

Buku Rancangan Pengajaran

DISAIN DAN ANALISIS ALGORITMA Kombinasi Interaktif Tatap Muka dan *e-Learning*



R. Yugo Kartono Isal



Fakultas Ilmu Komputer
Universitas Indonesia
2008

INFORMASI UMUM

| | | |
|------------------------------|---|--|
| Nama mata ajar | : | Disain dan Analisis Algoritma |
| Kode mata ajar | : | IKI 30100 |
| Diberikan pada | : | Tahun ketiga |
| Jumlah sks | : | 3 |
| Jenis sks | : | 3 x 50 menit pembelajaran mandiri/kelompok 1 x 50 menit latihan mandiri 2 x 50 menit diskusi di forum |
| Prasyarat | : | <ul style="list-style-type: none">- Matematika Diskret 2- Kalkulus- Struktur Data dan Algoritma |
| Kaitan dengan mata ajar lain | : | <ul style="list-style-type: none">- Sistem Cerdas- Teori Bahasa dan Automata |
| Dosen | : | R. Yugo Kartono Isal <yugo@cs.ui.ac.id> |
| Deskripsi Umum | : | Kuliah ini membahas beberapa metode perancangan algoritma untuk menyelesaikan masalah-masalah seperti metode <i>iterative</i> , <i>divide and conquer</i> , <i>dynamic programming</i> , <i>greedy</i> , <i>backtracking</i> , <i>branch and bound</i> . Pembahasan setiap algoritma disertai dengan dua aspek yang terpenting yaitu <i>correctness</i> dan <i>complexity</i> . Topik-topik yang akan dibahas meliputi: Mesin Turing; struktur data dasar: <i>linked list</i> ; Notasi <i>big Oh</i> , <i>big Theta</i> , <i>big Omega</i> , <i>recursion</i> , <i>Master Theorem</i> ; Algoritma-algoritma sorting: <i>InsertionSort</i> , <i>SelectionSort</i> , <i>Bose-NelsonSort</i> , <i>HeapSort</i> , <i>QuickSort</i> , <i>MergeSort</i> , <i>RadixSort</i> , <i>CountingSort</i> , <i>order statistik</i> ; <i>Dynamic programming</i> : <i>Matrix chain multiplication</i> , <i>LCS</i> ; Algoritma-algoritma <i>Greedy</i> : <i>Knapsack problem</i> , <i>TSP</i> , <i>Huffman codes</i> ; Algoritma-algoritma pada graphs: <i>BFS</i> , <i>DFS</i> , <i>connectivity</i> , <i>MST</i> , <i>shortest path</i> , <i>topological sort</i> , <i>maximum flow</i> ; Operasi-operasi pada matriks; <i>NP-completeness</i> ; <i>approximation algorithms</i> . |

Proses belajar

Pemelajaran kuliah ini merupakan kombinasi dari tiga metode:

- **Interaktif tatap muka**

Interaktif tatap muka akan dilakukan melalui kombinasi kuliah interaktif dan pemelajaran aktif melalui lembar kerja dan pemecahan problem. Diskusi lembar kerja dan pemecahan problem akan dibantu oleh tutor.

- ***E-Learning – self study* (Belajar mandiri)**

Pada sesi *e-Learning*, mahasiswa mempelajari modul yang disediakan, dan mengerjakan soal latihan yang diberikan pada modul.

- **Diskusi online**

Pada sesi diskusi, mahasiswa diharapkan menyampaikan pemahaman yang dimilikinya untuk didiskusikan dan dikonfirmasi dengan pemahaman mahasiswa lainnya. Mahasiswa juga membagi pengalaman belajarnya khususnya pada bagian materi-materi pilihan yang sesuai dengan minat.

DESKRIPSI UMUM PEMBELAJARAN

Mata ajar ini memberikan kemampuan dalam merancang dan menganalisa rancangan algoritma untuk menjawab permasalahan pemrograman dengan memperhatikan dua aspek utama yaitu: *Correctness* dan *Complexity* (algoritma yang dihasilkan benar dan efisien).

Sasaran pembelajaran

Tujuan umum tersebut dapat dijabarkan dalam sasaran pembelajaran utama dan penunjang berikut ini.

