BAB II LANDASAN TEORI

Bab ini berisi penjelasan tentang definisi, teori dan konsep yang digunakan penulis untuk mamahami cara yang benar untuk mendapatkan pola sekuensial (*sequential patterns*) dengan menggunakan algoritma *AprioriAll*.

2.1 *DATA MINING*

Data mining adalah analisis terhadap sekelompok data (biasanya berukuran besar) untuk mendapatkan hubungan yang tak terduga dan merangkum hasilnya dalam bentuk yang lebih mudah dipahami dan lebih mudah digunakan **[HAN01]**. Hubungan dan rangkuman yang dihasilkan biasanya berupa model (*models*) atau pola (*pattern*).

Data mining umumnya dikelompokkan menjadi berbagai tugas (*task*). Namun sejauh ini, belum ada kesepakatan yang universal dari pada ahli *data mining* dalam mengelompokan tugas (*task*) ini. Oleh karena itu, penulis memilih pengelompokkan tugas-tugas *data mining* yang memiliki pandangan yang lebih luas dari pengelompokkan lainnya sebagai berikut **[HAN01]**.

2.1.1 *Exploratory Data Analysis* (EDA)

Sesuai dengan namanya, tugas (*task*) *data mining* ini adalah mengekplorasi data tanpa kejelasan target informasi yang dicarinya. Umumnya teknik yang digunakan EDA ini bersifat interaktif dan visual. Hasilnya biasanya berupa grafik atau gambar.

2.1.2 *Descriptive Modeling*

Tujuan dari tugas (*task*) ini adalah mendeskripsikan data. Contoh deskripsi yang mungkin diperoleh yaitu:

- Model kepadatan data (*density estimation*).
- Model pengelompokkan data (*cluster analysis and segmentation*).
- Model hubungan antar variabel (*dependency modeling*).

2.1.3 *Predictive Modeling*

Tujuan dari tugas (*task*) ini adalah membangun model yang mampu melakukan prediksi. *Predictive Modeling* terbagi menjadi dua metode yaitu:

- *Classification*, dimana hasil prediksinya bersifat terkategori.
- *Regression*, dimana hasil prediksinya berisfat kuantitatif.

2.1.4 *Discovering Patterns and Rules*

Jika ketiga tugas (*task*) sebelumnya umumnya bertujuan membangun model, maka pada tugas (*task*) ini bertujuan untuk mendeteksi pola. Metode yang digunakan pada tugas (*task*) ini adalah *association rules*.

2.1.5 *Retrieval by Content*

Pada tugas (*task*) ini, dengan diketahuinya sebuah pola, maka dilakukan pencarian pola yang sama pada data. Tugas (*task*) ini biasanya menggunakan teks atau gambar sebagai data.

Penulis tidak akan membahas lebih dalam masing-masing pengelompokkan tugas (*task*) tersebut. Pengelompokkan tersebut hanya digunakan penulis untuk menujukkan posisi *association rules* dalam *data mining*.

2.2 *ASSOCIATION RULES*

Association rules merupakan salah satu metode dalam *data mining* yang pertama kali diperkenalkan oleh Agrawal dkk pada tahun 1993 melalui konsep *market-basket analysis* **[AGR94]**. *Market-basket* adalah kumpulan barang (*item*) yang dibeli oleh pembeli (*customer*) pada sebuah transaksi. Pedagang (*retailer*) akan menyimpan semua catatan transaksi yang telah terjadi. Pada suatu waktu, pedagang (*retailer*) akan melakukan analisis pada semua kumpulan transaksi tersebut untuk mencari sekelompok barang (*itemset*) yang muncul bersama-sama pada banyak transaksi. Hasil analisis akan digunakan pedagang (*retailer*) untuk meningkatkan penjualan dengan cara menata ulang sekelompok barang (*itemset*) tersebut pada lokasi yang berdekatan atau menyusun ulang katalog belanja. Dalam model matematika *association rules* dijelaskan sebagai berikut **[DUN03]**.

Jika diberikan sekumpulan *item* $I = \{I_1, I_2, \dots, I_m\}$, disebut juga *itemset*, dan diberikan *database* transaksi $D = \{t_1, t_2, \dots, t_n\}$ dimana $t_i = \{I_{i1}, I_{i2}, \dots, I_{ik}\}$ dan $I_{ij} \in I$, dimana $1 \leq j \leq k$ **[ANN07]**.

