8.2 - Teknik Greedy (part 2)

[KOMS120403]

Desain dan Analisis Algoritma (2023/2024)

Dewi Sintiari

Prodi S1 Ilmu Komputer
Universitas Pendidikan Ganesha

Week 9 (April 2024)

Daftar isi

Contoh 4.
Integer (1/0) knapsack problem

4. Integer (1/0) knapsack problem (1)

Masalah: diberikan n objek dan sebuah ransel dengan kapasitas K. Setiap objek memiliki bobot w_i dan profit p_i.

Bagaimana cara memilih objek yang akan dimasukkan ke dalam knapsack sedemikian sehingga total keuntungannya maksimal? Berat total objek tidak boleh melebihi kapasitas ransel.

Formulasi matematis dari soal 1/0 knapsack:

$$\text{Maksimalkan} \quad F = \sum_{i=1}^{n} p_i x_i$$

$$\text{atas kendala} \quad \sum_{i=1}^{n} w_i x_i \leq K$$

$$\text{dan } x_i = 0 \text{ atau } x_i = 1, \text{ untuk } i = 1, 2, \ldots, n$$

4. Integer (1/0) knapsack problem (2)

Ingatlah bahwa kompleksitas waktu dengan *exhaustive search* adalah $\mathcal{O}(n \cdot 2^n)$. Dapatkah Anda jelaskan mengapa?

Dengan menggunakan strategi greedy:

- Sertakan objek satu per satu ke ransel. Setelah dimasukkan, maka objek tidak dapat diambil kembali.
- Beberapa strategi greedy-heuristik yang dapat digunakan untuk memilih objek di ransel:
  1. Greedy by profit: pada setiap langkah, pilih objek dengan keuntungan maksimum
  2. Greedy by weight: pada setiap langkah, pilih objek dengan berat minimum
  3. Greedy by density: pada setiap langkah, pilih objek dengan nilai maksimum p_i/w_i

- Namun, tidak ada strategi di atas yang menjamin solusi optimal.

4. Integer (1/0) knapsack problem (3)

Contoh

Diberikan empat objek sebagai berikut, dan ransel berkapasitas $M = 16$.

$$(w_1, p_1) = (6, 12);\ (w_2, p_2) = (5, 15)$$

$$(w_3, p_3) = (10, 50);\ (w_4, p_4) = (5, 10)$$

Object properties				Greedy by			Optimal solution
i	w_i	p_i	p_i/w_i	weight	profit	density	
1	6	12	2	0	1	0	0
2	5	15	3	1	1	1	1
3	10	50	5	1	0	1	1
4	5	10	2	0	1	0	0
Solution set				$\{3, 2\}$	$\{2, 4, 1\}$	$\{3, 2\}$	15
Total weight				20	20	20	15
Total profit				28.2	31.0	31.5	65

4. Integer (1/0) knapsack problem (4)

Contoh

Diberikan enam objek sebagai berikut:

$$(w_1, p_1) = (100, 40);\ (w_2, p_2) = (50, 35);\ (w_3, p_3) = (45, 18)$$

$$(w_4, p_4) = (20, 4);\ (w_5, p_5) = (10, 10);\ (w_6, p_6) = (5, 2)$$

dan sebuah ransel dengan kapasitas $M = 100$.

4. Integer (1/0) knapsack problem (4)

Contoh

Diberikan enam objek sebagai berikut:

