# IN244 – Strategi Algoritmik

10 – Program Dinamis

# Program Dinamis

- metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*)
- sehingga solusi dari persoalan dapat dipandang dari **serangkaian keputusan** yang saling berkaitan.
- Istilah "dinamis" muncul karena perhitungan solusi menggunakan tabel yang ukurannya dapat berkembang.
- Kata "program" tidak berhubungan dengan pemrograman.

# Program Dinamis

Karakteristik penyelesaian persoalan dengan Program Dinamis:

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi dibangun secara bertahap; solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. kita menggunakan persyaratan **optimasi** dan **kendala** untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

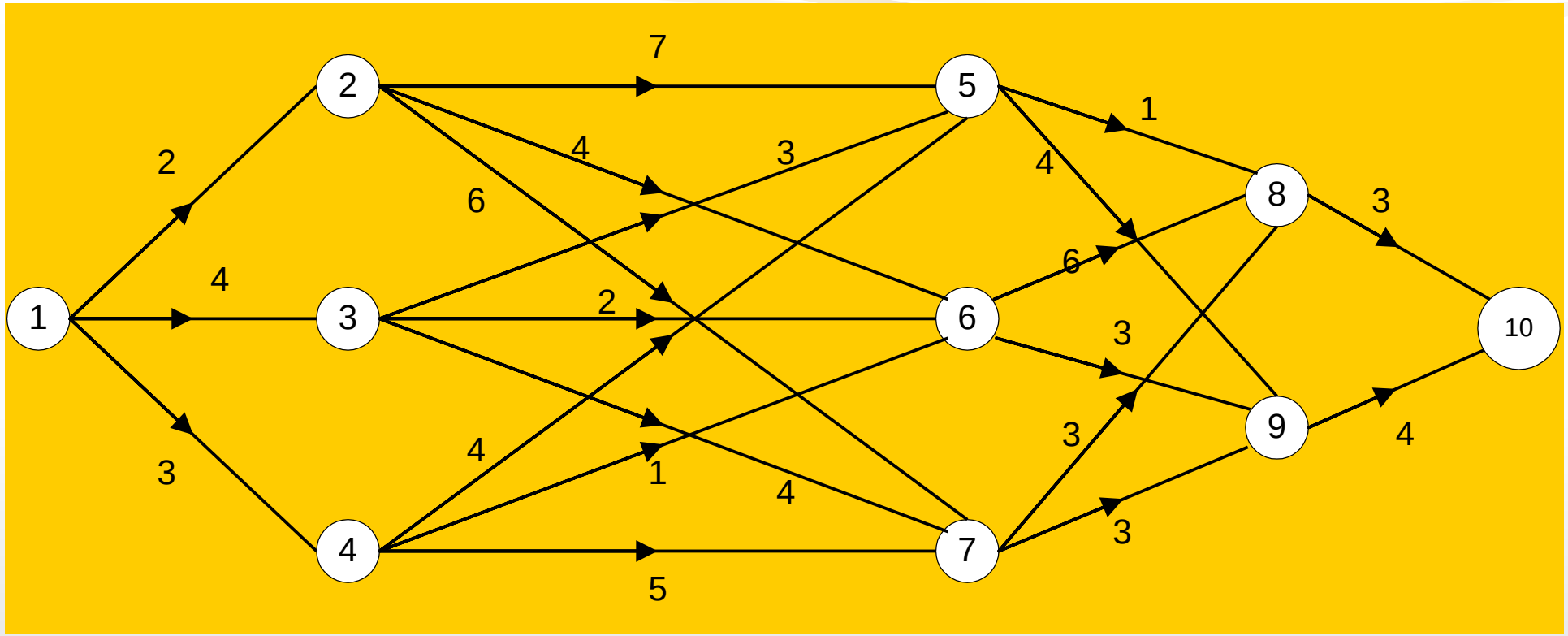
# Program Dinamis vs Greedy

- Perbedaan Algoritma *Greedy* dengan Program Dinamis:

➢ Greedy:
  - hanya satu rangkaian keputusan yang dihasilkan

➢ Program dinamis:
  - lebih dari satu rangkaian keputusan yang dipertimbangkan.

