Pengantar Strategi Algoritmik

Oleh: Rinaldi Munir

Masalah (*Problem*)

- **Masalah** atau **persoalan:** pertanyaan atau tugas yang kita cari jawabannya.

- Contoh-contoh masalah:

1. [**Masalah pengurutan**] Diberikan senarai (*list*) *S* yang terdiri dari *n* buah data bilangan bulat. Bagaimana mengurutkan *n* buah data tersebut sehingga terurut secara menaik?

   Jawaban dari masalah ini: barisan nilai di dalam senarai yang terurut menaik.

2. [**Masalah pencarian**] Tentukan apakah suatu bilangan *x* terdapat di dalam sebuah senarai *S* yang beriri *n* buah bilangan bulat!

Jawaban dari masalah ini: “ya” jika x ditemukan di dalam senarai, atau “tidak” jika *x* tidak terdapat di dalam senarai.

- **Instansiasi masalah**: parameter nilai yang diasosiasikan pada masalah.

- Jawaban terhadap instansiasi masalah disebut **solusi**

- Contoh: Selesaikan masalah pengurutan untuk

  S = [15, 4, 8, 11, 2, 10, 19] n = 7

  Solusi: S = [2, 4, 8, 10, 11, 15, 19].

Algoritma

- Untuk masalah dengan instansiasi yang besar, solusinya menjadi lebih sulit.

- Perlu sebuah prosedur umum yang berisi langkah-langkah penyelesaian masalah → algoritma

- **Algoritma**:urutan langkah-langkah untuk memecahkan suatu masalah

- Definisi lain:

  Algoritma adalah deretan langkah-langkah komputasi yang mentransformasikan data masukan menjadi keluaran [COR92].

  Algoritma adalah deretan instruksi yang jelas untuk memecahkan masalah, yaitu untuk memperoleh keluaran yang diinginkan dari suatu masukan dalam jumlah waktu yang terbatas. [LEV03].

  Algoritma adalah prosedur komputasi yang terdefinisi dengan baik yang menggunakan beberapa nilai sebagai masukan dan menghasilkan beberapa nilai yang disebut keluaran. Jadi, algoritma adalah deretan langkah komputasi yang mentransformasikan masukan menjadi keluran [COR89].

- Algoritmik: Studi mengenai algoritma.

- Notasi apapun dapat digunakan untuk menuliskan algoritma asalkan mudah dibaca dan dipahami.

- Algoritma dapat ditulis dengan notasi:
  1. Bagan alir (*flow chart*)
  2. Kalimat-kalimat deskriptif
  3. *Pseudo-code* (gabungan antara bahasa alami dengan bahasa pemrograman)

Analisis Algoritma

- Sebuah algoritma tidak hanya harus benar, tetapi juga harus mangkus (*efficient*)
- Ukuran kemangkusan algoritma: waktu dan ruang memori (*space*).
- Algoritma yang mangkus: algoritma yang meminimumkan kebutuhan waktu dan ruang

- Alat ukur kemangkusan algoritma:
  1. Kompleksitas waktu, $T(n)$
  2. Kompleksitas ruang, $S(n)$

- n = ukuran masukan yang diproses oleh algoritma

- $T(n)$: jumlah operasi yang dilakukan untuk menjalankan sebuah algoritma sebagai fungsi dari ukuran masukan n.

- $S(n)$: ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n

- Operasi yang dihitung hanyalah operasi dasar (*basic operation*) saja

- Operasi dasar: operasi khas yang mendasari suatu algoritma

- Misalnya:
  - operasi perbandingan elemen pada algoritma pengurutan/pencarian
  - operasi penjumlahan dan perkalian pada algoritma perkalian matriks

- Kompleksitas waktu asimptotik:

  - perkiraan kasar kebutuhan waktu algoritma dengan meningkatnya nilai *n*
  - menyatakan laju pertumbuhan waktu, bukan menyatakan jumlah operasi dasar sesungguhnya.

