PROGRAM STUDI TEKNIK INFORMATIKA
Sekolah Teknik Elrektro dan Informatika

INSTITUT TEKNOLOGI BANDUNG

Pengantar Strategi Algoritma

Bahan Kuliah IF2211 Strategi Algoritma

RINALDI MUNIR

Lab Ilmu dan Rekayasa Komputasi
Kelompok Keahlian Informatika

Institut Teknologi Bandung

Kampus ITB yang indah…

Foto oleh Eko Purwono (AR ITB)

Inilah STEI-ITB…

LabTek V, di sini Informatika ITB berada

Foto oleh Budi Rahardjo (EL ITB)

Salah satu mata kuliahnya….

IF2211 Strategi Algoritma

Apakah Strategi Algoritma itu?

Strategi algoritma (*algorithm strategies*) adalah:

- pendekatan umum
- untuk memecahkan persoalan secara algoritmis
- sehingga dapat diterapkan pada bermacam-macam persoalan
- dari berbagai bidang komputasi [Levitin, 2003]

- Nama lain: *algorithm design technique*

Persoalan (*Problem*)

- **Persoalan:** pertanyaan atau tugas (*task*) yang ingin kita cari jawabannya.

- Contoh-contoh persoalan:

1. [**Persoalan pengurutan**] Diberikan senarai (*list*) *S* yang terdiri dari *n* buah *integer*. Urutkan *n* buah *integer* tersebut sehingga terurut secara menaik!

   Jawaban: barisan nilai di dalam senarai yang terurut menaik.

2. [**Persoalan pencarian**] Apakah terdapat sebuah elemen bernilai *x* di dalam sebuah senarai *S* yang berisi *n* buah bilangan bulat?

Jawaban: "ya" jika x ditemukan di dalam senarai, atau "tidak" jika x tidak terdapat di dalam senarai.

- **Instansiasi persoalan**: parameter nilai yang diasosiasikan di dalam persoalan.

- Jawaban terhadap instansiasi persoalan disebut **solusi**

- Contoh: Selesaikan persoalan pengurutan untuk

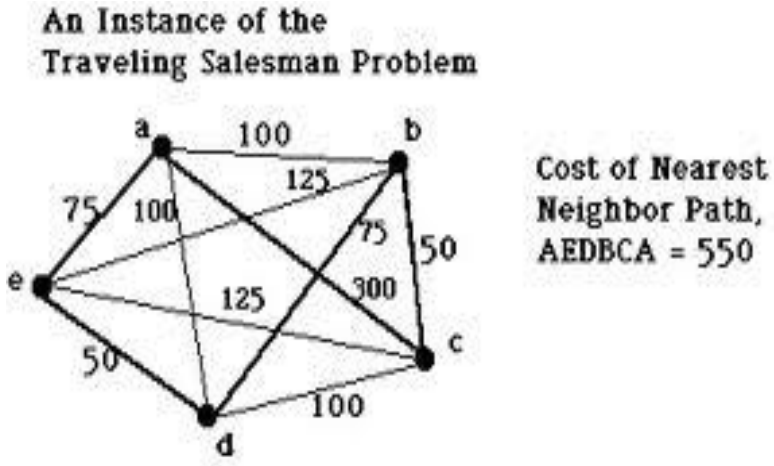
  S = [15, 4, 8, 11, 2, 10, 19] n = 7

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

Beberapa Contoh Persoalan Klasik

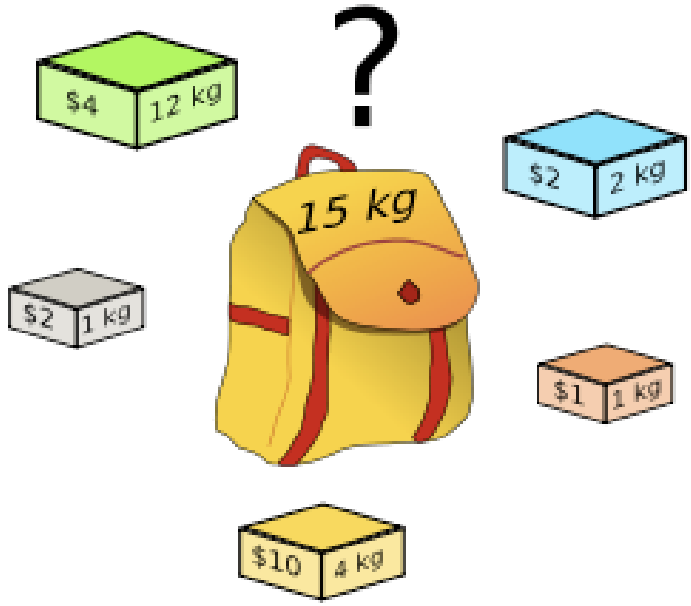
1. *Travelling Salesperson Problem* (*TSP*)