Sasaran pembelajaran utama

1. Mahasiswa mampu membuat rancangan algoritma untuk permasalahan pemrograman berdasarkan beberapa strategi rancangan seperti: *iterative, recursion, divide and conquer, dynamic programming, greedy, backtracking, branch and bound*.
2. Mahasiswa mampu membuktikan kebenaran algoritma iteratif.
3. Mahasiswa mampu menganalisa kompleksitas suatu algoritma dan merepresentasikannya menggunakan notasi-notasi standar.
4. Mahasiswa mampu memahami batasan kompleksitas dalam model komputasi komputer dan mampu memetakan permasalahan-permasalahan dalam kelompok batasan-batasan tersebut.

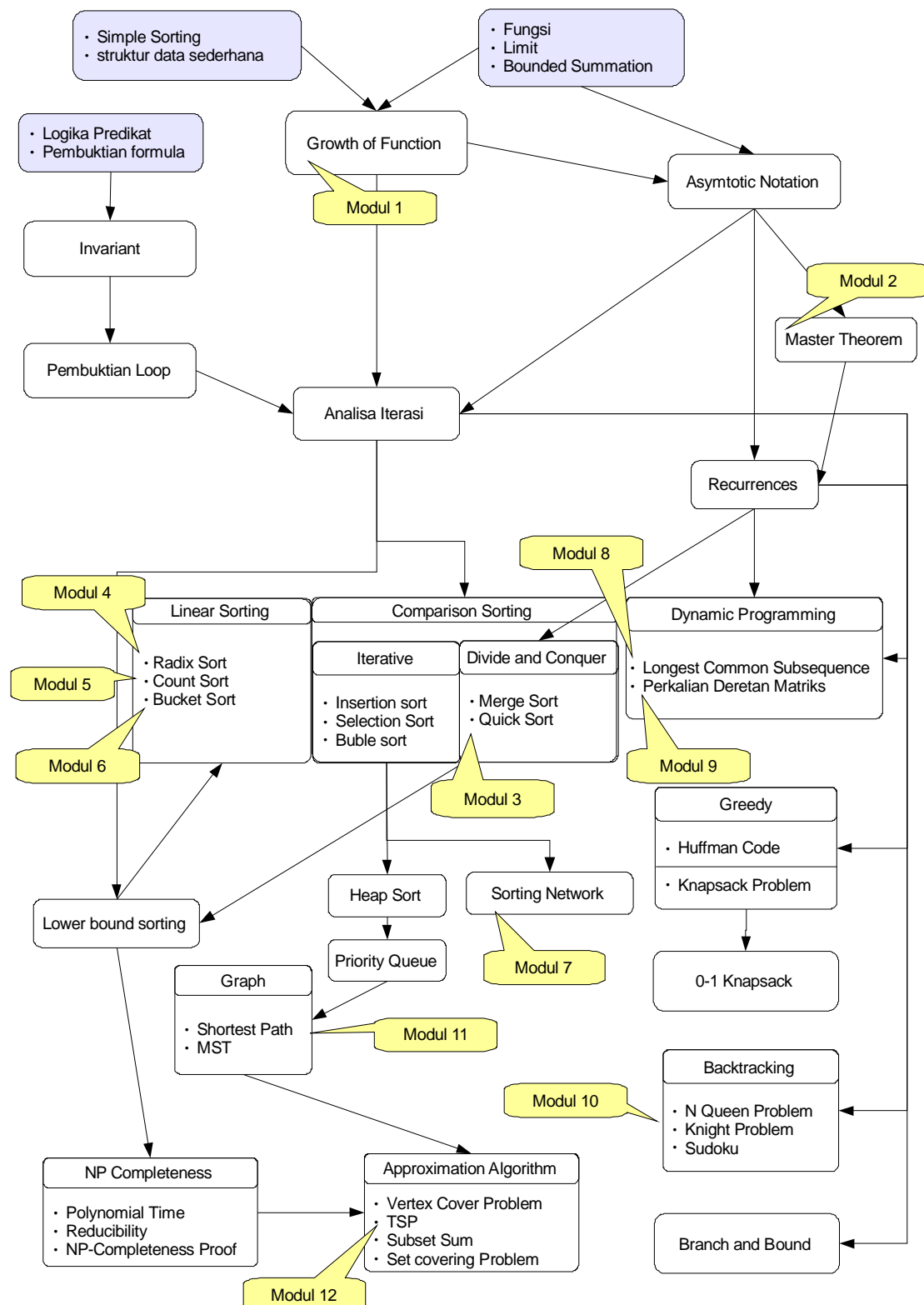
Sasaran pembelajaran penunjang

Setelah selesai mengikuti mata kuliah ini, peserta didik diharapkan memiliki kemampuan sebagai berikut:

1. Mahasiswa memahami peranan dan pentingnya desain algoritma dan struktur data dalam komputasi.
2. Mahasiswa memahami laju pertumbuhan fungsi yang merepresentasikan laju kompleksitas algoritma dan dapat menyusun urutan fungsi-fungsi yang diberikan berdasarkan laju pertumbuhannya.
3. Mahasiswa mengenal *Asymptotic Notation*.

4. Mahasiswa mengenali konsep invarian dalam loop dan dapat membuat invarian.
5. Mahasiswa dapat menguji kebenaran program iterasi, baik *partial correctness* maupun *total correctness* dari loop.
6. Mahasiswa dapat menganalisa kompleksitas algoritma iterasi dan menyatakannya dengan *asymptotic notation*.
7. Mahasiswa mengenal algoritma rekursif dan memahami analisa kompleksitasnya berdasarkan persamaan rekurensi dan *trace* eksekusi program.
8. Mahasiswa dapat menerapkan Master Theorem untuk menentukan kompleksitas algoritma rekursif melalui persamaan rekurensinya.
9. Mahasiswa memahami berbagai macam jenis sorting.
10. Mahasiswa dapat merancang dan menganalisa algoritma dengan metode *divide and conquer*.
11. Mahasiswa dapat merancang dan menganalisa algoritma dengan metode *dynamic programming*.
12. Mahasiswa dapat merancang dan menganalisa algoritma dengan metode *greedy*.
13. Mahasiswa dapat merancang dan menganalisa algoritma dengan metode *backtracking*.
14. Mahasiswa dapat merancang dan menganalisa algoritma dengan metode *branch and bound*.
15. Mahasiswa memahami adanya batasan kompleksitas/efisiensi algoritma pada sebuah model komputasi (*Turing Machine*).
16. Mahasiswa memahami adanya *approximation algorithms* dan contoh-contohnya.
17. Mahasiswa memahaminya adanya beberapa topik-topik lain yang memerlukan perancangan dan analisa yang lebih khusus. Mahasiswa diberikan kebebasan untuk memilih mendalami salah satunya melalui tugas makalah.

Diagram alur pokok bahasan



Keterangan:

- Kotak bertanda panah menyatakan ketersediaan media penunjang e-learning dan self study pada sub pokok bahasan yang ditunjuk.
- Warna latar gelap menyatakan sub pokok bahasan yang sudah harus dikuasai sebelumnya dan harus disediakan pada kuliah prasyarat.

Modul-Modul E-Learning

Pada kuliah ini digunakan bantuan dua belas (12) modul e-learning. Modul-modul e-learning tersebut adalah alat bantu peserta kuliah agar dapat memahami pokok bahasan secara mandiri dalam menunjang pemahaman. Berikut adalah 12 modul tersebut. Penjelasan rinci meliputi *objectives*, *topic mind map*, dan *self evaluation question*, dapat dilihat pada Bab 8.

