# Algoritma *Greedy* (Bagian 2)

IF2251 Strategi Algoritmik

Oleh: Rinaldi Munir

## 5. Penjadwalan *Job* dengan Tenggat Waktu (*Job Schedulling with Deadlines*)

**Persoalan:**

- Ada *n* buah *job* yang akan dikerjakan oleh sebuah mesin;

- tiap *job* diproses oleh mesin selama 1 satuan waktu dan tenggat waktu (*deadline*) setiap *job i* adalah $d_i \geq 0$;

- *job i* akan memberikan keuntungan sebesar $p_i$ jika dan hanya jika *job* tersebut

- Bagaimana memilih *job-job* yang akan dikerjakan oleh mesin sehingga keuntungan yang diperoleh dari pengerjaan itu maksimum?

- Fungsi obyektif persoalan ini:

  Maksimasi $F = \sum_{i \in J} p_i$

- Solusi layak: himpunan *J* yang berisi urutan *job* yang sedemikian sehingga setiap *job* di dalam *J* selesai dikerjakan sebelum tenggat waktunya.

- Solusi optimum ialah solusi layak yang memaksimumkan *F*.

**Contoh 7.** Misalkan A berisi 4 *job* ($n$ = 4):

$(p_1, p_2, p_3, p_4) = (50, 10, 15, 30)$

$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$

Mesin mulai bekerja jam 6.00 pagi.

| *Job* | Tenggat ($d_i$) | Harus selesai sebelum pukul |
|---|---|---|
| 1 | 2 jam | 8.00 |
| 2 | 1 jam | 7.00 |
| 3 | 2 jam | 8.00 |
| 4 | 1 jam | 7.00 |

## Pemecahan Masalah dengan *Exhaustive Search*

Cari himpunan bagian (*subset*) *job* yang layak dan memberikan total keuntungan terbesar.

Contoh: $(p_1, p_2, p_3, p_4) = (50, 10, 15, 30)$
$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$

| Barisan *job* | Keuntungan ($F$) | Keterangan |
|---|---|---|
| {} | 0 | layak |
| {1} | 50 | layak |
| {2} | 10 | layak |
| {3} | 15 | layak |
| {4} | 30 | layak |
| {1, 2} | - | tidak layak |
| {1, 3} | 65 | layak |
| {1, 4} | - | tidak layak |
| {2, 1} | 60 | layak |
| {2, 3} | 25 | layak |
| {2, 4} | - | tidak layak |
| {3, 1} | 65 | layak |
| {3, 2} | - | tidak layak |
| {3, 4} | - | tidak layak |
| **{4, 1}** | **0** | **layak (Optimum!)** |
| {4, 2} | - | tidak layak |
| {4, 3} | 45 | layak |

Solusi optimum: $J = \{4, 1\}$ dengan $F = 80$.

Kompleksitas algoritma *exhaustive search* : $O(n \cdot 2^n)$.

# Pemecahan Masalah dengan Algoritma *Greedy*

- Strategi *greedy* untuk memilih *job*:

Pada setiap langkah, pilih *job i* dengan $p_i$ yang terbesar untuk menaikkan nilai fungsi obyektif *F*.

Contoh: $(p_1, p_2, p_3, p_4) = (50, 10, 15, 30)$
$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$

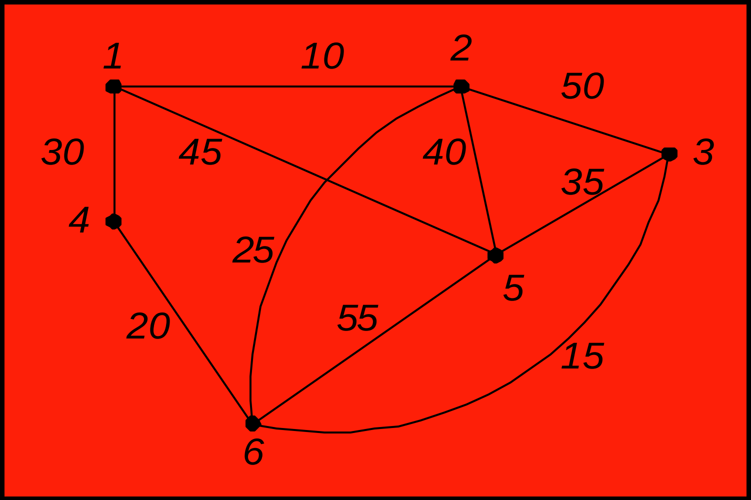
| Langkah | $J$ | $F = \sum p_i$ | Keterangan |
|---|---|---|---|
| 0 | {} | 0 | - |
| 1 | {1} | 50 | layak |
| 2 | {4,1} | 50 + 30 = 80 | layak |
| 3 | {4, 1, 3} | - | tidak layak |
| 4 | {4, 1, 2} | - | tidak layak |

Solusi optimal: $J$ = {4, 1} dengan $F$ = 80.