# Tinjau graf di bawah ini. Kita ingin menemukan lintasan terpendek dari 1 ke 10.



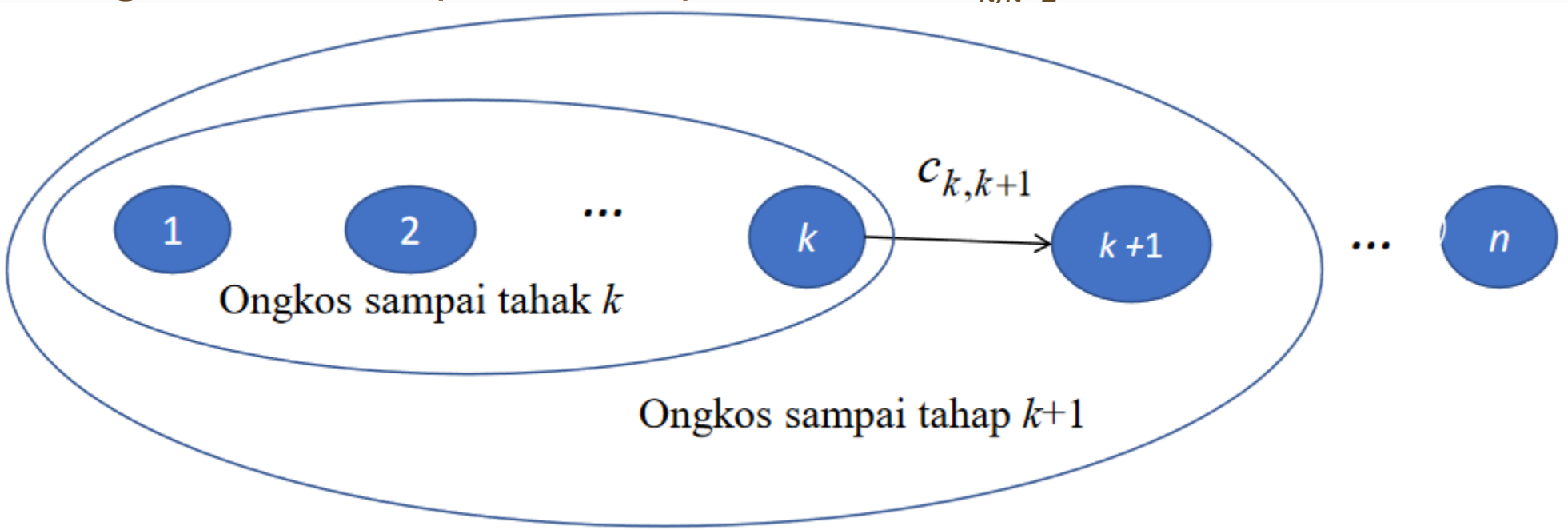

*Greedy*: 1 – 2 – 6 – 9 – 10 dengan *cost* = 2 + 4 + 3 + 4 = 13 (belum optimal)

# Prinsip Optimalitas

- Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan **Prinsip Optimalitas**.
- Prinsip Optimalitas:
  - *jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal*.
  - Prinsip optimalitas : jika kita bekerja dari tahap k ke tahap k + 1, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal.

# Prinsip Optimalitas

- ongkos pada tahap $k$ +1 = (ongkos yang dihasilkan pada tahap $k$ ) + (ongkos dari tahap $k$ ke tahap $k$ + 1 atau $c_{k,k+1}$ )



# Karakteristik Persoalan (1)

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.

# Karakteristik Persoalan (2)

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.

4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.