Modul 1: Growth of Function

Modul 2: Master Theorem

Modul 3: Quick Sort

Modul 4: Radix Sort

Modul 5: Count Sort

Modul 6: Bucket Sort

Modul 7: Sorting Network

Modul 8: Longest Common Subsequence

Modul 9: Perkalian Deretan Matriks

Modul 10: Shortest Path

Modul 11: Knight Problem

Modul 12: Traveling Salesman Problem

SUBPOKOK BAHASAN DAN RUJUKAN

| No | Pokok Bahasan | Subpokok bahasan | Rujukan | Modul |
|----|--------------------------|---|---------------------------------------|---------------------------------------|
| 1 | Dasar-dasar | 1. Peran algoritma dalam komputasi 2. Growth of function | Bab 1 Bab 3 | Modul 1 |
| 2 | Recurrences | 1. Ide rekursif 2. Analisa program rekursif 3. Master Theorem | Bab 4 | |
| 3 | Sorting | 1. Algoritma Sorting iterative 2. QuickSort 3. HeapSort 4. Lower bound pada sorting 5. Sorting dalam waktu linear 6. Sorting network | Bab 6 Bab 7 Bab 8 Bab 27 | Modul 2 Modul 3,4,5 Modul 7 |
| 4 | Pembuktian Loop | 1. Invariant 2. Partial correctness 3. Total correctness (terminasi loop) | | |
| 5 | Analisa Data Structure | 1. Struktur data dasar 2. Hash Table 3. Binary Search Trees | Bab 10 Bab 11 Bab 12 | |
| 6 | Dynamic Programming | 1. Ide perancangan 2. Contoh: LCS 3. Contoh: Matriks chain multiplication | Bab 15 | Modul 8 Modul 9 |
| 7 | Greedy Algorithms | 1. Ide perancangan 2. Contoh: Knapsack problem 3. Contoh: Huffman code | Bab 16 | |
| 8 | Backtracking | 1. Ide perancangan 2. Contoh: N Queen Problem 3. Contoh: Knight Problem 4. Contoh: Sudoku 5. Branch and bound | Bab 22 | Modul 10 |
| 9 | Graph | 1. Representasi 2. Algoritma: Shortest Path 3. Algoritma: Minimum spanning tree | Bab 23 Bab 24 Bab 25 | Modul 11 |
| 10 | NP-Completeness | 1. Konsep dan definisi 2. Polynomial time 3. Reducibility 4. NP-Completeness proof | Bab 34 | |
| 11 | Approximation Algorithms | 1. Motivasi 2. Contoh: Vertex Cover 3. Contoh: TSP 4. Subset sum 5. Set covering problem | Bab 35 | Modul 12 |

Rujukan

Utama

- [1] Cormen, T.H., et.al, *Introduction to Algorithms*. 2nd Edition. MIT Press/McGraw-Hill, 2001. <http://mitpress.mit.edu/algorithms>.

Penunjang

- [2] Weiss, M.A., *Data Structure and Problem Solving Using Java*. 3rd Edition. Pearson Education. Addison Wesley, 2004.

MATRIKS KEGIATAN**Metode pembelajaran:****Orientasi Belajar**

Belajar yang dilaksanakan dalam perkuliahan ini akan dilakukan secara *blended* (pencampuran antara perkuliahan tatap muka dengan belajar mandiri). *Blended learning* yang diterapkan akan menghasilkan rasio pembelajaran antara lain 30% tatap muka dan 70 % belajar mandiri.

1. Tatap Muka (TM)

Perkuliahan interaktif atau tatap muka dilakukan di kelas secara langsung. dalam perkuliahan ini metode yang dilakukan antara lain:

- a. Ceramah; dosen menjelaskan mengenai materi tertentu. Dan metode ini dapat dikombinasikan dengan tanya jawab secara langsung.
- b. Diskusi kelompok; mahasiswa membahas topik-topik tertentu yang ada dalam materi.

2. Belajar Mandiri (BM)

Belajar mandiri atau self-learning dilakukan tidak secara langsung bertatap muka melainkan dengan memanfaatkan SCeLE. Dengan SCeLE, pembelajaran akan melibatkan kegiatan-kegiatan sebagai berikut:

- a. Akses materi (A); mahasiswa dapat mengakses materi langsung yang telah disediakan di SCeLE.
- b. Diskusi (D); mahasiswa dapat melakukan diskusi dalam bentuk aktifitas forum yang ada di SCeLE.
- c. Chat (C); mahasiswa dapat pula berkomunikasi dengan pengajar dan teman-teman sekelas dalam kurun waktu yang telah ditentukan.

Tugas

1. Tugas Individu (I); dengan feedback langsung yang disesuaikan dengan kondisi.
2. Tugas Kelompok (K); dengan feedback langsung yang disesuaikan dengan kondisi.
3. Tugas Makalah (M); dengan feedback langsung yang disesuaikan dengan kondisi.
4. Quiz (Q); dilaksanakan pada topik-topik tertentu. Pada proses belajar tatap muka maupun belajar mandiri.