$$(w_1, p_1) = (100, 40);\ (w_2, p_2) = (50, 35);\ (w_3, p_3) = (45, 18)$$

$$(w_4, p_4) = (20, 4);\ (w_5, p_5) = (10, 10);\ (w_6, p_6) = (5, 2)$$

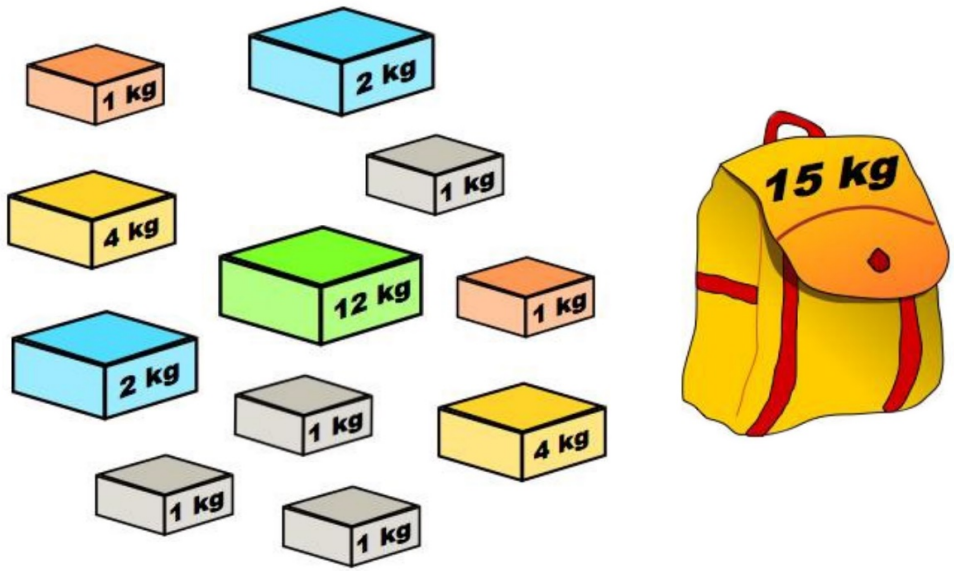
dan sebuah ransel dengan kapasitas $M = 100$.

Object properties				Greedy by			Optimal solution
i	w_i	p_i	p_i/w_i	weight	profit	density	
1	100	40	0.4	1	0	0	0
2	50	35	0.7	0	0	1	1
3	45	18	0.4	0	1	0	1
4	20	4	0.2	0	1	1	0
5	10	10	1.0	0	1	1	0
6	5	2	0.4	0	1	1	0
Total weight				100	80	85	100
Total profit				40	34	51	55

4. Integer (1/0) knapsack problem (5)

Kesimpulan:

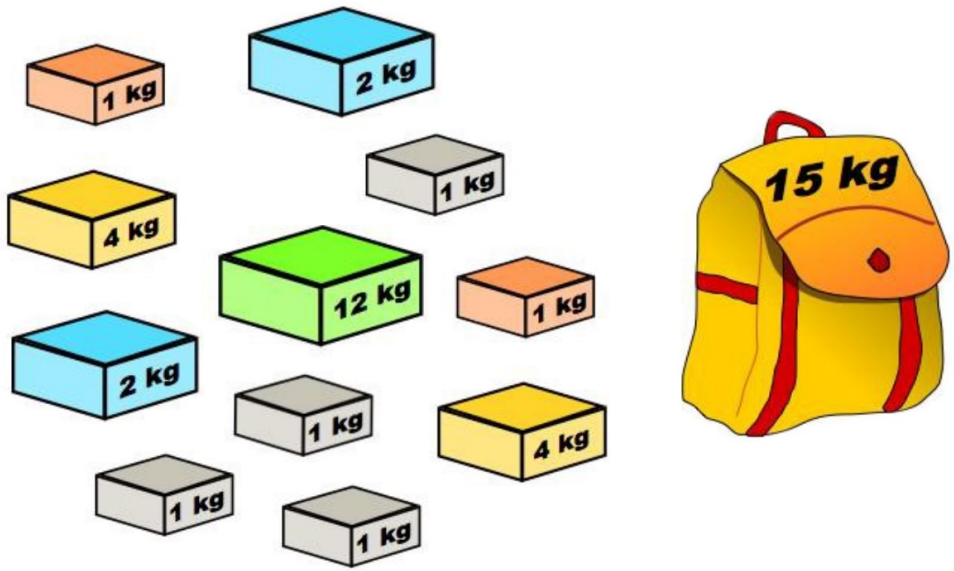
Apakah algoritma greedy selalu dapat menemukan solusi optimal untuk masalah *integer knapsack*?



4. Integer (1/0) knapsack problem (5)

Kesimpulan:

Apakah algoritma greedy selalu dapat menemukan solusi optimal untuk masalah *integer knapsack*?



Contoh 5.
Fractional knapsack problem

5. Fractional knapsack problem (1)

Fractional knapsack problem adalah varian dari masalah knapsack, tetapi penyelesaiannya tidak harus bilangan bulat, namun bisa dalam bentuk pecahan.