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 Problem*

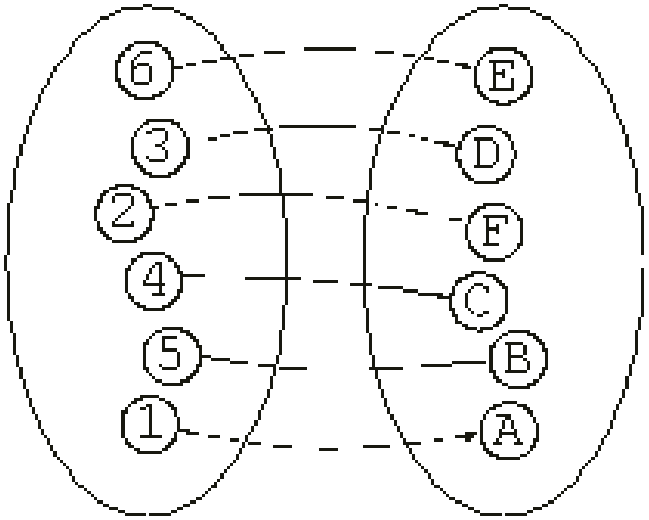
Persoalan: Diberikan *n* buah objek dan sebuah *knapsack* (karung, tas, ransel, dsb) dengan kapasitas bobot *K*. Setiap objek memiliki properti bobot (*weigth*) w_i dan keuntungan (*profit*) p_i.



Bagaimana cara memilih objek-objek yang dimasukkan ke dalam *knapsack* sedemikian sehingga tidak melebihi kapasitas *knapsack* namun memberikan keuntungan maksimal?

3. Persoalan penugasan (*assignment problem*)

Misalkan terdapat *n* orang dan *n* buah pekerjaan (*job*). Setiap orang akan di-*assign* dengan sebuah pekerjaan. Penugasan orang ke-*i* dengan pekerjaan ke-*j* membutuhkan biaya sebesar *c*(*i*, *j*). Bagaimana melakukan penugasan sehingga total biaya penugasan adalah seminimal mungkin?

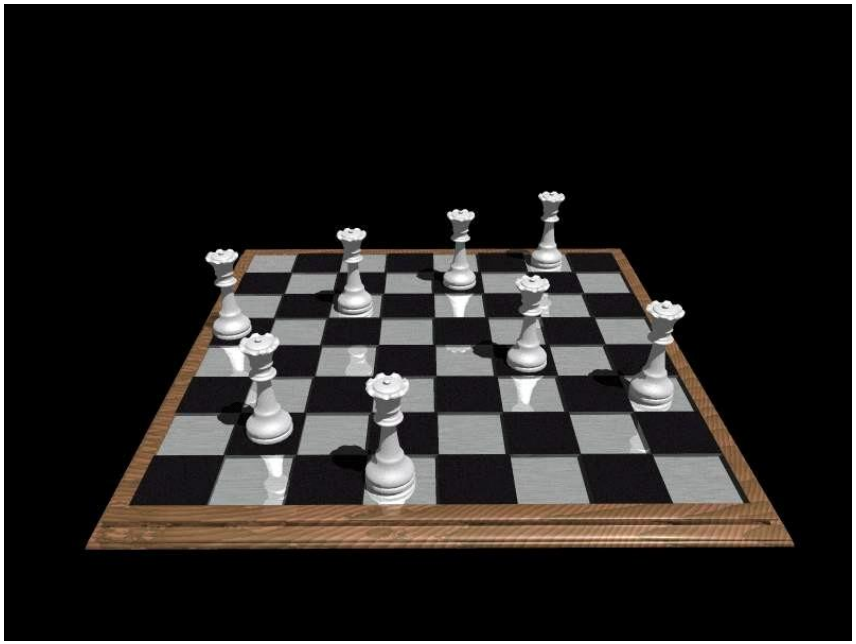


Contoh instansiasi persoalan:

$$C = \begin{bmatrix} Job\,1 & Job\,2 & Job\,3 & Job\,4 & \\ 9 & 2 & 7 & 8 & \text{Orang } a \\ 6 & 4 & 3 & 7 & \text{Orang } b \\ 5 & 8 & 1 & 4 & \text{Orang } c \\ 7 & 6 & 9 & 4 & \text{Orang } d \end{bmatrix}$$

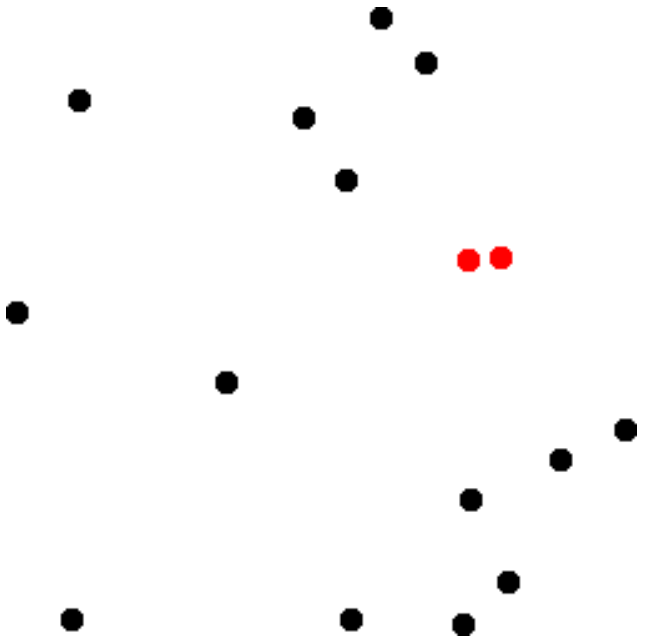
4. Persoalan N-Ratu *(The N-Queens Problem)*

Persoalan: Diberikan sebuah papan catur yang berukuran $N \times N$ dan N buah bidak 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 ?



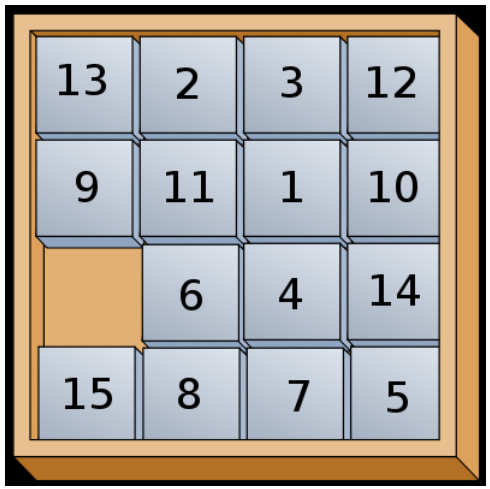
5. Mencari Pasangan Titik Terdekat *(Closest Pair)*

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



6. 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. Bagaimana langkah-langkah untuk mentransformasikan susunan awal menjadi susunan akhir?