Media Instruksional

1. Internet (I)
2. Presentasi Multimedia (M)
3. White board, infocus (Wbi)

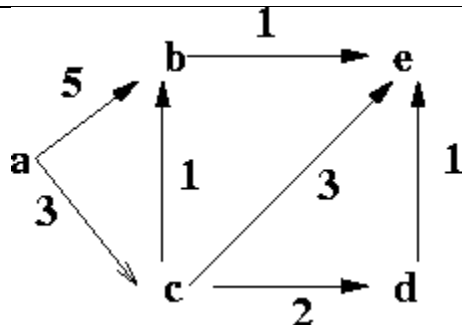
Sumber Pemelajaran

1. Buku Teks
2. Handout
3. Modul e-learning

Matriks Kegiatan Perkuliahan

| Minggu | Sasaran Pemelajaran | | Orientasi Belajar | |  Media Instruksional |  Tugas | Sumber Pemelajaran | | |
|--------|-----------------------|---------|---|---|--|--|---|---|--|
| | Umum | Khusus |  Tatap Muka |  Belajar Mandiri | | |  Subpokok bahasan |  Rujukan/ Buku Teks |  Modul ELearning |
| 1 | 4 | 1,2,3 | Ceramah | A | I, M & Wbi | - | 1.1 – 1.2 | Bab 1, 3 | Modul 1 |
| 2 | 1,3 | 7,8 | Ceramah | A | Wbi | - | 2.1 – 2.3 | Bab 4 | - |
| 3 | 1,3 | 6,9,10 | Diskusi | A, D, C | I, M & Wbi | Individu | 3.1 – 3.2 | Bab 6,7 | Modul 2,3,4 |
| 4 | 1,3 | 6,9,10 | Diskusi | A, D, C | I, M & Wbi | Quiz | 3.3 – 3.6 | Bab 8,27 | Modul 5,6,7 |
| 5 | 2 | 4,5 | Ceramah | D | I & Wbi | - | 4.1 – 4.3 | Handout | - |
| 6 | 1,4 | 1,2,6,9 | Ceramah | A | I & Wbi | Kelompok | 5.1 – 5.3 | Bab 10-12 | - |
| 7 | Ujian Tengah Semester | | | | | | | | |
| 8 | 1,3 | 11 | Diskusi | A, D, C | I, M & Wbi | - | 6.1 – 6.3 | Bab 15 | Modul 8,9 |
| 9 | 1,3 | 12 | Ceramah | A | I & Wbi | - | 7.1 – 7.3 | Bab 16 | - |
| 10 | 1,3 | 13,14 | Diskusi | A, D, C | M & Wbi | Kelompok | 8.1 – 8.3 | Bab 22 | Modul 10 |
| 11 | 1,3 | 13,14 | Ceramah | A | I & Wbi | Quiz | 8.4 – 8.5 | Handout | - |
| 12 | 1,2,3 | 1,17 | Diskusi | A, D, C | I, M & Wbi | - | 9.1 – 9.3 | Bab 23-25 | Modul 11 |
| 13 | 4 | 15,17 | Ceramah | A, D | I & Wbi | Makalah | 10.1-10.3 | Bab 34 | - |
| 14 | 1,4 | 16,17 | Ceramah | A | I & M | Quiz | 11.1-11.5 | Bab 35 | Modul 12 |
| 15 | Ujian Akhir Semester | | | | | | | | |

CONTOH-CONTOH PERTANYAAN PENGARAH

| Minggu | Soal | Ket |
|--------|---|-----|
| 5 | <p>1. Tentukan invarian mana yang cocok untuk program berikut:</p> <pre> i := n; while i > 0 do i := i - 1 </pre> <p>Input : sebuah bilangan positif</p> <p>output: bilangan tersebut menjadi 0</p> <p>(a) F</p> <p>(b) $0 \leq i$</p> <p>(c) $0 \leq i < n$</p> <p>(d) T</p> | |
| 5 | <p>Tentukan urutan fungsi-fungsi berikut ini:</p> <ul style="list-style-type: none"> $2^{\lg n}$ $n \lg n$ $n \cdot 2^n$ $(\sqrt{2})^{\lg n}$ $(\frac{3}{2})^n$ $\lg n$ n^2 n^3 n $n!$ 2^{2n} 2^n \sqrt{n} $4^{\lg n}$ $\lg n!$ | |
| 12 |  <p>Simulasikan algoritma shortest path pada contoh graph di atas.</p> | |