Formulasi permasalahan:

$$\text{Maksimalkan} \quad F = \sum_{i=1}^{n} p_i x_i$$

$$\text{sedemikian sehingga} \quad \sum_{i=1}^{n} w_i x_i \leq K$$

$$\text{dan} \quad 0 \leq x_i \leq 1, \ \text{untuk} \ i = 1, 2, \ldots, n$$

Pertanyaan: apakah mungkin untuk memecahkan masalah dengan *exhaustive search*?

5. Fractional knapsack problem (2)

Pertanyaan: apakah mungkin untuk memecahkan masalah dengan *exhaustive search*?

Karena $0 \leq x_i \leq 1$, maka terdapat tak hingga banyaknya bilangan yang mungkin untuk x_i.

Karena rentang nilai x_1 tidak diskrit, namun kontinu, jadi tidak mungkin diselesaikan dengan exhaustive search.

5. Fractional knapsack problem (3)

Contoh

Diberikan tiga objek sebagai berikut:

$$(w_1, p_1) = (18, 25);\ (w_2, p_2) = (15, 24);\ (w_3, p_3) = (10, 15)$$

dan ransel kapasitas $M = 20$.

Object properties				Greedy by		
i	w_i	p_i	p_i/w_i	profit	weight	density
1	18	25	1.4	1	0	0
2	15	24	1.6	2/15	2/3	1
3	10	15	1.5	0	1	1/2
Total weight				20	20	20
Total profit				28.2	31.0	31.5

Solusi optimal: $X = (0, 1, 1/2)$, dan keuntungan maksimum adalah $31,5$.

5. Fractional knapsack problem (4)

Teorema (Greedy by density menghasilkan solusi optimal)

Jika $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$, *maka algoritma Greedy dengan strategi memilih densitas maksimum* $\frac{p_i}{w_i}$ *memberikan solusi optimal.*

Proof.

Tugas: (berikan bukti yang serupa dengan yang dijelaskan pada "Masalah Pemilih Aktivitas" (pada slide bagian 1)) □

Algoritma:

- Hitung $\frac{p_i}{w_i}$ untuk $i = 1, 2, \ldots, n$
- Agar strategi ini berhasil, densitas $\frac{p_i}{w_i}$ diurutkan dalam urutan menurun.

5. Fractional knapsack problem (5)

```
1:  procedure FractionalKnapsack(C: objects set, K: real)
2:      for i ← 1 to n do
3:          x[i] ← 0                                   ▷ x is the solution set
4:      end for
5:      i ← 0;  totalwt ← 0;  intFrac ← True           ▷ 'totalwt': total weight, and 'intFrac': boolean var
        indicating if current object can be included fully
6:      while (i ≤ n) and intFrac do
7:          i ← i + 1
8:          if totalwt + w[i] ≤ K then
9:              x[i] ← 1                               ▷ Include object i to knapsack
10:             totalwt ← totalwt + w[i]               ▷ Include the fraction of object i to total weight
11:         else
12:             intFrac ← False
13:             x[i] ← (K − totalwt) / w[i]            ▷ Only a fraction of object i can be included to the knapsack
14:         end if
15:     end while
16:     return x
17: end procedure
```

Contoh 6.
Job scheduling with deadlines

6. Job scheduling with deadlines

Masalah: Diberikan n pekerjaan yang akan dilakukan oleh mesin. Setiap job diproses oleh mesin dalam satu satuan waktu (1 jam) dan deadline setiap job i adalah $d_i \geq 0$.

Pekerjaan i akan memberikan keuntungan p_i jika pekerjaan tidak dilakukan setelah batas waktu. Bagaimana memilih pekerjaan yang akan diproses oleh mesin agar keuntungan maksimal?

6. Job scheduling with deadlines

Contoh

Diberikan 4 pekerjaan ($n = 4$) dengan karakteristik sebagai berikut:

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

Misalkan mesin mulai bekerja pada jam 6 pagi, maka kita memiliki kendala sebagai berikut:

Job	**Deadline (d_i)**	**Must be done before**
1	2 hours	8 am
2	1 hour	7 am
3	2 hours	8 am
4	1 hour	7 am

6. Job scheduling with deadlines

Misalkan J menjadi himpunan pekerjaan, maka fungsi tujuan dari masalah ini adalah:

$$\text{Maximize } F = \sum_{i \in J} p_i$$

Himpunan solusi J adalah layak jika setiap pekerjaan di J dilakukan sebelum tenggat waktu.