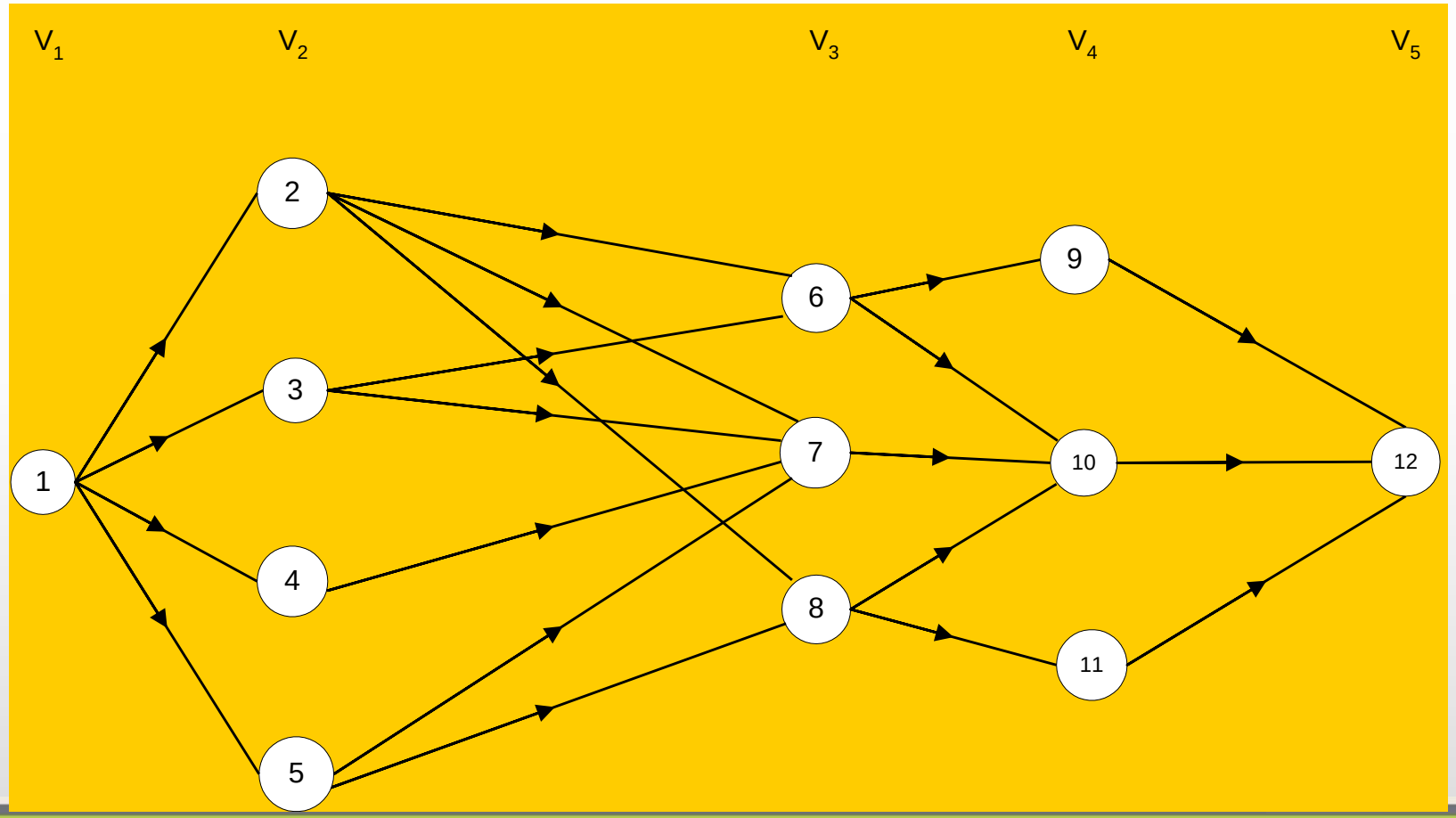
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.

6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.

7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *k* memberikan keputusan terbaik untuk setiap status pada tahap *k* + 1.

8. Prinsip optimalitas berlaku pada persoalan tersebut.

**Graf multitahap** (*multistage graph*). Tiap simpul di dalam graf tersebut menyatakan status, sedangkan $V_1$, $V_2$, ... menyatakan tahap.



# Pendekatan Program Dinamis

Misalkan $x_1, x_2, \ldots, x_n$ menyatakan peubah (*variable*) keputusan yang harus dibuat masing-masing untuk tahap 1, 2, …, $n$.

- Program dinamis maju (forward/ up-down)
  - Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap $n$.
  - Runtunan peubah keputusan adalah $x_1, x_2, \ldots, x_n$.
- Program dinamis mundur (backward/ bottom-up)
  - Program dinamis bergerak mulai dari tahap $n$, terus mundur ke tahap $n-1$, $n-2$, dan seterusnya sampai tahap 1.
  - Runtunan peubah keputusan adalah $x_n, x_{n-1}, \ldots, x_1$.