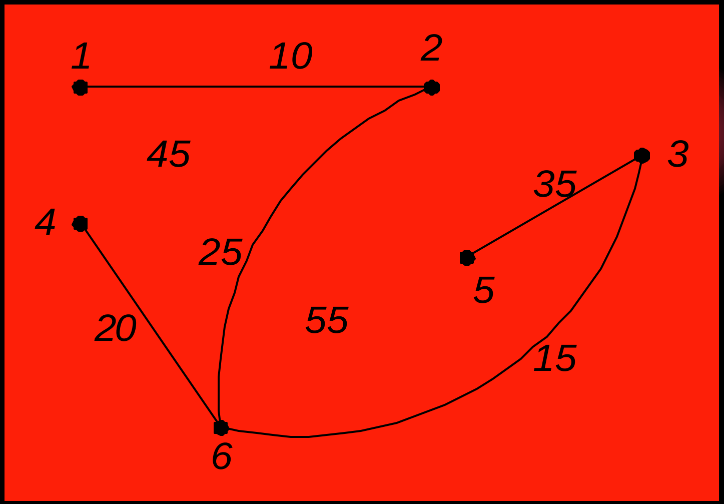
```
function JobSchedulling1(input C : himpunan_job) → himpunan_job
{ Menghasilkan barisan job yang akan diproses oleh mesin }

Deklarasi
  i : integer
  J : himpunan_job   { solusi }

Algoritma
  J ← {}
  while C ≠ {} do
     i ← job yang mempunyai p[i] terbesar
     C ← C - {i}
     if (semua job di dalam J ∪ {i} layak) then
        J ← J ∪ {i}
     endif
  endwhile
  { C = {} }
  return J
```

Kompleksitas algoritma *greedy* : $O(n^2)$.

# 6. Pohon Merentang Minimum




(a) Graf *G = (V, E)*




(b) Pohon merentang minimum

## (a) Algoritma Prim

- Strategi *greedy* yang digunakan:

  Pada setiap langkah, pilih sisi $e$ dari graf $G(V, E)$ yang mempunyai bobot terkecil dan bersisian dengan simpul-simpul di $T$ tetapi $e$ tidak membentuk sirkuit di $T$.

## *(a) Algoritma Kruskal*

- Strategi *greedy* yang digunakan:

  Pada setiap langkah, pilih sisi $e$ dari graf $G$ yang mempunyai bobot minimum tetapi $e$ tidak membentuk sirkuit di $T$.

Kompleksitas algoritma: $O(|E| \log |E|)$

## 7. Lintasan Terpendek (*Shortest Path*)

Beberapa macam persoalan lintasan terpendek:

- Lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*).
- Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
- Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*).
- Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Persoalan:

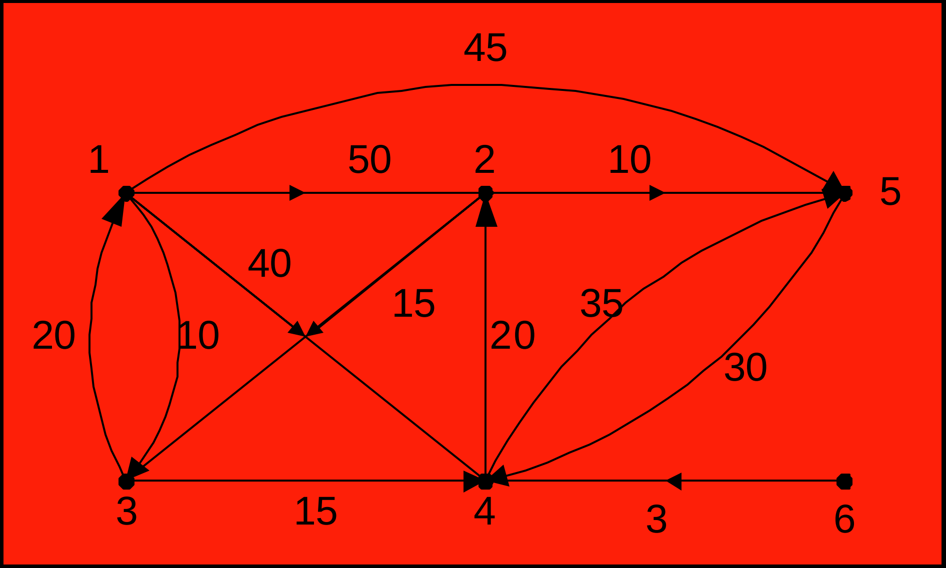
Diberikan graf berbobot $G = (V, E)$. Tentukan lintasan terpendek dari sebuah simpul asal $a$ ke setiap simpul lainnya di $G$.

Asumsi yang kita buat adalah bahwa semua sisi berbobot positif.

Strategi *greedy*:

Lintasan dibentuk satu per satu. Lintasan berikutnya yang dibentuk ialah lintasan yang meminimumkan jumlah jaraknya.