Definisi 1. *Association rules* dinyatakan dalam bentuk $R: X \rightarrow Y$, dimana $X, Y \subset I$ dan $X \cap Y = \emptyset$.

Definisi 2. *Support* dari sebuah *association rules* $R: X \rightarrow Y$ adalah persentase transaksi pada *database* yang mengandung $X \cup Y$.

$$Support(X \rightarrow Y) = P(X \cup Y) = \frac{|X \cup Y|}{|T|}$$

Definisi 3. *Confidence* dari sebuah *association rules* $R: X \rightarrow Y$ adalah jumlah transaksi yang mengandung $X \cup Y$ dibagi dengan jumlah transaksi yang mengandung X.

$$Confidence(X \rightarrow Y) = P(Y|X) = \frac{Support(X \cup Y)}{Support(X)}$$

Definisi 4. *Association rules problem* adalah untuk mencari semua *association rules* $R: X \rightarrow Y$ dengan *minimum support* dan *confidence* tertentu. Dimana *minimum support* adalah *support* yang ditentukan sebagai *input*.

Definisi 5. *Large (frequent) itemset* adalah sebuah itemset yang memiliki *support* lebih besar atau sama dengan *minimum support*.

Definisi 6. *i-itemset* adalah sebuah *itemset* yang memiliki jumlah *item* sebanyak *i item*.

Sebagai contoh untuk memahami definisi-definisi tersebut, diberikan sebuah *database* transaksi berikut ini.

TID	Itemset
001	A C D
002	B C E
003	A B C E
004	B E

Tabel 2.1 – *Transaction Database*

Dari contoh tabel di atas, TID (*transaction ID*) adalah nomor identitas transaksi yang terjadi dan *itemset* adalah kumpulan barang yang dibeli. Dari tabel tersebut juga diketahui terdapat empat transaksi dan lima jenis *item* yang diperdagangkan.

Misalkan ingin dicari nilai *support* untuk masing-masing *item* yang berarti mencari *itemset* yang panjangnya satu *item*, disebut juga *1-itemset*. *Support* untuk *1-itemset* diperoleh sebagai berikut.

Item	Jumlah	Support
A	2	2/4 = 50%
B	3	3/4 =75%
C	3	3/4 = 75%
D	1	1/4 = 25%
E	3	3/4 = 75%

Tabel 2.2 – *1-itemset*

Permaslahan *association rules* adalah mencari semua *itemset* (*1-itemset*, *2-itemset*, *3-itemset*, dst.) yang memenuhi nilai *support* tertentu, disebut juga *frequent itemset*. Sampai saat ini, untuk menyelesaikan permasalahan *association rules* telah dikenal tiga algoritma yaitu:

- Algoritma *Apriori* **[AGR94]**
- Algoritma *FP-Growth* **[HAN00]**
- Algoritma *CT-Pro* **[SUC05]**

Penulis tidak akan membahas lebih jauh dari masing-masing algoritma tersebut kecuali algoritma *Apriori* sebagai algoritma yang digunakan pada Tugas Akhir ini.

2.3 ALGORITMA *APRIORI*

Algoritma *Apriori* pertama kali diperkenalkan oleh Agrawal dkk pada tahun 1994 sebagai salah satu algoritma yang efisien untuk menyelesaikan *association rules problem* **[AGR94]**. Berikut ini adalah *pseudocode* Algoritma *Apriori*.

```
L1 = {large 1-itemsets};
for(k = 2; Lk-1 ≠ Ø; k++)
{
     Ck = apriori-gen(Lk-1);
     forall transactions t ∈ D
     {
          Ct = subset(Ck, t);
          forall candidates c ∈ Ct
          {
               c.count++;
          }

      }
      Lk = {c ∈ Ck | c.count ≥ minsup};
}
Answer = ∪k Lk;
```

Gambar 2.1 – Algoritma *Apriori* [AGR94]