# Prinsip Optimalitas

Prinsip optimalitas pada **PD maju**:

- ongkos pada tahap $k+1$ = (ongkos yang dihasilkan pada tahap $k$) + (ongkos dari tahap $k$ ke tahap $k+1$)

$$k = 1, 2, \ldots, n-1$$

Prinsip optimalitas pada **PD mundur**:

- ongkos pada tahap $k$ = (ongkos yang dihasilkan pada tahap $k+1$) + (ongkos dari tahap $k+1$ ke tahap $k$)

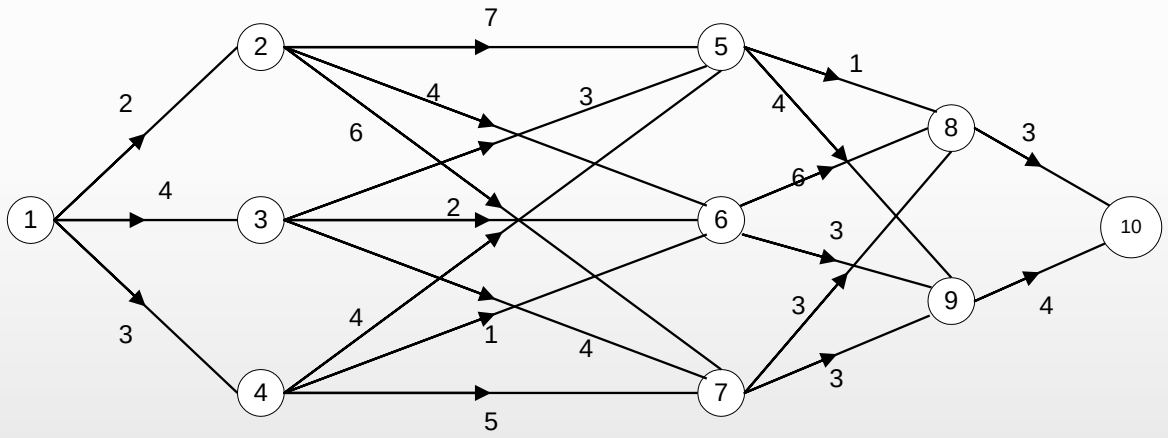
$$k = n, n-1, \ldots, 1$$

# Algoritma Program Dinamis

- Buat karakteristik struktur solusi optimal.
  - tahap, variable keputusan, status (state), dsb
- Definisikan secara rekursif nilai solusi optimal.
  - hubungan nilai optimal suatu tahap dengan tahap sebelumnya
- Hitung nilai solusi optimal secara maju atau mundur.
  - Menggunakan tabel
- Konstruksi solusi optimal (opsional).

# Contoh Program Dinamis Mundur

- Tentukan lintasan terpendek dari simpul 1 ke simpul 10



# Contoh Penyelesaian

- Misalkan $x_1$, $x_2$, …, $x_4$ adalah simpul-simpul yang dikunjungi pada tahap $k$ ($k$ = 1, 2, 3, 4).
- Maka rute yang dilalui adalah $1 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$ , yang dalam hal ini $x_4$ = 10
- *Tahap* (*k*) adalah proses memilih simpul tujuan berikutnya (ada 4 tahap).
- *Status* (*s*) yang berhubungan dengan masing-masing tahap adalah simpul-simpul di dalam graf.

# Contoh PD mundur

- Relasi rekurens berikut menyatakan lintasan terpendek dari status *s* ke $x_4$ pada tahap *k*:

$$f_4(s) = c_{sx_4}$$ (basis)