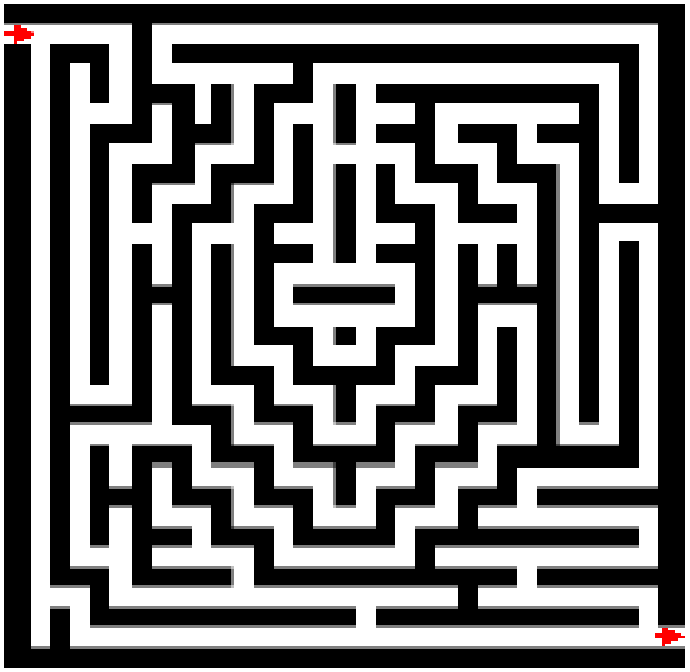


(a) Susunan awal

(b) Susunan akhir

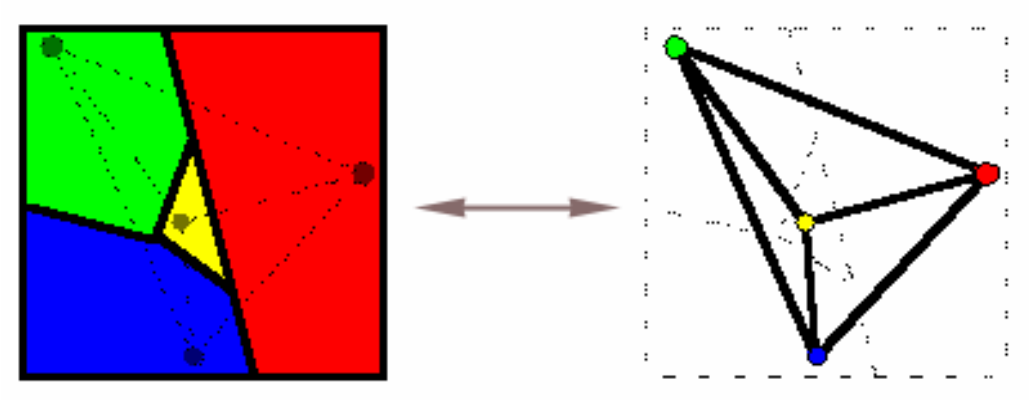
7. Menemukan jalan keluar dari labirin (Maze Problem)

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).



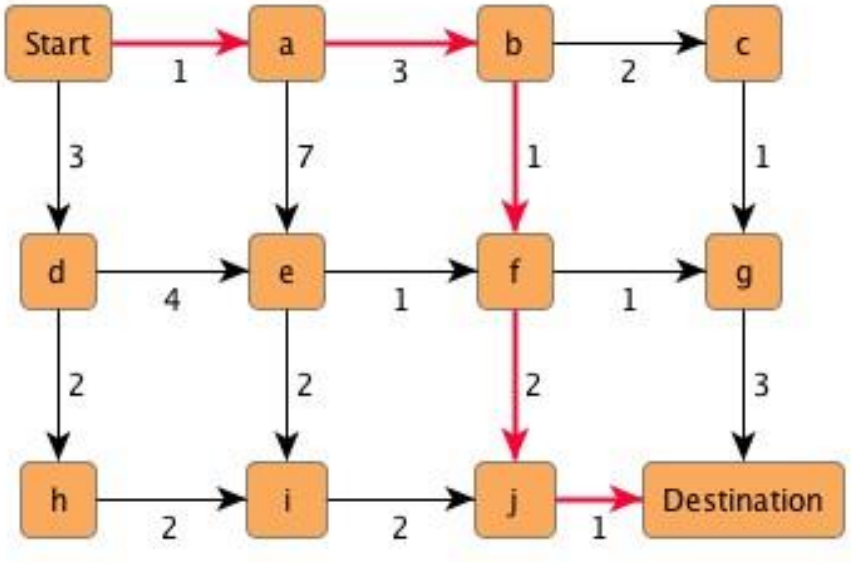
8. Pewarnaan Graf *(Graph Colouring)*

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



9. Lintasan terpendek (*shortest path*)

Persoalan: Diketahui *n* buah kota dan diberikan jarak antara dua buah kota yang bertetangga. Tentukan lintasan terpendek dari sebuah kota asal ke sebuah kota tujuan.