Berikut ini adalah *pseudocode* **apriori-gen**.

```
// join step
INSERT     INTO Ck
SELECT     p.item1, p.item2, …, p.itemk-1, q.itemk-1
FROM       Lk-1 p, Lk-1 q
WHERE      p.item1 = q.item1, …, p.itemk-2 = q.itemk-2,
           p.itemk-1 < q.itemk-1;

// prune step
forall itemsets c ∈ Ck
{
     forall( (k-1)-subsets s of c)
     {
          if(s ∉ Lk-1)
          {
               delete c from Ck;
          }
     }
}
```

Gambar 2.2 – *Function Apriori-Gen* [AGR94]

Dengan menggunakan algoritma *Apriori*, dapat dicari semua *large itemset* dari contoh tabel 2.1 misalkan dengan *minimum support* = 50% sebagai berikut **[KAN03]**.

Dari tabel 2.2 telah didapat *1-itemset*. Untuk mendapatkan *large 1-itemset*, maka *item* D yang memiliki nilai *support* yang lebih kecil dari nilai *minimum support* harus dihilangkan (*prune step*). Tabel 2.3 berikut berisi *large 1-itemset*.

Item	Jumlah	Support
A	2	50%
B	3	75%
C	3	75%
E	3	75%

Tabel 2.3 – *large 1-itemset*

Dari *large 1-itemset*, kemudian bisa terbentuk enam *candidate 2-itemset* pada tabel 2.4 berikut ini.

Item	Jumlah	Support (%)
{A, B}	1	25
{A, C}	2	50
{A, E}	1	25
{B, C}	2	50
{B, E}	3	75
{C, E}	2	50

Tabel 2.4 – *2-itemset*

Kemudian dengan menyingkirkan *candidate* yang tidak *frequent* diperoleh *large 2-itemset* pada tabel 2.5 berikut ini.

Item	Jumlah	Support (%)
{A, C}	2	50
{B, C}	2	50
{B, E}	3	75
{C, E}	2	50

Tabel 2.5 – *large 2-itemset*

Dari *large 2-itemset* terbentuk tiga *candidate 3-itemset* pada tabel 2.6 berikut ini.

Item	Jumlah	Support (%)
{A, B, C}	1	25
{A, C, E}	1	25
{B, C, E}	2	50

Tabel 2.6 – *3-itemset*

Dengan menyingkirkan *candidate* yang tidak *frequent* maka diperoleh *large 3-itemset* pada tabel 2.7 berikut ini.

Item	Jumlah	Support (%)
{B, C, E}	2	50

Tabel 2.7 – *large 3-itemset*

Karena tersisa hanya satu *candidate large 3-itemset*, maka tidak bisa terbentuk *candidate* untuk *4-itemset*. Prose algoritma *Apriori* berhenti sampai di sini. Jawaban akhirnya terdapat pada tabel 2.8 berikut ini.

Item	Jumlah	Support (%)
A	2	50
B	3	75
C	3	75
E	3	75
{A, C}	2	50
{B, C}	2	50
{B, E}	3	75
{C, E}	2	50
{B, C, E}	2	50

Tabel 2.8 – *All large itemsets*

Algoritma *Apriori* memiliki sifat dasar yang tekenal (sifat *Apriori*) yaitu setiap *subset* dari *frequent itemset* (*large itemset*) pasti juga merupakan *frequent itemset* (*large itemset*). Sifat ini terbukti karena setiap *candidate k-itemset* selalu disusun dari *frequent* (*k-1*)-*itemset*.

2.4 ALGORITMA *APRIORIALL*

Sampai saat ini telah dikenal beberapa algoritma yang dikembangkan khusus untuk mendapatkan pola sekuensial. Masing-masing algoritma menggunakan pendekatan yang berbeda-beda. Berikut ini beberapa algoritma tersebut **[ZHA04]** yaitu:

- Algoritma *AprioriAll, AprioriSome,* dan *DynamicSome* dikembangkan oleh Agrawal dan Srikant (1995) dengan pendekatan sifat *Apriori*. Algoritma *AprioriSome* dan *DynamicSome* merupakan variasi dari algoritma *AprioriAll*.
- Algoritma GSP (*Generalize Sequential Pattern*), dikembangkan oleh Srikant dan Agrawal (1996) dengan juga pendekatan sifat *Apriori*.
- Algoritma SPIRIT (*Sequential Pattern mIning with Regular expressIon consTraints*) dikembangkan oleh Galofalaksis (1999) dengan pendekatan *regular expression*.
- Algoritma *FreeSpan* dikembangkan oleh Han (2000) dengan menggunakan algoritma *FP-Growth*.
- Algoritma *PrefixSpan* dikembangkan oleh Pei (2001) dengan menggunakan algoritma *FP-Growth*.
- Algoritma SPADE dikembangkan oleh Zaki (2001) dengan berbasis struktur *lattice*.
- Algoritma MEMISP (*MEmory Indexing for Sequential Pattern Mining*) dikembangkan oleh Lin dan Lee (2002) dengan pendekatan *memory indexing*.