$$f_k(s) = \min_{x_k}\{c_{sx_k} + f_{k+1}(x_k)\}$$ (rekurens), $k = 1, 2, 3$

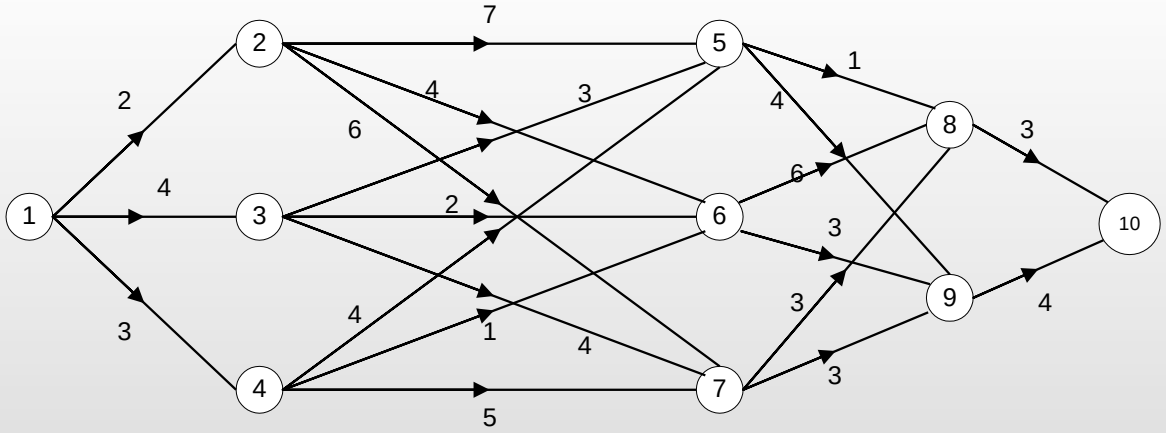
  - $x_k$ : peubah keputusan pada tahap $k$ ($k = 1, 2, 3$).
  - $c_{sx_k}$ : bobot (*cost*) sisi dari *s* ke $x_k$
  - $f_k(s, x_k)$ : total bobot lintasan dari *s* ke $x_k$
  - $f_k(s)$ : nilai minimum dari $f_k(s, x_k)$

- Tujuan program dinamis mundur:
  - mendapatkan $f_1(1)$ dengan cara mencari $f_4(s), f_3(s), f_2(s)$ terlebih dahulu.

# Tahap 4

$$f_4(s) = c_{sx_4}$$

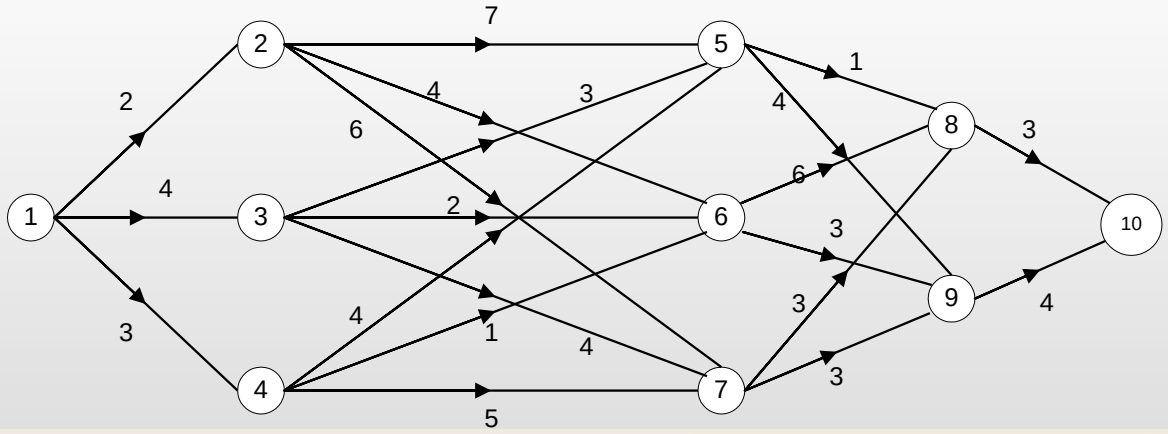
| $s$ | Solusi Optimum | |
|---|---|---|
| | $f_4(s)$ | $x_4^*$ |
| 8 | 3 | 10 |
| 9 | 4 | 10 |