EVALUASI HASIL PEMELAJARAN**Bentuk/jenis instrumen**

1. Tugas individu
2. Tugas Kelompok
3. Ujian Tertulis (*essay*, jawaban singkat)
4. Makalah

Skema Penilaian:

| | |
|----------------|------|
| Tugas Individu | 15 % |
| Tugas Kelompok | 15 % |
| UTS | 25 % |
| UAS | 25 % |
| Tugas Makalah | 20 % |

CONTOH-CONTOH SOAL

Contoh Soal Ujian

1. Which invariants would be sufficient to prove the validity of this program:

```
i := n;
while i > 0 do i := i - 1
```

The input is : a positive number.

The output is : the given number become 0.

- (a) F
- (b) $0 \leq i$
- (c) $0 \leq i < n$
- (d) T

explanation:

2. Below is a program to check if all element in an array, are having the same value.
input: an array
output: are the elements of the array having the same value.
program:

```
i:=0;
s:=T;
while i < n - 1 do
{ s := s ^ (a[i] = a[i + 1]) ;
  i := i+1
}
```

Choose the correct invariant:

- (a) $(s = (\text{FORALL } k : 0 \leq k < i : a[k] = a[k + 1]))$
 \wedge
 $0 \leq i < n$
- (b) $0 \leq i < n$
- (c) $(s = (\text{FORALL } k : 0 \leq k < i : a[k]))$
 \wedge
 $0 \leq i < n$
- (d) $(s = (\text{SIGMA } k : 0 \leq k < i : a[k]))$
 \wedge
 $0 \leq i < n$

explanation:

3. Which of the facts below are the most acceptable statements about "algorithm correctness" ?

- (a) a correct algorithm should eventually stop
- (b) a correct algorithm will always have trivial invariant
- (c) a correct algorithm solve the given problem
- (d) a correct algorithm may not terminate

explanation:

4. Which of the following are not relevant in proving the correctness of an algorithm:

- (a) The invariant should show that the expected result is achieved when the loop terminate
- (b) The inisialisation of the loop should maintain the invariant.
- (c) The invariant should always be correct with in the execution of the loop.
- (d) The invariant should show that the loop will terminate

explanation:

Part 2:

5. Below is the classic Euclid's Algorithm to find

GCD (Greatest Common Divisor - FPB: Faktor Persekutuan Terbesar)

input: Given two positive numbers.

Output: the GCD of those numbers.

Algorithm:

```
int GCD(int a, int b){
    int a0, a1, temp;
    a0 := a;
    a1 := b;
    while a1 != 0 do {
        temp := a1;
        a1 := a0 mod a1;
        a0 := temp
    }
    return a0;
}
```

Choose the correct invariant and prove the algorithm!

6. Consider the sum of array example.

input : an array aa[0..N]

output: the sum of all element in array aa.

We would like to have less iteration,
we modify the algorithm such as:

```
i := 0;
total = 0;
while i < N/2 {
    total := total + aa[i] + aa[N-1-i];
    i := i + 1;
}
```

Is the algorithm still correct?

if yes, prove it.

if no, fix it, and prove that your revised version is correct.