Sebuah solusi optimal adalah solusi layak yang memaksimalkan F.

6. Job scheduling with deadlines

Set of jobs	Total profit (F)	Description
$\{\}$	0	feasible
$\{1\}$	50	feasible
$\{2\}$	10	feasible
$\{3\}$	15	feasible
$\{4\}$	30	feasible
$\{1,2\}$	-	not feasible
$\{1,3\}$	65	feasible
$\{1,4\}$	-	not feasible
$\{2,1\}$	60	feasible
$\{2,3\}$	25	feasible
$\{2,4\}$	-	not feasible
$\{3,1\}$	65	feasible
$\{3,2\}$	-	not feasible
$\{3,4\}$	-	not feasible
$\{4,1\}$	80	feasible $\rightarrow$ optimal
$\{4,2\}$	-	not feasible
$\{4,3\}$	45	feasible

Banyaknya pekerjaan maksimal yang bisa dilakukan: **dua pekerjaan**.

6. Job scheduling with deadlines

Kompleksitas strategi exhaustive search: $\mathcal{O}(n \cdot 2^n)$.

Strategi greedy untuk memilih pekerjaan:

> *Pada setiap langkah, pilih pekerjaan i dengan p_i terbesar untuk meningkatkan nilai objektif F (atau total profit)".*

Misalkan J adalah himpunan pekerjaan yang dipilih. Kita menempatkan pekerjaan i di J dalam urutan tertentu sedemikian sehingga semua pekerjaan selesai sebelum tenggat waktu.

Langkah	J	$F = \Sigma p_i$	**Keterangan**
0	$\{\}$	0	-
1	$\{1\}$	50	feasible
2	$\{4, 1\}$	$50 + 30 = 80$	feasible
3	$\{4, 1, 3\}$	-	not feasible
4	$\{4, 1, 2\}$	-	not feasible

6. Job scheduling with deadlines

Contoh

Diberikan 4 pekerjaan dengan spesifikasi sebagai berikut:

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

Step	J	$F = \sum_i p_i$	**Description**
0	$\{\}$	0	feasible
1	$\{1\}$	50	feasible
2	$\{4, 1\}$	50+30 = 80	feasible
3	$\{4, 1, 3\}$	-	not feasible
4	$\{4, 1, 2\}$	-	not feasible

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

6. Job scheduling with deadlines

Algorithm 1 Job scheduling algorithm

```
1:  procedure JobSchdlng(d[1..n], p[1..n]: array of integers)
2:      J ← {}
3:      for i ← 1 to n do
4:          k ← job with highest profit
5:          if all jobs in J ∪ {k} is feasible then
6:              J ← J ∪ {k}
7:          end if
8:      end for
9:      return J
10: end procedure
```

Kompleksitas:

- Memilih pekerjaan dengan p_i terbesar: $\mathcal{O}(n)$
- While loop n kali ($=$ jumlah pekerjaan di C)
- Kompleksitas *job scheduling*: $\mathcal{O}(n^2)$

6. Job scheduling with deadlines

Revisi: pekerjaan diurutkan berdasarkan keuntungan mereka (secara menurun/*descending*).

Algorithm 2 Job scheduling algorithm

```
1: procedure JobSchdlng2(d[1..n], p[1..n]: array of integers)
2:     J ← {1}
3:     for i ← 2 to n do
4:         if all jobs in J ∪ {i} is feasible then
5:             J ← J ∪ {i}
6:         end if
7:     end for
8:     return J
9: end procedure
```

Kompleksitas: ?

Contoh 7. Kode Huffman

7. Kode Huffman (1)

Prinsip encoding dan decoding

Encoding/decoding adalah terjemahan pesan yang mudah dipahami.

Encoding: cara karakter apa pun dipahami dalam penyimpanan atau transmisi komputer dari satu mesin ke mesin lain.

Decoding: proses mengembalikan pesan yang disandikan ke pesan asli.

7. Kode Huffman (2)

Fixed-length versus Variable-length codes

Skema fixed length encoding menggunakan
jumlah byte yang konstan/*fixed* untuk mewakili karakter yang berbeda.