- Tiga cara menyatakan waktu asimptotik:

  1. $O(f(n))$: untuk batas atas laju kebutuhan waktu
  2. $\Omega(g(n))$: untuk batas bawah laju kebutuhan waktu
  3. $Q(h(n))$: jika $f(n) = g(n)$

Strategi algoritmik

- Nama mata kuliah di IF ITB saja

- Di luar negeri dikenal dengan nama: **Teknik Perancangan Algoritma** (*Algorithm Design Techniques*)

- **Strategi algoritmik**: pendekatan umum untuk memecahkan masalah secara algoritmis yang dapat diterapkan pada beraneka ragam masalah guna mencapai tujuan

- Strategi algoritmik bertujuan mencari algoritma yang mangkus untuk memecahkan masalah

- Ukuran kemangkusan algoritma dinyatakan dengan notasi *O*, *W*, atau *Q*.

- Dua alasan mempelajari strategi algoritmik:
  1. memberikan panduan untuk merancang algoritma bagi masalah baru.

  2. mengklasifikasikan algoritma berdasarkan gagasan perancangan yang mendasarinya.

Klasifikasi Strategi Algoritmik:

1. Strategi solusi langsung (*direct solution strategies*)
   - Algoritma *Brute Force*
   - Algoritma *Greedy*

2. Strategi berbasis pencarian pada ruang status (*state-space base strategies*)
   - Algoritma *Backtracking*
   - Algoritma *Branch and Bound*

3. Strategi solusi atas-bawah (*top-down solution strategies*)

   - Algoritma *Divide and Conquer*.

4. Strategi solusi bawah-atas (*bottom-up solution strategies*)

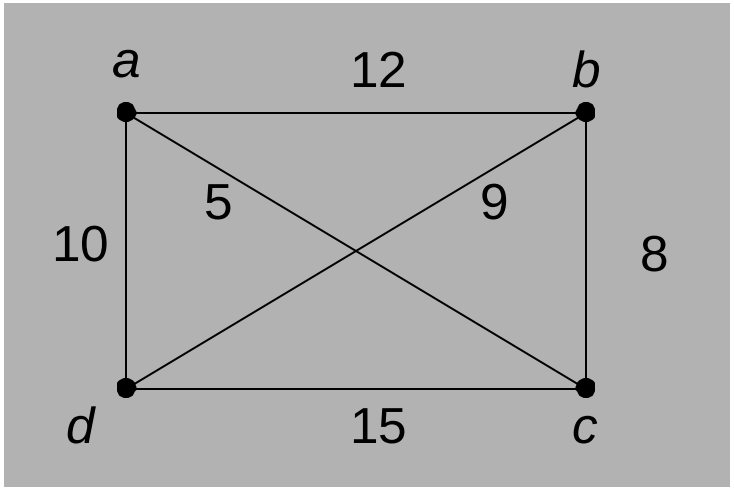
   - *Dynamic Programming*.

- **Catatan**: klasifikasi ini tidak kaku, bisa berbeda bergantung pendekatan ayng digunakan.

Beberapa Masalah Klasik

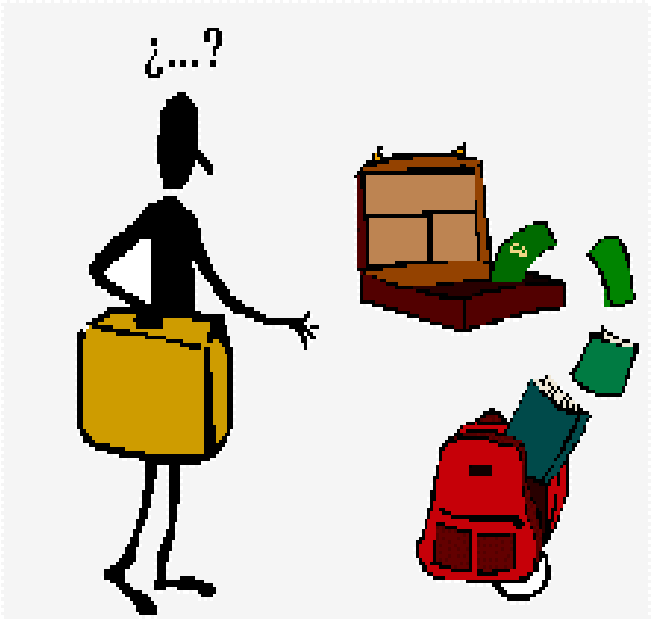
1. *Travelling Salesperson Problem* (*TSP*)

