan y, \t pilih 's' \n\n");
    printf(" \t Jika ingin keluar, \t\t pilih 'e' \n\n");
    printf(" Pilihan Anda : ");
    gets(opsi);
    tolower(opsi[0]);

    while (strlen(opsi) != 1 || (opsi[0] != 'x' && opsi[0] != 'y'
    && opsi[0] != 's' && opsi[0] != 'e'))
    {
        printf(" Input Salah. Ulangi : ");
        gets(opsi);
        tolower(opsi[0]);
    }
}

void brute_force_LD()
{
    LD_Min = 0;

    if (n == 0)
    {

```

```

    LD_Min = 0;
}

else
{
    for (i = 0; i <= m; i++)
    {
        LD[0][i] = i;
    }

    for (j = 0; j <= n; j++)
    {
        LD[j][0] = j;
    }

    for (i = 1; i <= m; i++)
    {
        for (j = 1; j <= n; j++)
        {
            LD_Min = min(LD[j-1][i], LD[j][i-1]) + 1;

            if (x2[i-1] == y2[j-1])
            {
                cost[j][i] = 0;
                LD[j][i] = min(LD_Min, LD[j-1][i-1] + cost[j][i]);
            }

            else
            {
                cost[j][i] = 1;
                LD[j][i] = min(LD_Min, LD[j-1][i-1] + cost[j][i]);
            }
        }
    }
}

LD_Min = LD[n][m];
}

```

```

void tabel()
{
    for (i = 0; i <= 3*(m+1)+2; i++)
    {
        gotoxy(3+i,16); printf("_\n");
    }

    for (j = 0; j <= 3*(n+1)+1; j++)
    {
        gotoxy(5,15+j); printf("| \n");
    }

    for (i = 0; i <= m+1; i++)
    {
        for (j = 0; j <= n+1; j++)
        {
            if (i == 0)
            {
                if (j <= 1)

```

```

        {
            gotoxy(3,15+3*j); printf(" ");
        }

        else
        {
            gotoxy(2,15+3*j); printf(" %c", y[j-2]);
        }
    }

    if (i == 1)
    {
        if (j == 0)
        {
            gotoxy(6,15); printf(" ");
        }

        else
        {
            gotoxy(6,15+3*j); printf(" %d", j-1);
        }
    }

    if (i > 1)
    {
        if (j == 0)
        {
            gotoxy(3+3*i,15); printf(" %c", x[i-2]);
        }

        if (j == 1)
        {
            gotoxy(3+3*i,18); printf(" %d", i-1);
        }

        if (j > 1)
        {
            gotoxy(3+3*i,15+3*j); printf(" %d", LD[j-1][i-1]);
        }
    }
}

printf("\n");
}
}

```

```

void runut_balik()
{
    int k[100][100], ibefore[100][100], jbefore[100][100];

    for (i = 1; i <= m; i++)
    {
        for (j = 1; j <= n; j++)
        {
            k[j][i] = 1;
        }
    }
}

```

```

tabel();
i = m;
j = n;
ibefore[n][m] = m + 1;
jbefore[n][m] = n + 1;

while (i <= m && j <= n)
{
    int icur, jcur;

    while (i > 0 && j > 0 && i <= m && j <= n)
    {
        if (k[j][i] <= 1 && LD[j][i] - 1 == LD[j-1][i])
        {
            k[j][i] = 2;
            gotoxy(10+3*(i-1),20+3*(j-1)); printf("|");
            gotoxy(10+3*(i-1),19+3*(j-1)); printf("|");
            j = j - 1;
            ibefore[j][i] = i;
            jbefore[j][i] = j+1;
        }

        else if (k[j][i] <= 2 && LD[j][i] - cost[j][i] == LD[j-1][i-1])
        {
            k[j][i] = 3;
            gotoxy(9+3*(i-1),20+3*(j-1)); printf("\\");
            gotoxy(8+3*(i-1),19+3*(j-1)); printf("\\");
            i = i - 1;
            j = j - 1;
            ibefore[j][i] = i+1;
            jbefore[j][i] = j+1;
        }

        else if (k[j][i] <= 3 && LD[j][i] - 1 == LD[j][i-1])
        {
            k[j][i] = 4;
            gotoxy(9+3*(i-1),21+3*(j-1)); printf("-");
            gotoxy(8+3*(i-1),21+3*(j-1)); printf("-");
            i = i - 1;
            ibefore[j][i] = i+1;
            jbefore[j][i] = j;
        }

        else
        {
            icur = ibefore[j][i];
            jcur = jbefore[j][i];
            i = icur;
            j = jcur;
        }

        getchar();
    }

    if (j == 0)
    {
        while (i > 0)
        {

```

```

        k[j][i] = 4;
        gotoxy(9+3*(i-1),18); printf("-");
        gotoxy(8+3*(i-1),18); printf("-");
        i = i - 1;
        ibefore[j][i] = i+1;
        jbefore[j][i] = 0;
    }

    k[j][i] = 4;
}

else
{
    if (i == 0)
    {
        while (j > 0)
        {
            k[j][i] = 4;
            gotoxy(7,20+3*(j-1)); printf("|");
            gotoxy(7,19+3*(j-1)); printf("|");
            j = j - 1;
            ibefore[j][i] = 0;
            jbefore[j][i] = j+1;
        }

        k[j][i] = 4;
    }

    else
    {
        break;
    }
}

while (k[j][i] > 3 && i <= m && j <= n)
{
    icur = ibefore[j][i];
    jcur = jbefore[j][i];
    i = icur;
    j = jcur;
}

getchar();
}
}

```