# Tahap 3

$$f_3(s) = \min_{x_3}\{c_{sx_3} + f_4(x_3)\}$$

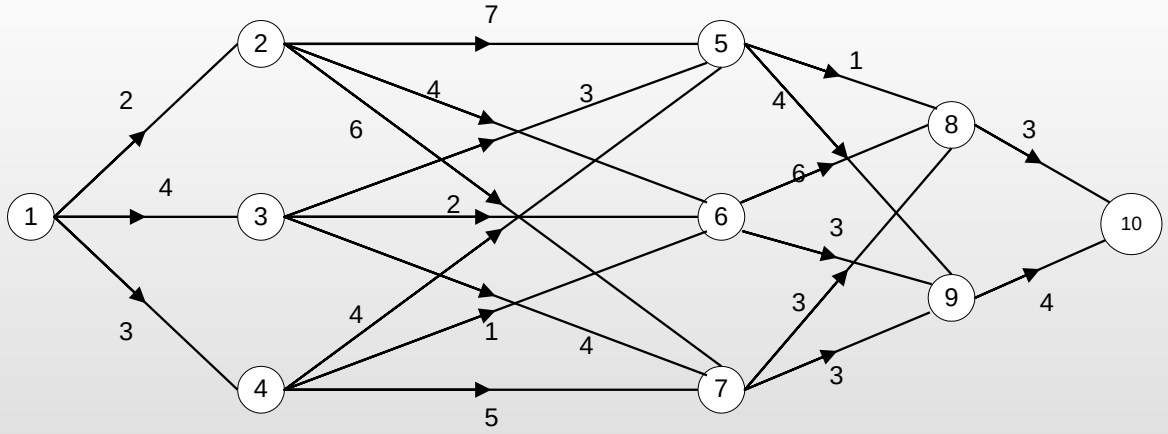
| $s$ \ $x_3$ | $f_3(s, x_3) = c_{s,x3} + f_4(x_3)$ | | Solusi Optimum | |
|---|---|---|---|---|
| | 8 | 9 | $f_3(s)$ | $x_3^*$ |
| 5 | 4 | 8 | 4 | 8 |
| 6 | 9 | 7 | 7 | 9 |
| 7 | 6 | 7 | 6 | 8 |



# Tahap 2

$$f_2(s) = \min_{x_2}\{c_{sx_2} + f_3(x_2)\}$$

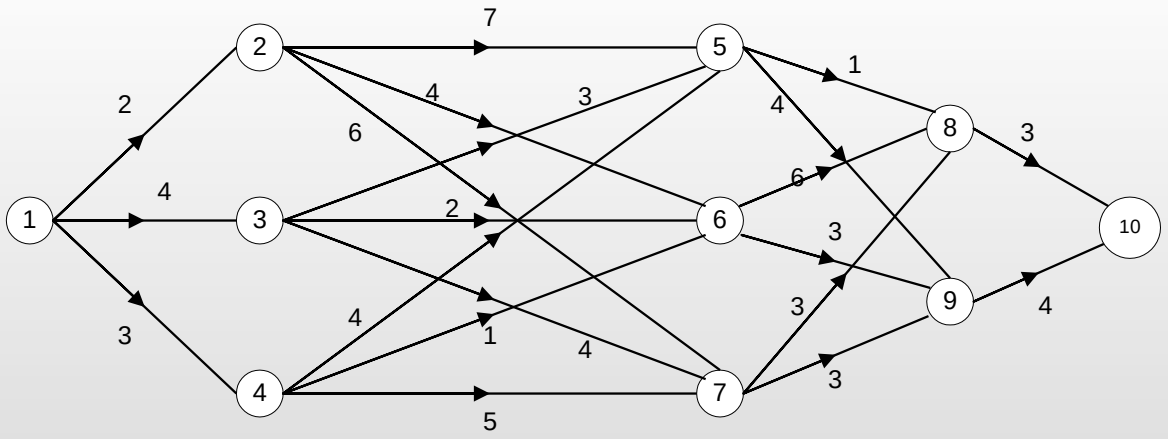
| $s$ \ $x_2$ | $f_2(s, x_2) = c_{s,x2} + f_3(x_2)$ | | | Solusi Optimum | |
|---|---|---|---|---|---|
| | 5 | 6 | 7 | $f_2(s)$ | $x_2^*$ |
| 2 | 11 | 11 | 12 | 11 | 5 atau 6 |
| 3 | 7 | 9 | 10 | 7 | 5 |
| 4 | 8 | 8 | 11 | 8 | 5 atau 6 |