Skema variable length encoding menggunakan jumlah byte yang berbeda
untuk mewakili karakter yang berbeda.

7. Kode Huffman (3)

Kode Huffman digunakan untuk kompresi data.

Kode panjang tetap (fixed-length code)

Diberikan sebuah pesan dengan panjang 100.000 karakter dengan frekuensi sebuah huruf muncul dalam pesan sebagai berikut:

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

Contoh: penyandian 'bad' adalah 001000011

Dengan metode ini, pengkodean 100.000 karakter membutuhkan 300.000 bit.

7. Kode Huffman (3)

Kode Huffman digunakan untuk kompresi data.

Kode panjang tetap (fixed-length code)

Diberikan sebuah pesan dengan panjang 100.000 karakter dengan frekuensi sebuah huruf muncul dalam pesan sebagai berikut:

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

Contoh: penyandian 'bad' adalah 001000011

Dengan metode ini, pengkodean 100.000 karakter membutuhkan 300.000 bit.

Prinsip dasar Kode Huffman:

- semakin sering karakter muncul, semakin pendek *encoding*, dan sebaliknya.

7. Kode Huffman (4)

Kode Huffman panjang tak-konstan (variable-length code)

Karakter	a	b	c	d	e	f
Frekuensi	45%	13%	12%	16%	9%	5%
Encoding	0	101	100	111	1100	1100

Contoh: penyandian bad adalah 1010111

Dengan metode ini, pengkodean 100.000 karakter membutuhkan:

$$(0.45 \times 1 + 0.13 \times 3 + 0.12 \times 3 + 0.16 \times 3 + 0.09 \times 4 + 0.05 \times 4) \times 10^5$$
$$= 224,000 \text{ bits}$$

Rasio kompresi $= \frac{300{,}000-224{,}000}{300{,}000} \times 100\% = 25,5\%.$

7. Kode Huffman (5)

- Algoritma greedy untuk membentuk Kode Huffman bertujuan untuk meminimalkan panjang kode biner untuk semua karakter dalam pesan $(M_1, M_2, \ldots, M_n)$.
- Kita membangun weighted binary tree. Setiap *simpul daun/leaf node* menunjukkan karakter dalam pesan, dan *simpul dalam/internal nodes* menunjukkan penggabungan karakter tersebut.
- Setiap sisi dalam pohon diberi label 0 atau 1 secara konsisten (misalnya: kiri diberi '0' dan kanan diberi '1').
- Meminimalkan kode biner untuk setiap karakter sama dengan meminimalkan panjang lintasan dari akar ke daun.

7. Kode Huffman (6)

Algoritma:

1. Hitung frekuensi setiap karakter dalam pesan. Representasikan setiap karakter dengan pohon dengan satu simpul (atau single-node tree), dan setiap simpul dipasangkan dengan frekuensi karakter yang sesuai.

2. Terapkan strategi greedy: di setiap langkah, gabungkan dua pohon yang memiliki frekuensi terkecil di akar. Akar baru memiliki frekuensi sama dengan jumlah frekuensi dari dua pohon yang menyusunnya.

3. Ulangi langkah ke-2 sampai akhirnya kita mendapatkan satu pohon Huffman. Ini membentuk pohon biner.

4. Berikan label pada setiap sisi pohon dengan 0 atau 1 (misalnya sisi berorientasi kiri diberi label 0 dan sisi berorientasi kanan diberi label 1.

5. Setiap lintasan dari akar, setiap daun pohon mewakili string biner untuk setiap karakter, dengan frekuensi seperti yang ditunjukkan pada daun yang sesuai.

7. Kode Huffman (7)

Berapakah kompleksitas waktunya?

$\mathcal{O}(n \log n)$

- Gunakan *heap*[*] untuk menyimpan bobot setiap pohon, setiap iterasi membutuhkan waktu sebanyak $\mathcal{O}(\log n)$ untuk menentukan bobot termkecil dan menyisipkan bobot baru.
- Terdapat sebanyak $\mathcal{O}(n)$ iterasi, satu untuk setiap item.