# Contoh 8.



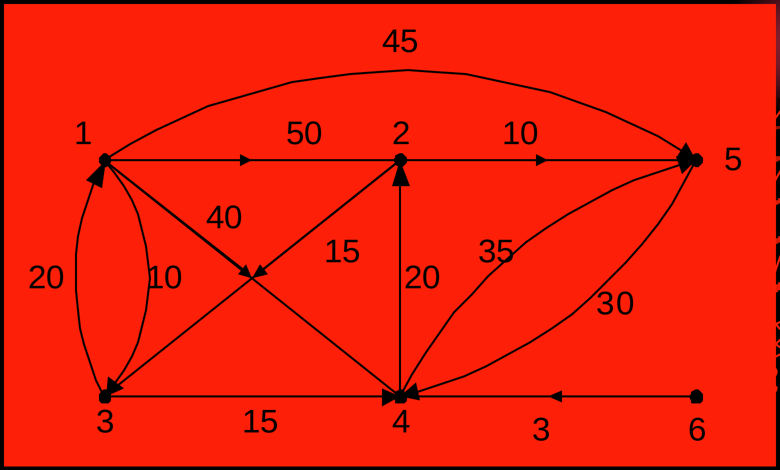

| Simpul asal | Simpul tujuan | Lintasan terpendek | Jarak |
|---|---|---|---|
| 1 | 3 | $1 \rightarrow 3$ | 10 |
| 1 | 4 | $1 \rightarrow 3 \rightarrow 4$ | 25 |
| 1 | 2 | $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$ | 45 |
| 1 | 5 | $1 \rightarrow 5$ | 45 |
| 1 | 6 | tidak ada | - |

# Algoritma Dijkstra

Strategi *greedy*:

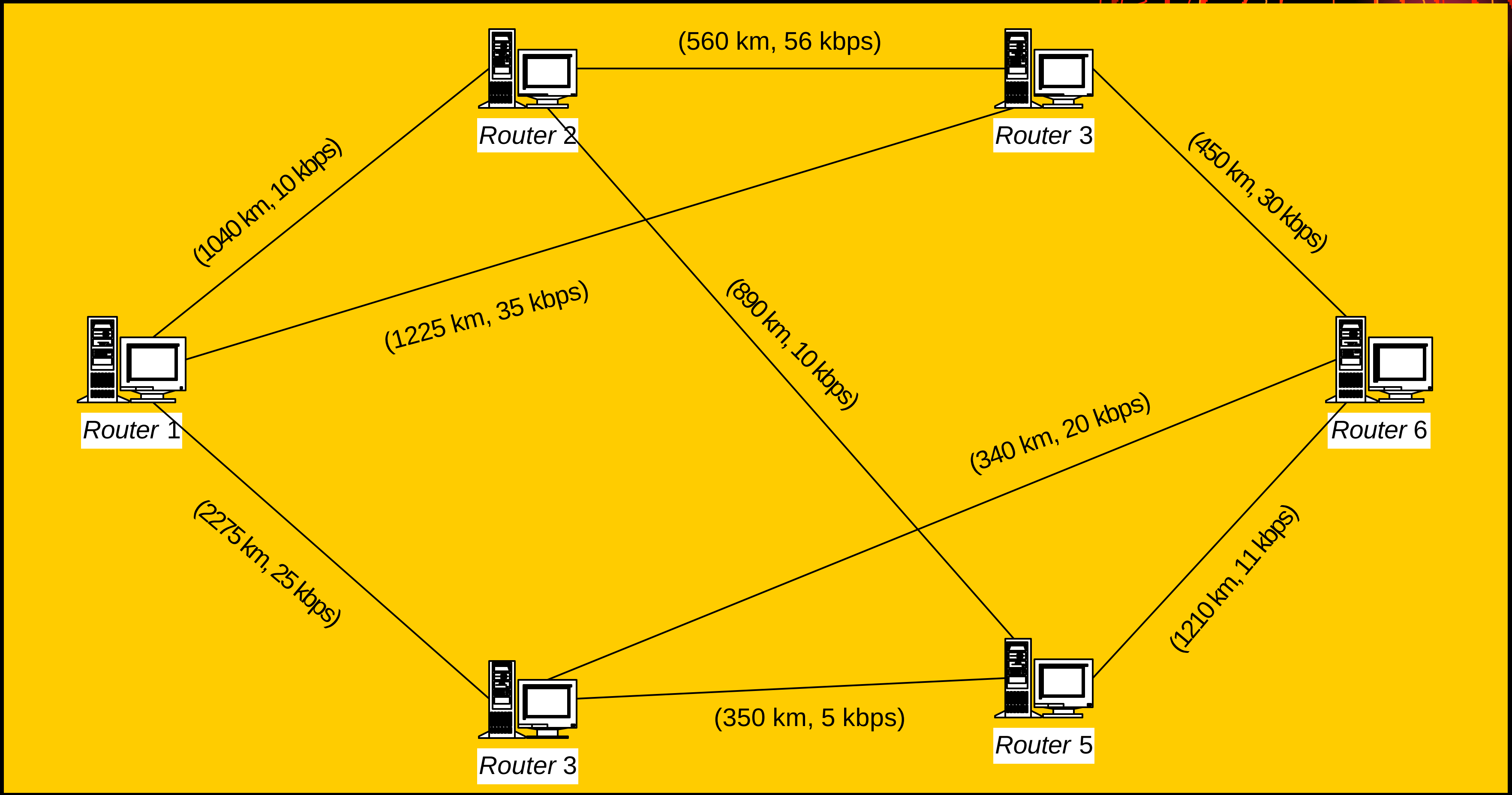
Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih.

Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih.




| Lelaran | Simpul yang dipilih | Lintasan | *S* 1 | 2 | 3 | 4 | 5 | 6 | *D* 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inisial | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 *(1,2)* | 10 *(1,3)* | 40 *(1,4)* | 45 *(1,5)* | ∞ *(1,6)* |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | ∞ | 50 *(1,2)* | 10 *(1,3)* | 40 *(1,4)* | 45 *(1,5)* | ∞ *(1,6)* |
| 2 | 3 | 1, 3 | 1 | 0 | 1 | 0 | 0 | 0 | ∞ | 50 *(1,2)* | 10 *(1,3)* | 25 *(1,3,4)* | 45 *(1,5)* | ∞ *(1,6)* |
| 3 | 4 | 1, 3, 4 | 1 | 0 | 1 | 1 | 0 | 0 | ∞ | 45 *(1,3,4,2)* | 10 *(1,3)* | 25 *(1,3,4)* | 45 *(1,5)* | ∞ *(1,6)* |
| 4 | 2 | 1, 3, 4, 2 | 1 | 1 | 1 | 1 | 0 | 0 | ∞ | 45 *(1,3,4,2)* | 10 *(1,3)* | 25 *(1,3,4)* | 45 *(1,5)* | ∞ *(1,6)* |
| 5 | 5 | 1, 5 | 1 | 1 | 1 | 1 | 1 | 0 | ∞ | 45 | 10 | 25 | 45 | ∞ |

# Aplikasi algoritma Dijkstra:
# → *Routing* pada jaringan komputer



# Lintasan terpendek (berdasarkan delai):

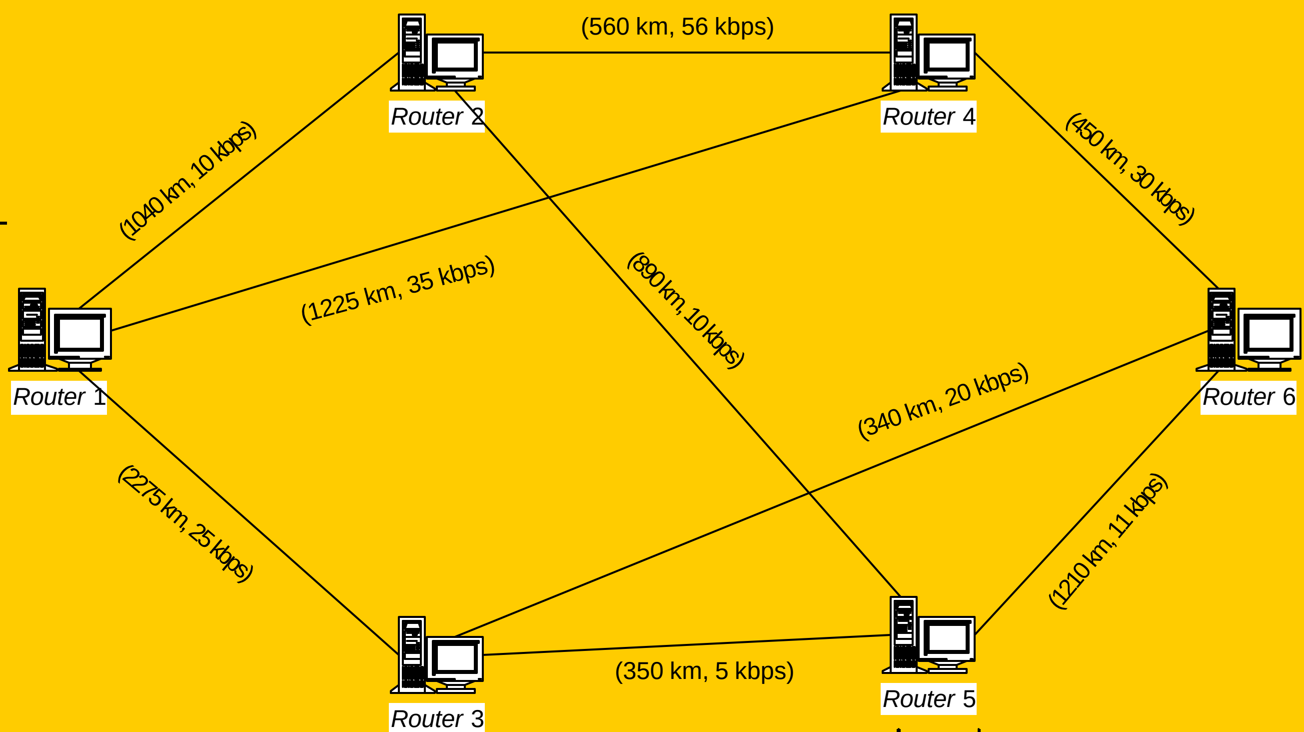
| *Router* Asal | *Router* Tujuan | Lintasan Terpendek |
|---|---|---|
| 1 | 1 | - |
| | 2 | 1, 4, 2 |
| | 3 | 1, 4, 6, 3 |
| | 4 | 1, 4 |
| | 5 | 1, 4, 2, 5 |
| | 6 | 1, 4, 6 |
| 2 | 1 | 2, 4, 1 |
| | 2 | - |
| | 3 | 2, 4, 6, 3 |
| | 4 | 2, 4 |
| | 5 | 2, 5 |
| | 6 | 2, 4, 6 |
| 3 | 1 | 3, 6, 4, 1 |
| | 2 | 3, 6, 4, 2 |
| | 3 | - |
| | 4 | 3, 6, 4 |
| | 5 | 3, 5 |
| | 6 | 3, 6 |
| 4 | 1 | 4, 1 |
| | 2 | 4, 2 |
| | 3 | 4, 6, 2 |
| | 4 | 4, 6, 3 |
| | 5 | 4, 2, 5 |
| | 6 | 4, 6 |