# Tahap 1

$$f_1(s) = \min_{x_1}\{c_{sx_1} + f_2(x_1)\}$$

| $s$ \ $x_1$ | $f_1(s, x_1) = c_{s,x1} + f_2(x_1)$ | | | Solusi Optimum | |
|---|---|---|---|---|---|
| | 2 | 3 | 4 | $f_1(s)$ | $x_1^*$ |
| 1 | 13 | 11 | 11 | 11 | 3 atau 4 |



# Solusi Optimum yang diperoleh

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | Panjang Lintasan Terpendek |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 8 | 10 | 11 |
| | 4 | 5 | 8 | 10 | 11 |
| | | 6 | 9 | 10 | 11 |

Jadi ada tiga lintasan terpendek dari 1 ke 10, yaitu

$1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10$

$1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10$

$1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$

Dengan panjang lintasan 11

# Contoh Program Dinamis Maju : Knapsack 0/1

Carilah solusi persoalan Knapsack 0/1, jika diberikan n = 3, dengan M = 5 sbb:

| Barang ke-$i$ | $w_i$ | $p_i$ |
|---|---|---|
| 1 | 2 | 65 |
| 2 | 3 | 80 |
| 3 | 1 | 30 |

# Contoh Program Dinamis Maju : Knapsack 0/1

- Tahap (*k*) adalah proses memasukkan barang ke dalam Knapsack (ada 3 tahap untuk contoh kasus).
- Status (*y*) menyatakan kapasitas muat knapsack yang tersisa setelah memasukkan barang pada tahap sebelumnya.
- Dari tahap ke-1, kita masukkan objek ke-1 ke dalam knapsack untuk setiap satuan kapasitas knapsack sampai batas kapasitas maksimumnya.
- Karena kapasitas knapsack adalah bilangan bulat, maka pendekatan ini praktis.

# Cara kerja PD Maju

- Misalkan ketika memasukkan objek pada tahap *k*, kapasitas muat knapsack sekarang adalah $y - w_k$.
- Untuk mengisi kapasitas sisanya, kita menerapkan prinsip optimalitas dengan mengacu pada nilai optimum dari tahap sebelumnya untuk kapasitas sisa $y - w_k$
- Nilai optimum pada tahap sebelumnya adalah $f_{k-1}(y - w_k)$.

# Keputusan tahap ke-k

- Selanjutnya, kita bandingkan
  - nilai keuntungan dari objek pada tahap $k$ (yaitu $p_k$) + $f_{k-1}(y - w_k)$ dengan
  - keuntungan pengisian hanya $k - 1$ macam objek, $f_{k-1}(y)$.
- Jika $p_k + f_{k-1}(y - w_k) < f_{k-1}(y)$, maka objek yang ke-$k$ tidak dimasukkan ke dalam karung,
- Jika $p_k + f_{k-1}(y - w_k) > f_{k-1}(y)$, maka objek yang ke-$k$ dimasukkan.

# Relasi rekurens

- Relasi rekurens untuk persoalan ini adalah
  - $f_0(y) = 0, \quad y = 0, 1, 2, \ldots, M$ (basis)
  - $f_k(y) = -\infty, \quad y < 0$ (basis)
  - $f_k(y) = \max\{f_{k-1}(y), p_k + f_{k-1}(y - w_k)\}$, (rekurens)
    $k = 1, 2, \ldots, n$
- fk(y) adalah keuntungan optimum pada tahap k untuk kapasitas knapsack sebesar y.
- fo(y) = 0 adalah nilai persoalan knapsack kosong (tidak ada persoalan knapsack dengan kapasitas y)
- fk(y) = -∞, adalah nilai persoalan knapsack untuk kapasitas negatif.
- Solusi optimum dari persoalan knapsack adalah fn(M).