Selain algoritma, juga dikenal beberapa pengembangan teknik lebih lanjut dalam mendapatkan pola sekuensial yang lebih kompleks seperti *multi-dimensional sequential pattern mining*, *incremental mining of sequential patterns*, dan *periodic pattern analysis* **[ZHA04]**.

Alasan utama algoritma *AprioriAll* dipilih oleh penulis sebagai algoritma yang digunakan untuk mencari pola sekuensial yaitu:

- Pola sekuensial (akan dijelaskan pada bagian berikutnya) bisa dipandang sebagai *itemset* yang memiliki sifat keterurutan, sehingga setiap *itemset* yang bersifat *frequent*, maka semua anggota *itemset* tersebut juga pasti bersifat *frequent*. Oleh karena itu, pola sekuensial memiliki sifat *Apriori* **[DUN03]**.
- Algoritma *AprioriAll* memang dibuat khusus untuk mencari pola sekuensial dengan prinsip kerja yang mirip dengan algoritma *Apriori*. Sehingga terdapat banyak kesamaan dalam implementasi algoritma *Apriori* dan algoritma *AprioriAll,* seperti sama-sama menggunakan teknik *candidate generation* **[AGR95]**.
- Algoritma *Apriori* berpotensi terus dikembangkan sehingga terdapat versi yang memiliki *performance* lebih baik daripada algoritma sejenisnya **[FAH06]**, contohnya algoritma *Apriori* yang dikembangkan oleh Christian Bogelt **[CHR09]**. Oleh karena banyak kesamaan antara algoritma Apriori dan algoritma *AprioriAll*, sehingga *performance* yang lebih baik dari implementasi versi pengembangan selanjutnya bisa dimiliki oleh algoritma *AprioriAll* di masa mendatang.

Berikut ini adalah *pseudocode* algoritma *AprioriAll*.

```
L1 = {large 1-sequences};
for(k = 2; Lk-1 ≠ ∅; k++)
{
     Ck = AprioriAll-gen(Lk-1);
     foreach customer-sequence c in database
     {
          Increment the count of all candidate in Ck
          that are contained in c.
      }
      Lk = Candidates in Ck with minimum support;
}
Answer = Maximal Sequences in ∪k Lk;
```

Gambar 2.3 – Algoritma *AprioriAll* [AGR95]

Berikut ini adalah *pseudocode* **AprioriAll-gen**.

```
// join step
INSERT     INTO Ck
SELECT     p.litemset1, …, p.litemsetk-1, q.litemsetk-1
FROM       Lk-1 p, Lk-1 q
WHERE      p.litemset1 = q.litemset1, …, p.litemsetk-2 = q.litemsetk-2;

// prune step
Delete all sequence c ∈ Ck such that some (k-1)-subsequence of c is not
in Lk-1.
```

Gambar 2.4 – *Function AprioriAll-Gen* [AGR95]

Kegunaan algoritma *AprioriAll* akan dijelaskan pada bagian selanjutnya. Namun secara umum prinsip kerjanya mirip dengan algoritma *Apriori*, perbedaan utamanya adalah algoritma *AprioriAll* mengolah *sequences* yang memiliki sifat keterurutan bukan *itemsets* biasa.

2.5 *MINING SEQUENTIAL PATTERN (MSP)*

Pencarian pola sekuensial, yang disebut *mining sequential pattern (MSP)* pertama kali diperkenalkan oleh Agrawal dan Srikant pada tahun 1995 melalui contoh penggalian pola sekuensial pada *transaction database* **[AGR95]**. Diberikan *transaction database* yang terdiri dari atribut nomor identitas pelanggan (*customer ID)*, waktu transaksi (*transaction time*) dan barang (*item*) yang dibeli pada masing-masing transaksi. Diasumsikan setiap pembeli memiliki identitas yang diperoleh dari kartu kredit atau kartu pelanggan sehingga nomor identitas bisa diperoleh. Tujuan akhirnya adalah mendapatkan semua kelompok barang yang muncul berurutan dengan presentase tertentu pada *transaction database* tersebut.