| *Router* Asal | *Router* Tujuan | Lintasan Terpendek |
|---|---|---|
| 5 | 1 | 5, 2, 4, 1 |
| | 2 | 5, 2 |
| | 3 | 5, 3 |
| | 4 | 5, 2, 4 |
| | 5 | - |
| | 6 | 5, 3, 6 |
| 6 | 1 | 6, 4, 1 |
| | 2 | 6, 4, 2 |
| | 3 | 6, 3 |
| | 4 | 6, 4 |
| | 5 | 6, 3, 5 |
| | 6 | - |

| Asal | Tujuan | Via |
|---|---|---|
| 2 | 1 | 4 |
| 2 | 2 | - |
| 2 | 3 | 4 |
| 2 | 4 | 2 |
| 2 | 5 | 2 |
| 2 | 6 | 4 |

| Asal | Tujuan | Via |
|---|---|---|
| 4 | 1 | 4 |
| 4 | 2 | 4 |
| 4 | 3 | 6 |
| 4 | 4 | - |
| 4 | 5 | 2 |
| 4 | 6 | 4 |

| Asal | Tujuan | Via |
|---|---|---|
| 1 | 1 | - |
| 1 | 2 | 4 |
| 1 | 3 | 4 |
| 1 | 4 | 4 |
| 1 | 5 | 4 |
| 1 | 6 | 4 |




| Asal | Tujuan | Via |
|---|---|---|
| 6 | 1 | 4 |
| 6 | 2 | 4 |
| 6 | 3 | 6 |
| 6 | 4 | 6 |
| 6 | 5 | 3 |
| 6 | 6 | - |

| Asal | Tujuan | Via |
|---|---|---|
| 3 | 1 | 6 |
| 3 | 2 | 6 |
| 3 | 3 | - |
| 3 | 4 | 6 |
| 3 | 5 | 3 |
| 3 | 6 | 3 |

| Asal | Tujuan | Via |
|---|---|---|
| 5 | 1 | 2 |
| 5 | 2 | 5 |
| 5 | 3 | 5 |
| 5 | 4 | 2 |
| 5 | 5 | - |
| 5 | 6 | 3 |

8. **Pemampatan Data dengan Algoritma Huffman**

Prinsip kode Huffman:

- karakter yang paling sering muncul di dalam data dengan kode yang lebih pendek;
- sedangkan karakter yang relatif jarang muncul dikodekan dengan kode yang

# *Fixed-length code*

| Karakter | *a* | *b* | *c* | *d* | *e* | *f* |
| --- | --- | --- | --- | --- | --- | --- |
| Frekuensi | 45% | 13% | 12% | 16% | 9% | 5% |
| Kode | 000 | 001 | 010 | 011 | 100 | 111 |

‘bad’ dikodekan sebagai ‘001000011’

Pengkodean 100.000 karakter membutuhkan 300.000 bit.

## *Variable-length code (Huffman code)*

| Karakter | *a* | *b* | *c* | *d* | *e* | *f* |
|---|---|---|---|---|---|---|
| Frekuensi | 45% | 13% | 12% | 16% | 9% | 5% |
| Kode | 0 | 101 | 100 | 111 | 1101 | 1100 |

‘bad’ dikodekan sebagai ‘1010111 ’

Pengkodean 100.000 karakter membutuhkan $(0,45 \times 1 + 0,13 \times 3 + 0,12 \times 3 + 0,16 \times 3 + 0,09 \times 4 + 0,05 \times 4) \times 100.000 = 224.000$ bit

Nisbah pemampatan:

$(300.000 – 224.000)/300.000 \times 100\% = 25,3\%$

Algoritma Greedy untuk Membentuk Kode Huffman:

1. Baca semua karakter di dalam data untuk menghitung frekuensi kemunculan setiap karakter. Setiap karakter penyusun data dinyatakan sebagai pohon bersimpul tunggal. Setiap simpul di-*assign* dengan frekuensi kemunculan karakter tersebut.