# Tahap 1

| Barang ke-$i$ | $w_i$ | $p_i$ |
|---|---|---|
| 1 | 2 | 65 |
| 2 | 3 | 80 |
| 3 | 1 | 30 |

- $f_1(y) = \max\{f_0(y), p_1 + f_0(y - w_1)\}$
- $= \max\{f_0(y), 65 + f_0(y - 2)\}$

| $y$ | | | Solusi Optimum | |
|---|---|---|---|---|
| | $f_0(y)$ | $65 + f_0(y-2)$ | $f_1(y)$ | $(x_1^*, x_2^*, x_3^*)$ |
| 0 | **0** | -∞ | 0 | (0, 0, 0) |
| 1 | **0** | -∞ | 0 | (0, 0, 0) |
| 2 | 0 | **65** | 65 | (1, 0, 0) |
| 3 | 0 | **65** | 65 | (1, 0, 0) |
| 4 | 0 | **65** | 65 | (1, 0, 0) |
| 5 | 0 | **65** | 65 | (1, 0, 0) |

# Tahap 2

| Barang ke-$i$ | $w_i$ | $p_i$ |
|---|---|---|
| 1 | 2 | 65 |
| 2 | 3 | 80 |
| 3 | 1 | 30 |

- $f_2(y) = \max\{f_1(y), p_2 + f_1(y - w_2)\}$
- $= \max\{f_1(y), 80 + f_1(y - 3)\}$

| $Y$ | | | Solusi Optimum | |
|---|---|---|---|---|
| | $f_1(y)$ | $80 + f_1(y-3)$ | $f_2(y)$ | $(x_1^*, x_2^*, x_3^*)$ |
| 0 | **0** | $80 + (-\infty) = -\infty$ | 0 | (0, 0, 0) |
| 1 | **0** | $80 + (-\infty) = -\infty$ | 0 | (0, 0, 0) |
| 2 | **65** | $80 + (-\infty) = -\infty$ | 65 | (1, 0, 0) |
| 3 | 65 | 80 + 0 = **80** | 80 | (0, 1, 0) |
| 4 | 65 | 80 + 0 = **80** | 80 | (0, 1, 0) |
| 5 | 65 | 80 + 65 = **145** | 145 | (1, 1, 0) |

# Tahap 3

| Barang ke-$i$ | $w_i$ | $p_i$ |
|---|---|---|
| 1 | 2 | 65 |
| 2 | 3 | 80 |
| 3 | 1 | 30 |

- $f_3(y) = \max\{f_2(y), p_3 + f_2(y - w_3)\}$
- $= \max\{f_2(y), 30 + f_2(y - 1)\}$

| $Y$ | $f_2(y)$ | $30 + f_2(y-1)$ | Solusi Optimum $f_3(y)$ | Solusi Optimum $(x_1^*, x_2^*, x_3^*)$ |
|---|---|---|---|---|
| 0 | **0** | $30 + (-\infty) = -\infty$ | 0 | (0, 0, 0) |
| 1 | **0** | 30 + 0 = 30 | 30 | (0, 0, 1) |
| 2 | **65** | 30 + 0 = 30 | 65 | (1, 0, 0) |
| 3 | 80 | 30 + 65 = **95** | 95 | (1, 0, 1) |
| 4 | 80 | 30 + 80 = **110** | 110 | (0, 1, 1) |
| 5 | **145** | 30 + 80 = 110 | 145 | (1, 1, 0) |

- Solusi optimum: X(1,1,0) dengan f= 145

# LATIHAN

- Tinjau persoalan 0/1 *Knapsack* dengan 6 objek:

  $w_1 = 3$; $p_1 = 10$

  $w_2 = 5$; $p_2 = 5$

  $w_3 = 4$; $p_3 = 6$

  $w_4 = 2$; $p_4 = 4$

  $w_5 = 1$; $p_5 = 5$

  $w_6 = 1$; $p_6 = 3$

  Kapasitas *knapsack* $W = 7$

- Temukan solusinya dgn Program Dinamis

# Referensi


- Anany Levitin, Introduction to The Design and Analysis of Algorithms, Pearson, 2012.
- S.Dasgupta, et al. Algorithms, 2006.
- Cormen, et al. Algorithms, MGH,2009.
- Strategi Algoritma – Rinaldi Munir (ITB) :
  - https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/stmik.htm
- Khan Academy, Computer Science: Algorithm, https://www.khanacademy.org/computing/computer-science/algorithms