Persoalan: Diberikan *n* buah kota serta diketahui jarak antara setiap kota satu sama lain. Temukan perjalanan (*tour*) terpendek yang dimulai dari sebuah kota dan melalui setiap kota lainnya hanya sekali dan kembali lagi ke kota asal keberangkatan.



2. *Integer Knapsack (1/0 Knapsack)*

Persoalan: Diberikan *n* buah objek dan sebuah *knapsack* (karung, tas, buntilan, dsb) dengan kapasitas bobot *K*. Setiap objek memiliki properti bobot (*weigth*) w_i dan keuntungan (*profit*) p_i. Bagaimana memilih objek-objek yang dimasukkan ke dalam *knapsack* sehingga tidak melebihi kapasitas *knapsack* namun memaksimumkan total keuntungan yang diperoleh.

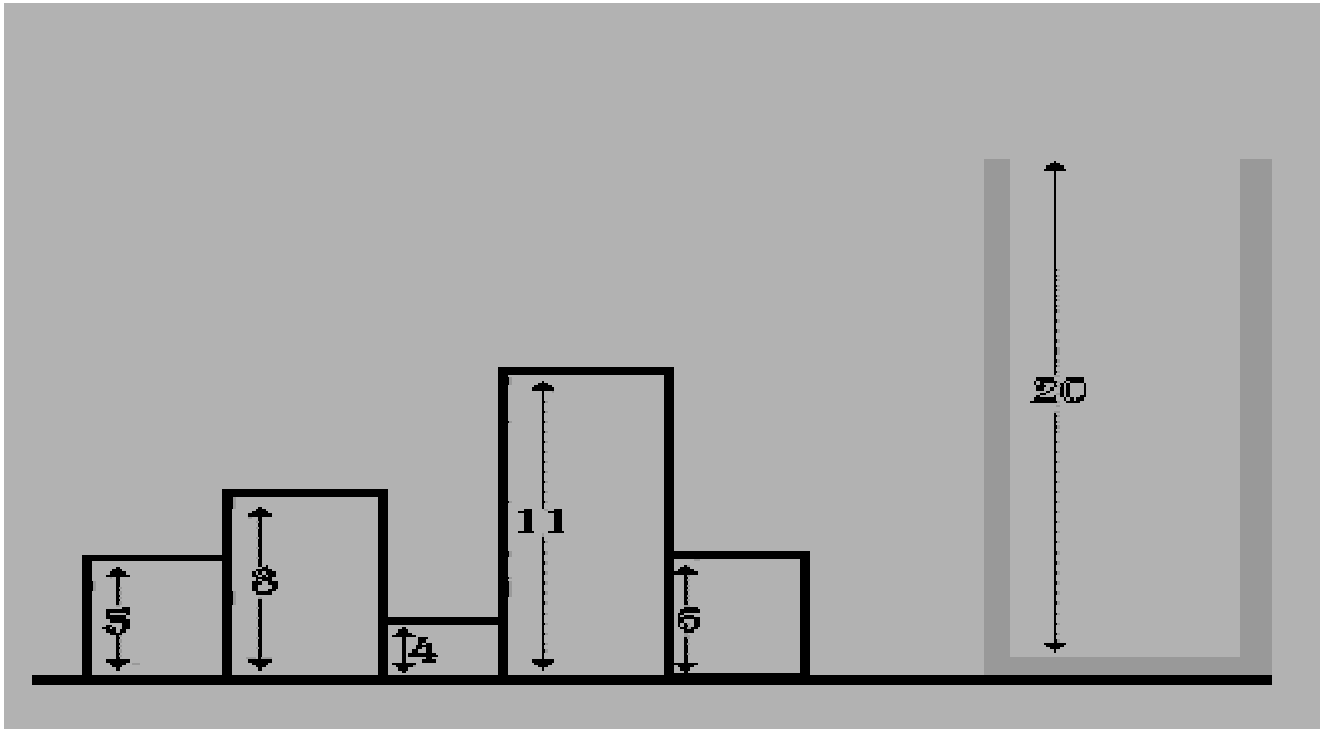


Contoh instansiasi masalah:

$n = 5$; $K = 20$

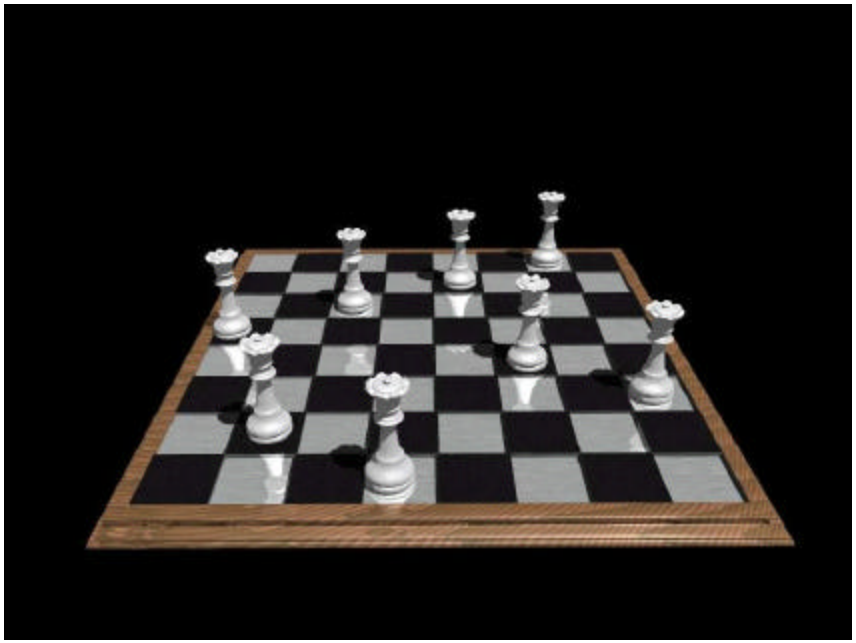
$w_1 = 5$; $p_1 = 20$; $w_2 = 8$; $p_2 = 30$

$w_3 = 4$; $p3 = 50$; $w_4 = 6$; $p_4 = 10$



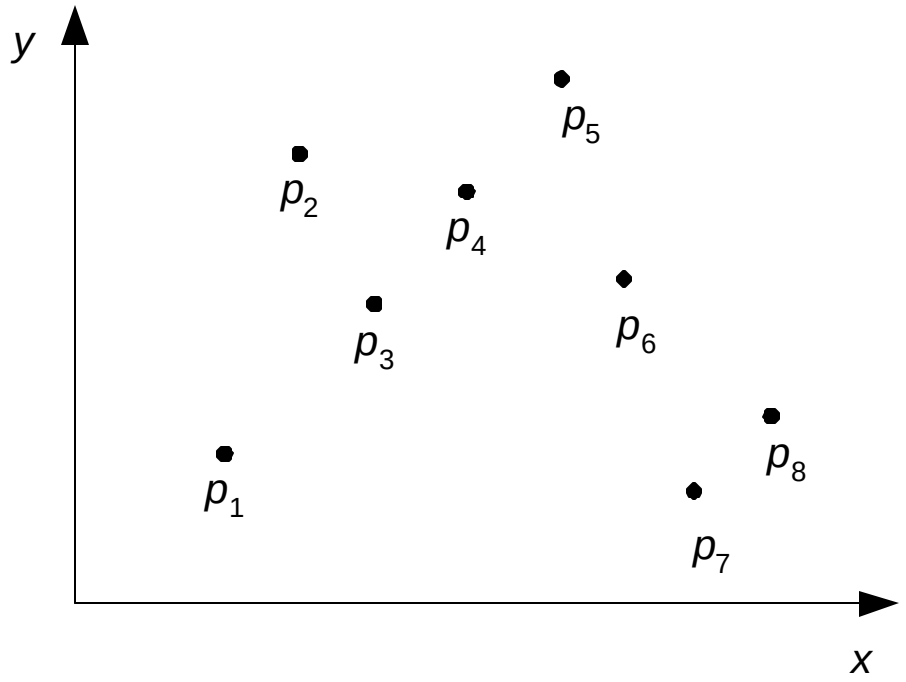
3. Persoalan N-Ratu *(The N-Queens Problem)*