Misalkan *transaction database* tersebut berasal toko rental DVD, dimana semua pelanggannya memiliki identitas yang dikenal, maka contoh pola sekuensial yang mungkin muncul adalah pola peminjaman suatu judul DVD dalam suatu transaksi yang diikuti dengan peminjaman judul DVD sekuelnya pada transaksi berikutnya. Dengan asumsi satu transaksi dengan transaksi lainnya terjadi dalam waktu yang berbeda.

Contoh aplikasi nyata pencarian pola sekuensial terdapat pada permasalahan **[ZAI07]** berikut ini:

- Analisis perilaku belanja konsumen, misalnya untuk mendapatkan pola sekuensial produk yang dibeli oleh konsumen pada rentang waktu tertentu.
- Analisis kriminalitas berantai (*serial crime*), misalnya untuk mendeteksi pola sekuensial pencurian atau penipuan kartu kredit (*fraud*).
- Analisis bencana alam, misalnya untuk mendapatkan pola sekuensial kejadian alam tertentu yang mendahului suatu bencana alam.

- Analisis saham (*stocks*), misalnya untuk mendapatkan pola sekuensial jenis-jenis saham yang sering ikut turun apabila terjadi penurunan harga suatu jenis saham tertentu.
- Analisis *web log access*, misalnya untuk mendapatkan pola sekuensial halaman situs yang sering diakses berurutan.
- Analisis *DNA sequences*, misalnya untuk menemukan pola sekuensial DNA tertentu yang dikandung pada beberapa jenis asam amino (protein).

Berikut ini adalah definisi-definisi yang terkait dengan pencarian pola sekuensial (*mining sequential patterns*) dalam model matematika **[AGR95]**.

Definisi 7. *Sequence* s dinotasikan sebagai $(s_1 s_2 \dots s_n)$ dimana s_j adalah sebuah *itemset*.

Definisi 8. *Sequence* $(a_1 a_2 \dots a_n)$ disebut *subsequence* dari *sequence* $(b_1 b_2 \dots b_n)$ jika terdapat bilangan bulat $i_1 < i_2 < \cdots < i_n$ sehingga

$$a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$$

Sebagai contoh untuk memahami definisi 8, *sequence* ((3) (4 5) (8)) merupakan *subsequence* dari ((7) (3 8) (9) (4 5 6) (8)) karena (3) ⊆ (3 8), (4 5) ⊆ (4 5 6), dan (8) ⊆ (8). Namun *sequence* ((3) (5)) **bukan** *subsequence* dari ((3 5)) karena (3) ⊆ (3 5) tetapi (5) tidak memiliki pasangan lagi yang tersisa untuk dicocokan.

Definisi 9. *Sequence s* disebut *maximal sequence* jika *s* bukan merupakan *subsequence* dari *sequence* lainnya.

Definisi 10. *Mining Sequential Patterns problem* adalah mencari semua *maximal sequence* yang memenuhi *minimum support* tertentu.

Terdapat lima fase yang diperlukan untuk mendapatkan pola sekuensial dengan menggunakan algoritma *AprioriAll* yaitu *Sort Phase*, *Litemset Phase*, *Transformation Phase*, *Sequence Phase* dan *Maximal Phase*.

Tabel berikut ini akan digunakan sebagai contoh untuk menjelaskan fase-fase tesebut untuk mendapatkan pola sekuensial **[AGR95]**. Diberikan *customer transaction database* berikut yang telah terurut berdasarkan *customer ID* dan *transaction time*.