[*] `https://www.wikiwand.com/en/Heap_(data_structure)`

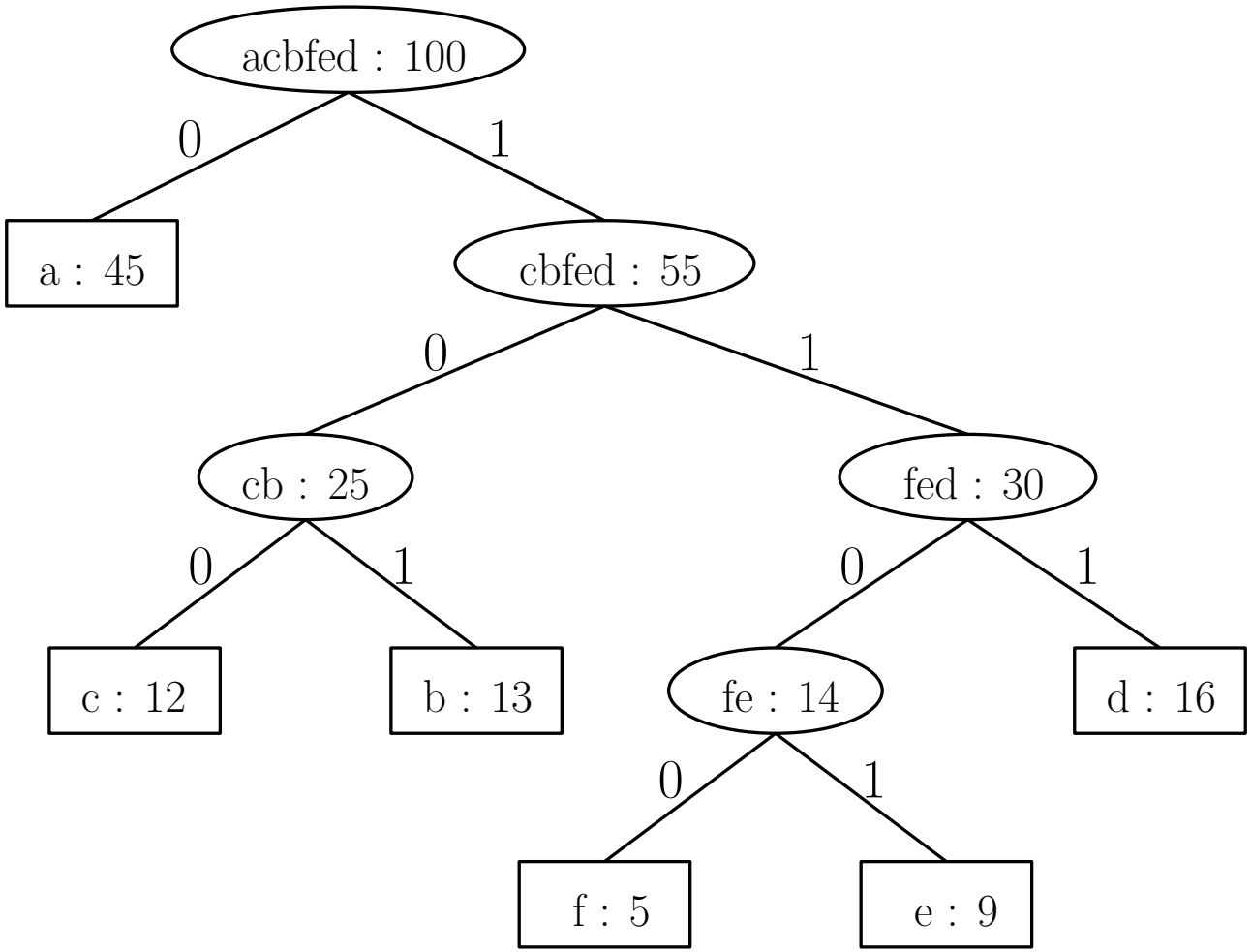
7. Kode Huffman (8)

Latihan: Diberikan pesan dengan panjang 100. Pesan terdiri dari huruf a, b, c, d, e, f. Frekuensi setiap huruf dalam pesan adalah sebagai berikut:

Karakter	a	b	c	d	e	f
Frekuensi	45%	13%	12%	16%	9%	5%

Temukan kode Huffman untuk setiap karakter dalam pesan.