Persoalan: Diberikan sebuah papan catur yang berukuran $N \times N$ dan delapan buah ratu. Bagaimana menempatkan N buah ratu (Q) itu pada petak-petak papan catur sedemikian sehingga tidak ada dua ratu atau lebih yang terletak pada satu baris yang sama, atau pada satu kolom yang sama, atau pada satu diagonal yang sama ?



4. Mencari Pasangan Titik yang Jaraknya Terdekat *(Closest Pair)*

Persoalan: Diberikan n buah titik, tentukan dua buah titik yang terdekat satu sama lain.



5. Permainan *15-Puzzle*

Persoalan: Diberikan sebuah 15-*puzzle* yang memuat 15 buah ubin (*tile*) yang diberi nomor 1 sampai 15, dan satu buah slot kosong yang digunakan untuk menggerakkan ubin ke atas, ke bawah, ke kiri, dan ke kanan. Misalkan diberikan keadaan awal dan keadaaan akhir susunan ubin. Kita ingin menransformasikan susunan awal menjadi susunan akhir.

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

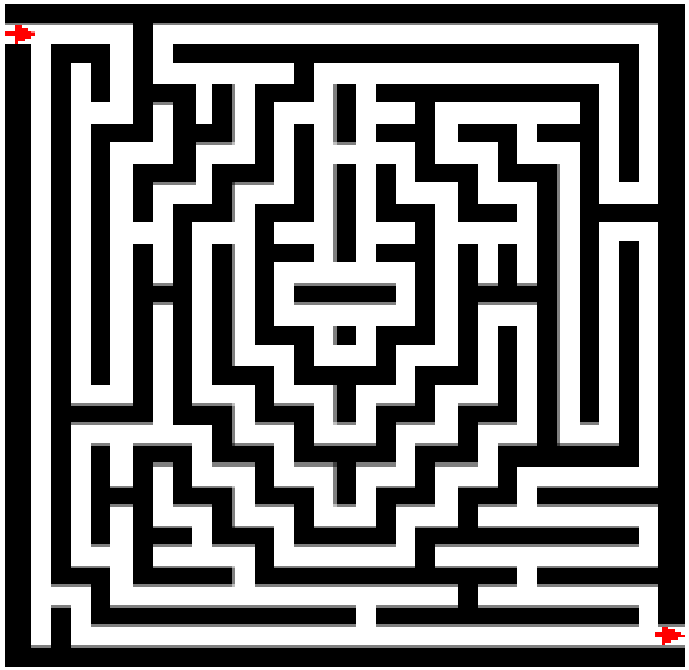
(a) Susunan awal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b) Susunan akhir

6. Menemukan jalan keluar dari labirin (Maze Problem)

Persoalan: Diberikan sebuah labirin dengan satu atau lebih pintu masuk dan satu atau lebih pintu keluar. Temukan jalan yang harus dilalui sehingga seseorang dapat keluar dengan selamat dari labirin tersebut (tidak tersesat di dalamnya).



7. Pewarnaan Graf *(Graph Colouring)*

Persoalan: Diberikan sebuah graf *G* dengan *n* buah simpul dan disediakan m buah warna. Warnailah seluruh simpul graf *G* sedemikian sehin24gga tidak ada dua buah simpul bertetangga yang mempunyai warna sama (Perhatikan juga bahwa tidak seluruh warna harus dipakai)