Customer ID	Transaction Time	Items Bought
1	28 Juni 2009	30
1	29 Juni 2009	90
2	28 Juni 2009	10, 20
2	29 Juni 2009	30
2	30 Juni 2009	40, 60, 70
3	28 Juni 2009	30, 50, 70
4	28 Juni 2009	30
4	29 Juni 2009	40, 70
4	30 Juni 2009	90
5	30 Juni 3009	90

Tabel 2.9 – *Customer Transaction Database*

2.5.1 *Sort Phase*

Tujuan utama fase ini adalah mendapatkan *customer sequences* dari *database* yang telah terurut. *Customer sequences* setiap *customer* diperoleh dengan menggabungkan semua transaksi, dimana masing-masing transaksi yang berisi *itemset* bisa dipandang sebagai sebuah *sequence*.

Customer ID	Customer Sequence
1	(30) (90)
2	(10 20) (30) (40 60 70)
3	(30 50 70)
4	(30) (40 70) (90)
5	(90)

Tabel 2.10 – *Customer Sequence*

2.5.2 *Litemset Phase*

Tujuan utama fase ini adalah mendapatkan semua *large itemset* (*litemset*) berdasarkan nilai *minimun support* tertentu. Untuk mendapatkan *litemset* bisa digunakan algoritma *Apriori* **[AGR94]** biasa namun terdapat perbedaan definisi *support* **[AGR95]**.

Jika pada *association rules*, *support* pada sebuah *itemset* didefinisikan sebagai jumlah kemunculan *itemset* tersebut dibagi dengan jumlah semua transaksi, tetapi pada *mining sequential patterns*, *support* didefinisikan sebagai jumlah *customer* yang memiliki *itemset* tersebut dibagi dengan jumlah semua *customer* **[AGR95]**.

Item	Association Rules Support	Sequential Pattern Support
10	1/10	1/5
20	2/10	1/5
30	4/10	4/5
40	2/10	2/5
50	1/10	1/5
60	1/10	1/5
70	3/10	3/5
90	3/10	3/5

Tabel 2.11 – Perbedaan *Support*

Large itemset (*litemset*) yang diperoleh dengan menggunakan algoritma *Apriori* dan *minimun support* **25%** (minimal dimiliki oleh 2 *customer*) serta diberikan penomoran (*mapping*) dengan bilangan bulat yang berurutan, bisa dilihat pada tabel berikut ini.

Litemset	Mapped to
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

Tabel 2.12 – *Litemsets*

Tujuan penomoran tersebut adalah untuk memperlakukan setiap *litemset* sebagai entitas tunggal sehingga bisa dibandingkan dalam *constant time* dan mengurangi waktu untuk pemeriksaan *subsequence*.

2.5.3 *Transformation Phase*

Tujuan utama dari fase ini adalah mendapatkan *transformed sequences*. *Transformed sequences* diperoleh dengan aturan berikut:

- Setiap transaksi pada *customer sequence* diganti dengan *litemset* yang bersesuaian.
- Jika sebuah transaksi tidak memiliki *litemset*, maka tidak dimasukkan ke *transformed sequences*.
- Jika sebuah *customer sequence* tidak memiliki *litemset*, maka tidak dimasukkan ke *transformed sequences*.

Tabel berikut ini merupakan hasil *transformation phase*.

Customer ID	Customer Sequence	Transformed Sequence	After Mapping
1	(30) (90)	{(30)} {(90)}	{1} {5}
2	(10 20) (30) (40 60 70)	{(30)} {(40), (70), (40 70)}	{1} {2, 3, 4}
3	(30 50 70)	{(30), (70)}	{1, 3}
4	(30) (40 70) (90)	{(30)} {(40), (70), (40 70)} {(90)}	{1} {2, 3, 4} {5}
5	(90)	{(90)}	{5}

Tabel 2.13 – *Transformed Sequences*

2.5.4 *Sequence Phase*

Tujuan utama fase ini adalah mendapatkan semua *large sequence*. *Large sequences* adalah *sequence* yang memenuhi nilai *minimum support*. Dengan menggunakan algoritma *AprioriAll* dan minimum support **25%** (minimal dimiliki 2 *customer*) diperoleh *large sequence* berikut ini **[AGR95]**.

No.	Large Sequence
1	1 5
2	1 2
3	1 3
4	1 4
5	2 3
6	2 4
7	3 4
8	1 2 3
9	1 2 4
10	1 3 4
11	2 3 4
12	1 2 3 4

Tabel 2.14 – *Large Sequences* (*mapping mode*)

2.5.5 *Maximal Phase*

Tujuan utama fase ini adalah mendapatkan semua *maximal sequences* dari *large sequences* yang telah diperoleh pada fase sebelumnya.