7. Kode Huffman (10)



7. Kode Huffman (11)

Huffman code:

- a : 0
- b : 101
- c : 100
- d : 111
- e : 1101
- f : 1100

Contoh 8.
Traveling salesman problem

8. Traveling salesman problem (1)

Diberikan daftar kota dan jarak antara setiap pasangan kota, berapakah rute terpendek yang mengunjungi setiap kota tepat satu kali dan kembali ke kota asal?

Contoh:

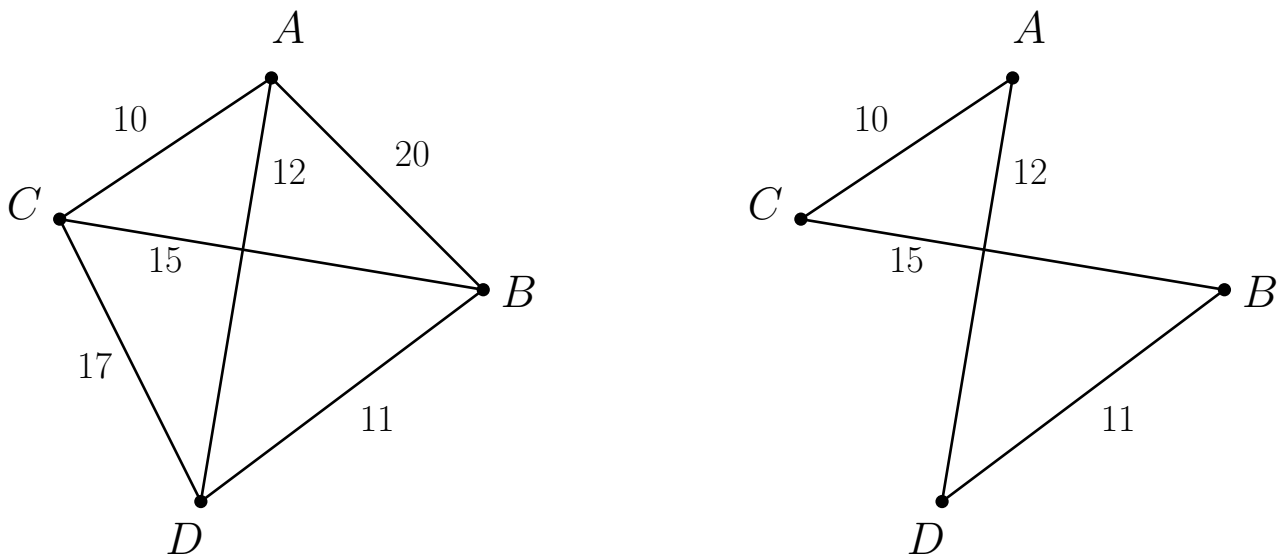


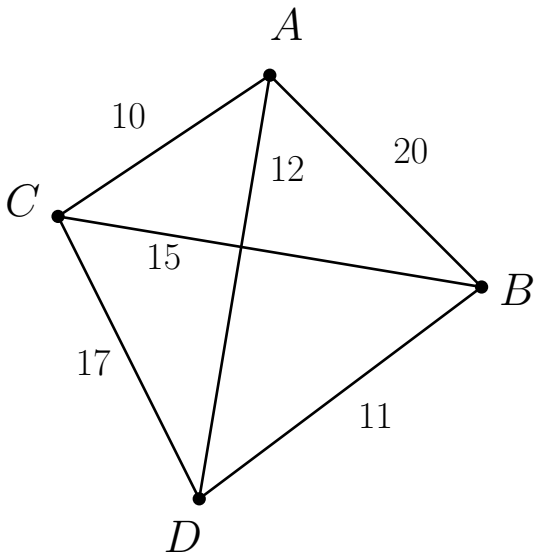
Figure: Graf dengan sisi berbobot dan solusi dari TSP-nya

8. Traveling salesman problem (2): algorithm and example

1. Diberikan graf berikut, kita ingin mencari TSP dari simpul A. Misalkan simpul dari graf input G menjadi: $v_1, v_2 \ldots, v_n$

2. Misalkan tur dimulai dari v_1. Simpul berikutnya dipilih "secara greedy"

   - Pada setiap langkah i, pilih simpul v_j yang bobot sisinya (v_i, v_j) diminimalkan.