10. *Partition Problem*

Persoalan: Diberikan *n* buah bilangan bulat sembarang (mungkin ada elemen berulang). Bagaimana cara membaginya (mem-partisi) menjadi beberapa buah upa-himpunan (*subset*) sehingga jumlah seluruh nilai di dalam setiap upa-himpunan sama?

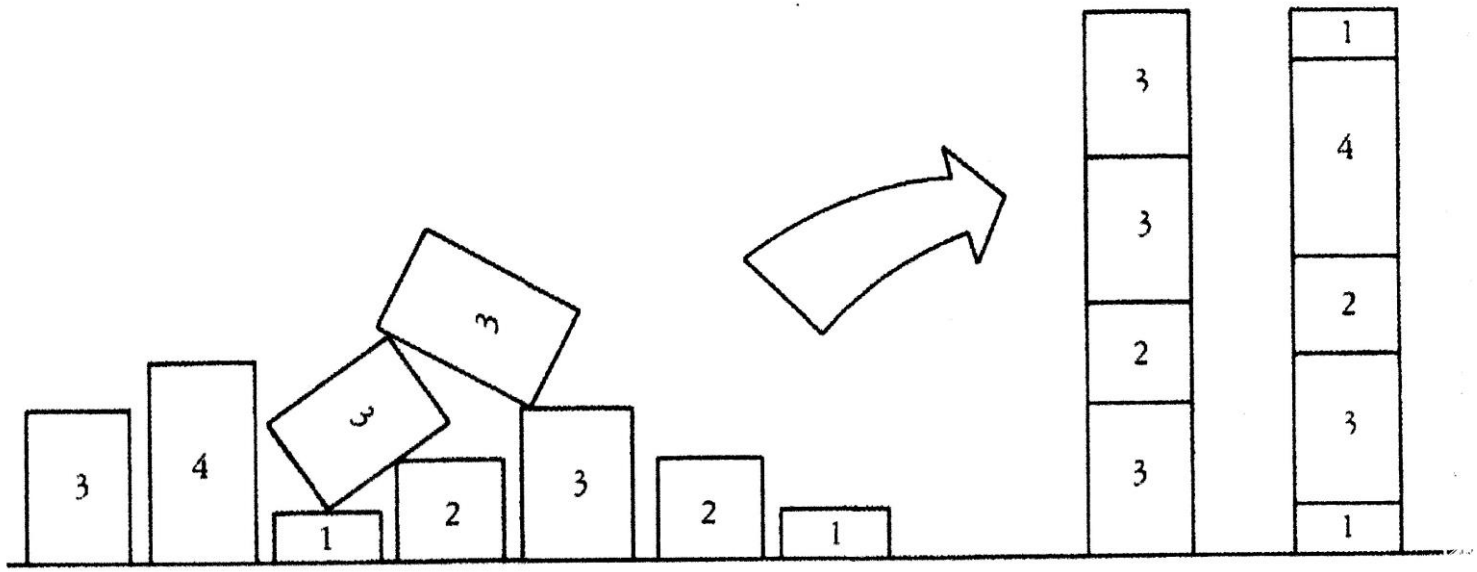


Figure 30.1 An example of the partition problem

Algoritma

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

- Perlu sebuah prosedur umum yang berisi langkah-langkah penyelesaian persoalan → **algoritma**

- **Algoritma**: urutan langkah-langkah untuk memecahkan suatu persoalan, dengan memproses masukan menjadi luaran.

Analisis Algoritma

- Tujuan analis algoritma: mengukur kinerja (*performance*) algoritma dari segi kemangkusannya (*efficient*)

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

  n = ukuran masukan yang diproses oleh algoritma

- $T(n)$: jumlah tahapan komputasi yang dilakukan di dalam algoritma sebagai fungsi dari ukuran masukan n.