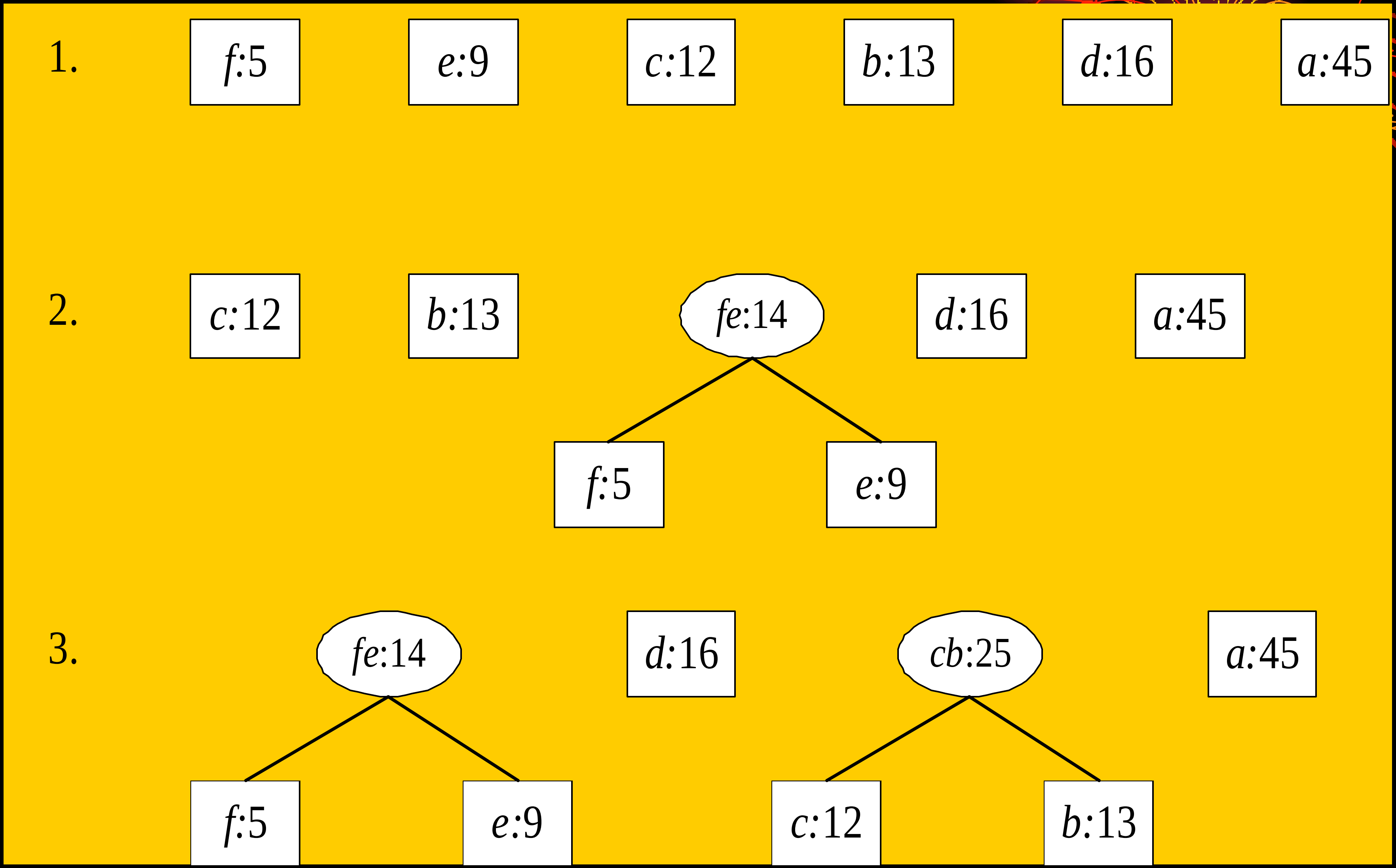
2. Terapkan strategi *greedy* sebagai berikut: gabungkan dua buah pohon yang mempunyai frekuensi terkecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.

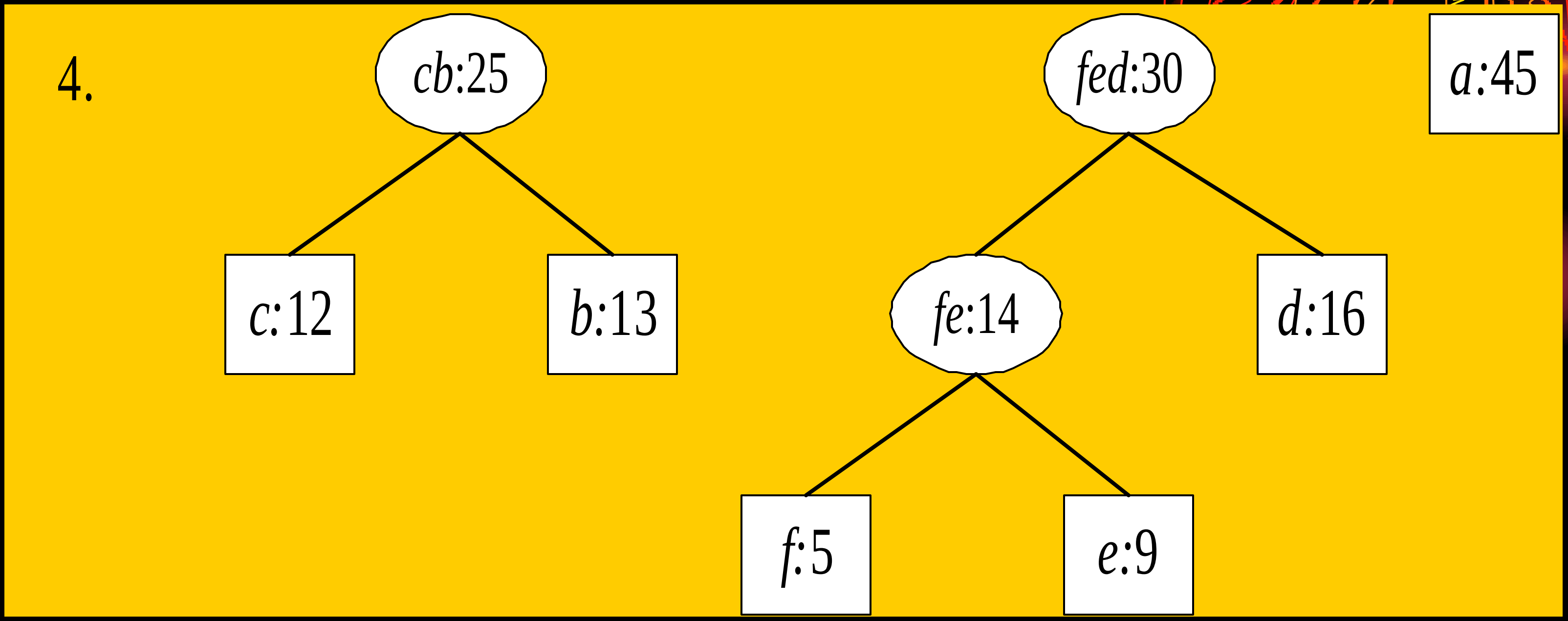
3. Ulangi langkah 2 sampai hanya tersisa satu buah pohon Huffman.