MODUL-MODUL PEMELAJARAN MANDIRI

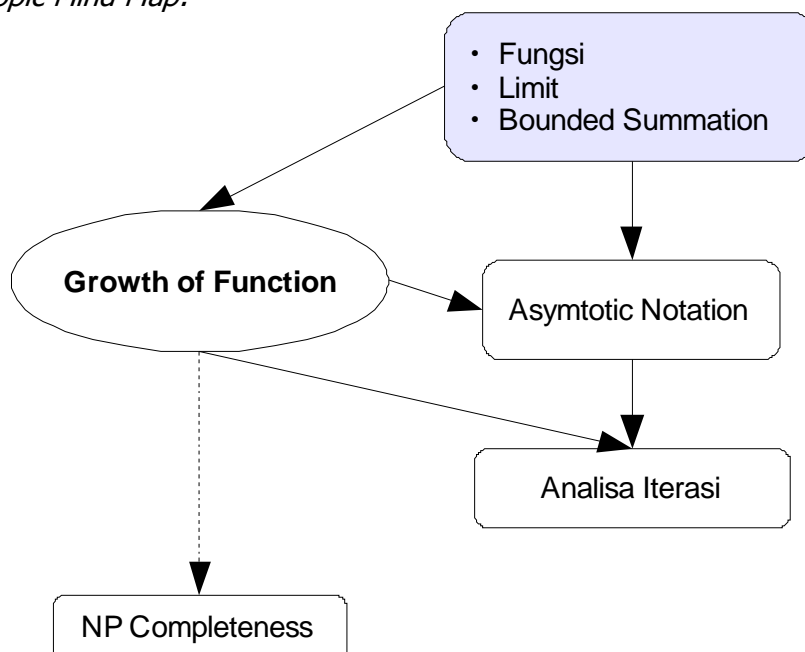
Pada kuliah ini digunakan dua belas (12) modul e-learning untuk belajar mandiri. Berikut penjelasan rinci tiap modul meliputi *session objectives*, *topic mind map*, dan *self evaluation question*. Keterkaitan modul-modul ini dalam peta pemelajaran dapat dilihat pada Bab 8.

Modul 1: Growth of Function

Modul ini diharapkan dapat membantu mahasiswa membedakan laju pertumbuhan antara suatu fungsi dengan fungsi lain. Dengan bantuan gambar dan animasi diharapkan mahasiswa dapat lebih mudah memahami dan mengingat perbedaan serta mampu mengurutkan fungsi-fungsi tersebut.

Objectives:

- Mahasiswa memahami laju pertumbuhan suatu fungsi
- Mahasiswa mengenal perbedaan laju pertumbuhan beberapa fungsi
- Mahasiswa dapat mengelompokkan beberapa kelompok utama laju pertumbuhan fungsi
- Mahasiswa dapat menentukan urutan laju pertumbuhan fungsi.

Topic Mind Map:

Soal Evaluasi Diri Pemelajaran Mandiri:

Gambarkan laju pertumbuhan dan tentukan urutan fungsi-fungsi berikut ini:

- $2^{\lg n}$ • $n \lg n$ • $n \cdot 2^n$ • $(\sqrt{2})^{\lg n}$ • $(\frac{3}{2})^n$
- $\lg n$ • n^2 • n^3 • n • $n!$
- 2^{2n} • 2^n • \sqrt{n} • $4^{\lg n}$ • $\lg n!$

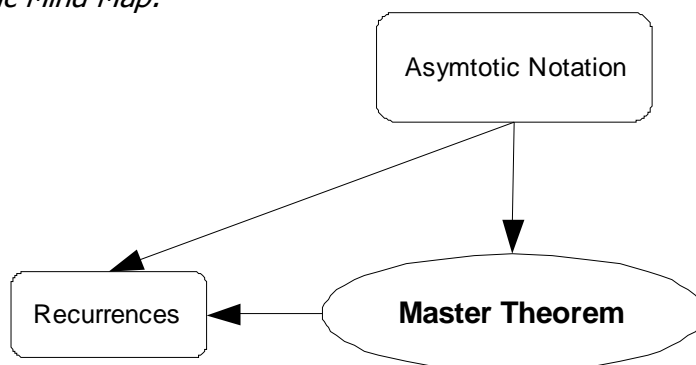
Modul 2: Master Theorem

Pokok bahasan Master Theorem membutuhkan hafalan dan kemampuan mahasiswa untuk melihat pola dan kesesuaian formula rekurensi yang hendak dianalisa dengan formula pada Master Theorem. Modul ini diharapkan dapat membantu mahasiswa untuk melihat dan mengingat pola tersebut.

Objectives:

- Memahami pola Master Theorem
- Memahami syarat-syarat menerapkan Master Theorem
- Dapat menerapkan Master Theorem
- Dapat mengenal persamaan rekurensi yang tidak bisa diselesaikan menggunakan Master Theorem.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Lengkapi (...) pada pernyataan berikut, upayakan hal ini dapat anda lakukan luar kepala!