  Tahapan komputasi: operasi-operasi yang terdapat di dalam algoritma (operasi perbandingan, operasi aritmetika, pengaksesan larik, dll)

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

- Tiga notasi asimptotik kebutuhan waktu algoritma:

  1. $O(f(n))$: batas lebih atas kebutuhan waktu algoritma
  2. $\Omega(g(n))$: batas lebih bawah kebutuhan waktu algoritma
  3. $\Theta(h(n))$: jika dan hanya jika $O(h(n)) = \Omega(h(n))$

Strategi Algoritma

1. Algoritma *Brute-Force*
2. Algoritma *Greedy*
3. Algoritma *Divide and Conquer*
4. Algoritma *Decrease and Conquer*
5. Algoritma *Bactracking*
6. Algoritma *Branch and Bound*
7. *Dynamic programming*

Mengapa Perlu Mempelajari Strategi Algoritma?

- Ada dua alasan (Levitin, 2003):
  1. Memberikan panduan (*guidance*) untuk merancang algoritma bagi persoalan baru.

  2. Dapat mengklasifikasikan algoritma berdasarkan gagasan perancangan yang mendasarinya.

Klasifikasi Strategi Algoritma:

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*.

   - Algoritma *Decrease and Conquer*

   - *Dynamic Programming*.

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

   - *Dynamic Programming*.

Pokok Bahasan Kuliah

1. Algoritma *brute force*
2. Algoritma *greedy*
3. Algoritma *divide and conquer*
4. Algoritma *decrease and conquer*
5. *DFS* dan *BFS*
6. Algoritma *backtracking*
7. Algoritma *branch and bound*
8. Algoritma *A**, *Best First Search*, dan *UCS*
9. Program dinamis (*dynamic programming*)
10. *String matching + regular expression (regex)*
11. Teori P, NP, dan NP-*Completes*

Buku Referensi Kuliah

1. Anany Levitin, *Introduction to the Design & Analysis of Algorithms*, Addison-Wesley, 2003.

2. Bhardwaj, Anuj; Verma, Parag, *Design and Analysis of Algorithm*, Alpha Science International Ltd., 2017, can be accessed at: http://portal.igpublish.com/iglibrary/obj/APSB0000252?dtbs=&searchid=1573133164973plapRvm7MQJMAwkcwThUs from https://lib.itb.ac.id/

3. Khan Academy, *Computer Science: Algorithm*, can be accessed at https://www.khanacademy.org/computing/computer-science/algorithms

4. Coursera, *Data Structures and Algorithms Specialization*, 2019, can be accessed at: https://www.coursera.org/learn/algorithmic-toolbox?specialization=data-structures-algorithms

5. Stuart J Russell & Peter Norvig, *Resources of topics in Artificial Intelligence: A Modern Approach*, 3rd Edition, Global Edition Paperback, Pearson, 2016, http://aima.cs.berkeley.edu/

6. Brandons Kerritt, *All You Need to Know About Big O Notation* [Python Examples], 2019, can be accessed at https://dev.to/brandonskerritt/all-you-need-to-know-about-big-o-notation-python-examples-2k4o

7. Coursera, *Machine Learning Clustering and Retrieval: Complexity of Brute Force Algorithm*, 2019, can be accessed at https://www.coursera.org/lecture/ml-clustering-and-retrieval/complexity-of-brute-force-search-5R6q3

8. Rinaldi Munir, Diktat kuliah IF2251 Strategi Algoritmik, Teknik Informatika ITB

9. Marin Vlastelica Pogančić, The Branch and Bound Algorithm, 2019, can be accessed at https://towardsdatascience.com/the-branch-and-bound-algorithm-a7ae4d227a69

10. Jurafsky and James H. Martin, *Speech and Language Processing*, 2019, can be accessed at https://web.stanford.edu/~jurafsky/slp3/

11. Chua Hock-Chuan, Regular Expressions (Regex), 2018, can be accessed at https://www.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html

12. Ellis Horrowitz & Sartaj Sahni, *Computer Algorithms*, Computer Science Press, 1998.

13. Richard E. Neapolitan, *Foundations of Algorithms*, D.C. Heath and Company, 1996

14. Thomas H. Cormen, Introduction to Algorithms, The MIT Press, 1992.

SELAMAT BELAJAR