	A	B	C	D
A	-	20	10	12
B	20	-	15	11
C	10	15	-	17
D	12	11	17	-

Solusi greedy: $A \xrightarrow{10} C \xrightarrow{15} B \xrightarrow{11} D \xrightarrow{12} A$, dengan bobot total = 48.

Bagian 4. Kekurangan algoritma greedy

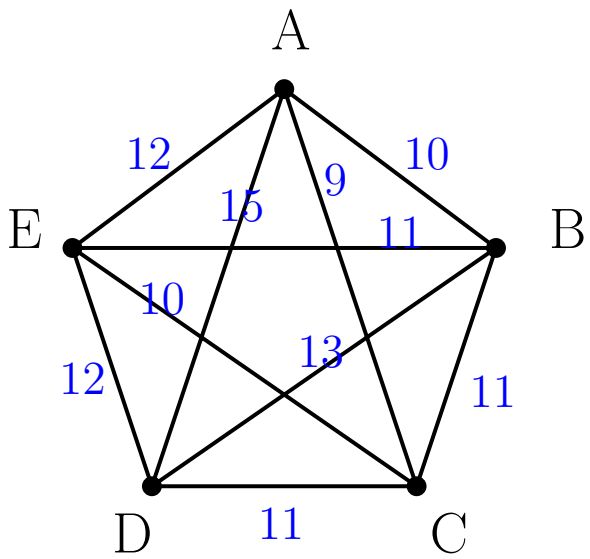
Kekurangan algoritma greedy

Warning! Optimum global belum tentu merupakan solusi optimal. Ini bisa berupa solusi *sub-optimum* atau solusi *pseudo-optimum*.

- Algoritma greedy tidak mencoba semua kandidat solusi yang mungkin (seperti dalam *exhaustive search*).
- Ada banyak fungsi $\mathrm{SELECTION}$ yang berbeda, jadi kita harus memilih fungsi yang sesuai untuk mendapatkan solusi yang optimal.

Solusi greedy tidak selalu optimal (1)

Diberikan graf berikut, kita ingin mencari TSP yang dimulai dari simpul A.



	A	B	C	D	E
A	-	10	9	15	12
B	10	-	11	13	11
C	9	11	-	11	10
D	15	13	11	-	12
E	12	11	10	12	-

Greedy solution: $A \xrightarrow{9} C \xrightarrow{10} E \xrightarrow{11} B \xrightarrow{13} D \xrightarrow{15} A$ dengan bobot $= 58$.

Optimal solution: $A \xrightarrow{10} B \xrightarrow{11} C \xrightarrow{11} D \xrightarrow{12} E \xrightarrow{12} A$ dengan bobot $= 56$.

Solusi greedy tidak selalu optimal (2)

Contoh (Greedy solution $\neq$ optimal solution)

1. *Set of coins:* $\{1, 3, 4, 5\}$, *amount of money: 7*
   - *Greedy solution:* $7 = 5 + 1 + 1$ *(3 coins)*
   - *Optimal solution:* $7 = 4 + 3$ *(2 coins)*
2. *Set of coins:* $\{1, 7, 10\}$, *amount of money: 15*
   - *Greedy solution:* $15 = 10 + 1 + 1 + 1 + 1 + 1$ *(6 coins)*
   - *Optimal solution:* $15 = 7 + 7 + 1$ *(3 coins)*
3. *Set of coins:* $\{1, 10, 15\}$, *amount of money: 20*
   - *Greedy solution:* $20 = 15 + 1 + 1 + 1 + 1 + 1$ *(6 coins)*
   - *Optimal solution:* $20 = 10 + 10$ *(2 coins)*

Fun fact: untuk sistem IDR, algoritma greedy selalu memberikan solusi optimal untuk masalah pertukaran koin

Kekurangan algoritma greedy

Tapi mengapa algoritma greedy masih digunakan?

- Dibandingkan dengan menggunakan algoritma waktu eksponensial untuk mendapatkan solusi optimal aktual, algoritma greedy dapat digunakan untuk mendapatkan perkiraan solusi optimal (misalnya, pada penyelesaian TSP).
- Jika algoritma greedy menghasilkan solusi optimal, maka hal ini harus dibuktikan secara matematis.
- Namun, lebih mudah untuk menunjukkan bahwa solusi greedy tidak optimal (yakni dengan memberikan contoh penyangkal).

end of slide...