Berikut ini adalah *pseudocode* untuk mendapatkan *maximal sequences*.

```
for(k = n; k > 1; k--)
{
     foreach k-sequence s_k
     {
          Delete from S all subsequences of s_k
     }
}
```

Gambar 2.5 – *Pseudocode Maximal Sequences* [AGR95]

Hasil dari algoritma *maximal sequences* bisa dilihat pada tabel berikut ini.

No.	Maximal Sequence (*mapping mode*)	Maximal Sequence (*normal mode*)
1	1 5	(30) (90)
2	1 2 3 4	(30) (40) (70) (40 70)

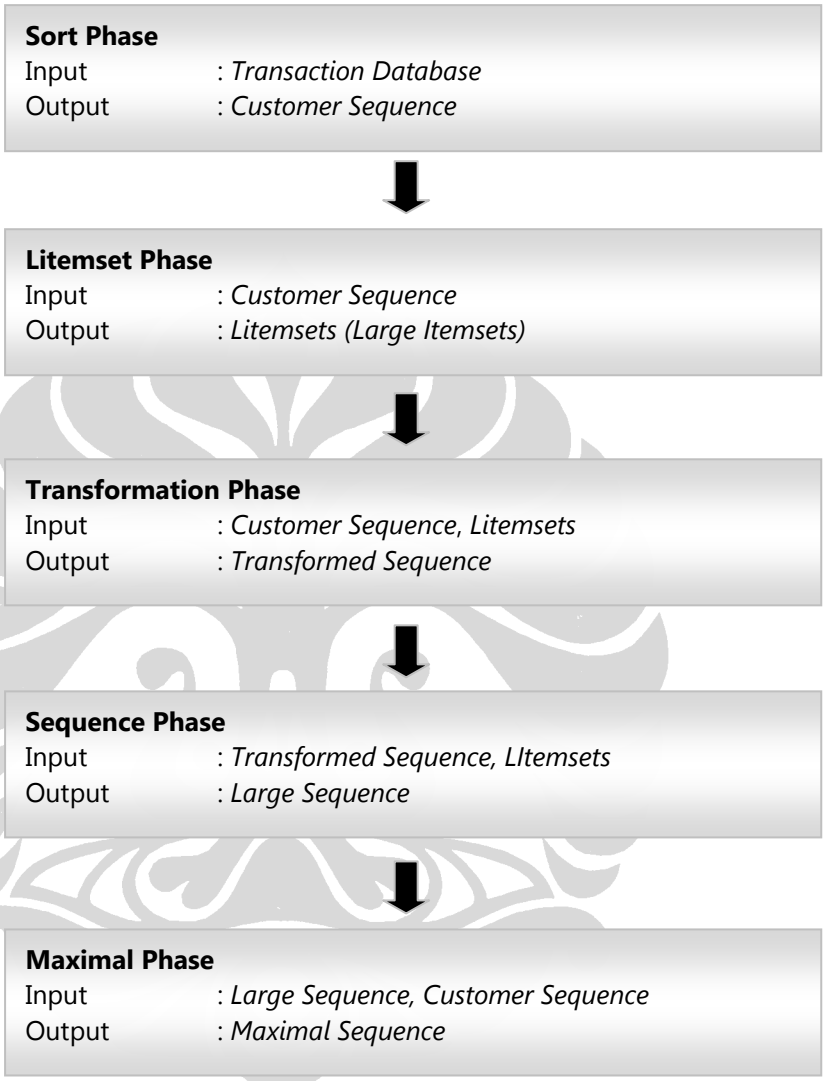
Tabel 2.15 – *Maximal Sequences*

Algoritma *maximal sequences* harus dijalankan sekali lagi pada masing-masing transaksi sehingga diperoleh hasil akhir pada tabel berikut ini.

No.	Maximal Sequence
1	(30) (90)
2	(30) (40 70)

Tabel 2.16 – *Result*

Diagram berikut ini gambaran besar tentang konsep *Mining Sequential Patterns* (MSP) yang telah dijelaskan sebelumnya.



Gambar 2.6 – Diagram MSP