Kompleksitas algoritma Huffman: $O(n \log n)$ untuk $n$ karakter.

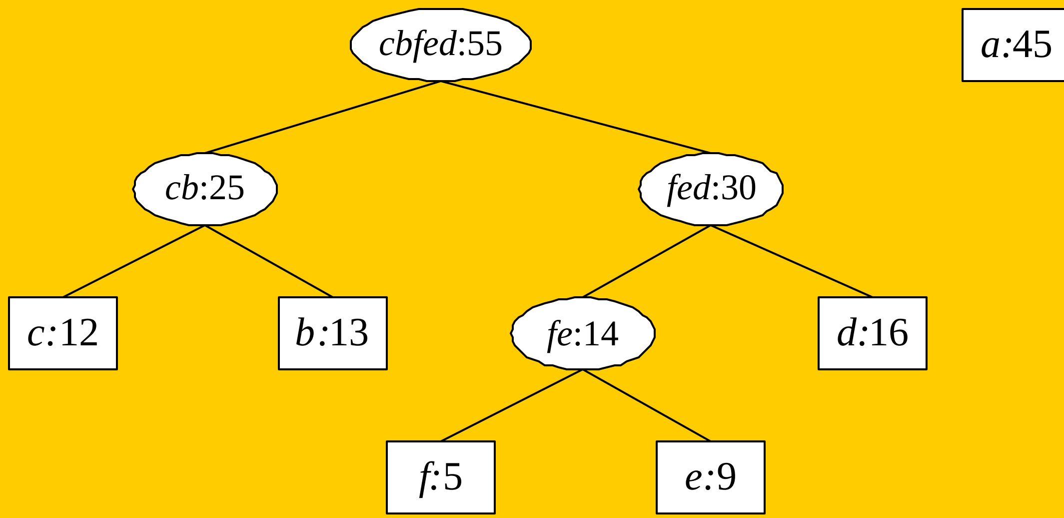
- **Contoh 9.**

| Karakter | *a* | *b* | *c* | *d* | *e* | *f* |
|---|---|---|---|---|---|---|
| Frekuensi | 45 | 13 | 12 | 16 | 9 | 5 |

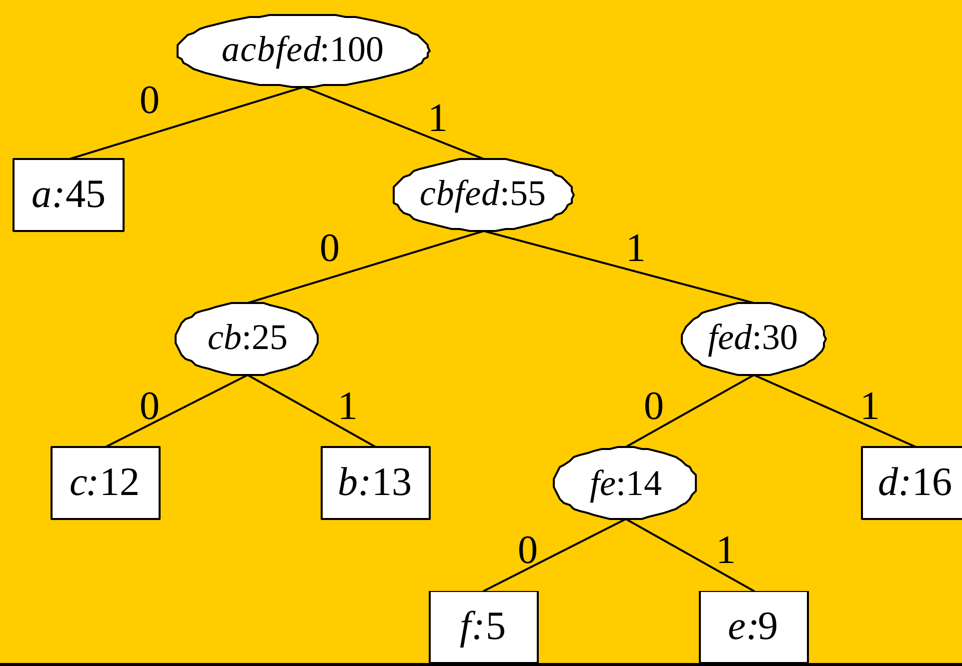

1.
f:5
e:9
c:12
b:13
d:16
a:45
2.
c:12
b:13
fe:14
d:16
a:45
f:5
e:9
3.
fe:14
d:16
cb:25
a:45
f:5
e:9
c:12
b:13


4.
cb:25
fed:30
a:45
c:12
b:13
fe:14
d:16
f:5
e:9

5.


cbfed:55
a:45
cb:25
fed:30
c:12
b:13
fe:14
d:16
f:5
e:9

6


acbfed:100
0
1
a:45
cbfed:55
0
1
cb:25
fed:30
0
1
0
1
c:12
b:13
fe:14
d:16
0
1
f:5
e:9

# Pecahan Mesir (*Egyptian Fraction*)

**Persoalan**: Diberikan pecahan *p*/*q*. Dekomposisi pecahan menjadi jumlah dari sejumlah … erbeda:

$$\frac{p}{q} = \frac{1}{k_1} + \frac{1}{k_2} + \ldots + \frac{1}{k_n}$$

yang d… ini… k…

$$\frac{2}{5} = \frac{1}{3} + \frac{1}{15}$$

$$\frac{5}{7} = \frac{1}{2} + \frac{1}{5} + \frac{1}{70}$$

$$\frac{87}{100} = \frac{1}{2} + \frac{1}{5} + \frac{1}{11}$$

Contoh:

- Pecahan yang diberikan mungkin mempunyai lebih dari satu representasi Mesir

  Contoh: 8/15 = 1/3 + 1/5

  8/15 = 1/2 + 1/30

- Kita ingin mendekompoisinya dengan jumlah unit pecahan <u>sesedikit</u> mungkin

Algoritma *greedy*: pada setiap langkah, tambahkan unit pecahan terbesar ke representasi yang baru terbentuk yang jumlahnya tidak melebihi nilai pecahan yang diberikan

Rincian algoritma:

1. Mulai dengan $i = 1$
2. Jika $p = 1$, maka $k_i = q$. STOP
3. $1/k_i$ = pecahan terbesar yang lebih kecil dari $p/q$
4. $p/q = p/q - 1/k_i$
5. Ulangi langkah 2.

- Contoh keluaran:

  8/15 = 1/2 + 1/30

  tetapi,

  26/133 = 1/6 + 1/35 + 1/3990

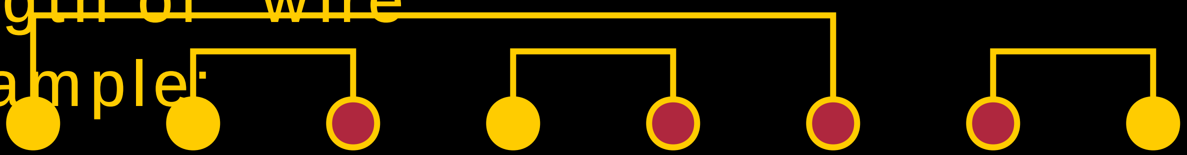
  seharusnya

  26/133 = 1/7 + 1/19

- Kesimpulan: algoritma *greedy* untuk masalah pecahan Mesir tidak selalu optimal

# Connecting wires

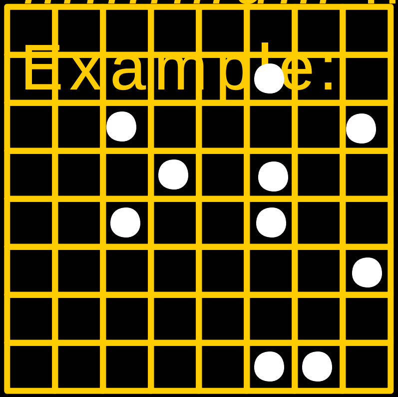
- There are n white dots and n black dots, equally spaced, in a line
- You want to connect each white dot with some one black dot, with a minimum total length of “wire”
- Example:



- Total wire length above is 1 + 1 + 1 + 5 = 8
- Do you see a greedy algorithm for doing this?
- Does the algorithm guarantee an optimal solution?
  - Can you prove it?

# Collecting coins

- A checkerboard has a certain number of coins on it
- A robot starts in the upper-left corner, and walks to the bottom left-hand corner
  - The robot can only move in two directions: right and down
  - The robot collects coins as it goes
- You want to collect *all* the coins using the *minimum* number of robots
- Example:



- Do you see a greedy algorithm for doing this?
- Does the algorithm guarantee an optimal solution?
  - Can you prove it?
  - Can you find a counterexample?