Master Theorem: Diketahui $a \geq 1$ dan $b \geq 1$, jika $f(n)$ sebuah fungsi bilangan non-negatif, dan jika $T(n)$ didefinisikan sebagai persamaan rekurensi: $T(n) = \dots$

Maka kemungkinan $T(n)$ secara asimptotik adalah :

- ❶ Jika $f(n) = O(n^{\log_b a - \epsilon})$, untuk $\epsilon > 0$, maka $T(n) = \dots$
- ❷ Jika $f(n) = \dots$, untuk $k \geq 0$, maka $T(n) = \dots$
- ❸ Jika $f(n) = \dots$, untuk $\epsilon > 0$, dan jika $a f(n/b) \leq c f(n)$, untuk $c < 1$ dan n yang cukup besar, maka $T(n) = \dots$

Coba terapkan *Master Theorem* pada persamaan berikut:

- ❶ $T(n) = 8 T(n/2) + \Theta(n^2)$
- ❷ $T(n) = 4 T(n/2) + \Theta(n^3)$
- ❸ $T(n) = 8 T(n/2) + \Theta(n^3 \lg n)$
- ❹ $T(n) = 5 T(n/2) + \Theta(n^2)$
- ❺ $T(n) = 27 T(n/3) + \Theta(n^3 \lg n)$
- ❻ $T(n) = 5 T(n/2) + \Theta(n^3)$
- ❼ $T(n) = 27 T(n/3) + \Theta(\frac{n^3}{\lg n})$

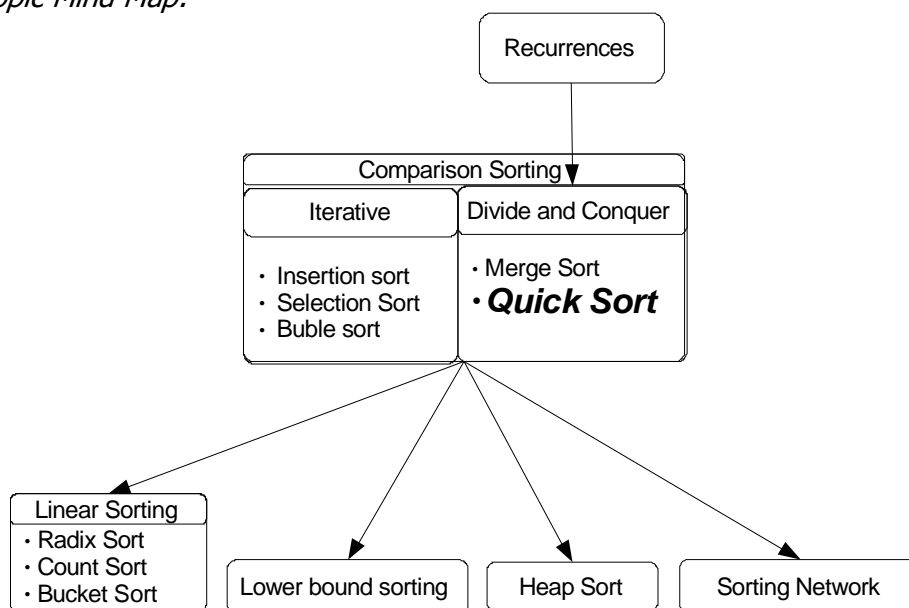
Modul 3: QuickSort

Modul ini memberikan contoh animasi untuk algoritma QuickSort. Animasi ini dibutuhkan agar memudahkan mahasiswa untuk membayangkan cara kerja algoritma sorting khususnya algoritma QuickSort dan partisinya.

Objectives:

- Mahasiswa dapat memahami cara kerja algoritma QuickSort
- Mahasiswa dapat memahami cara kerja partisi dalam algoritma QuickSort
- Mahasiswa dapat mengenali invariant yang dibutuhkan algoritma ini
- Mahasiswa dapat menganalisa kompleksitas algoritma QuickSort.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Tentukan langkah-langkah algoritma QuickSort untuk mengurutkan deretan bilangan berikut ini: 7, 4, 9, 3, 10, 3, 6, 2, 5.

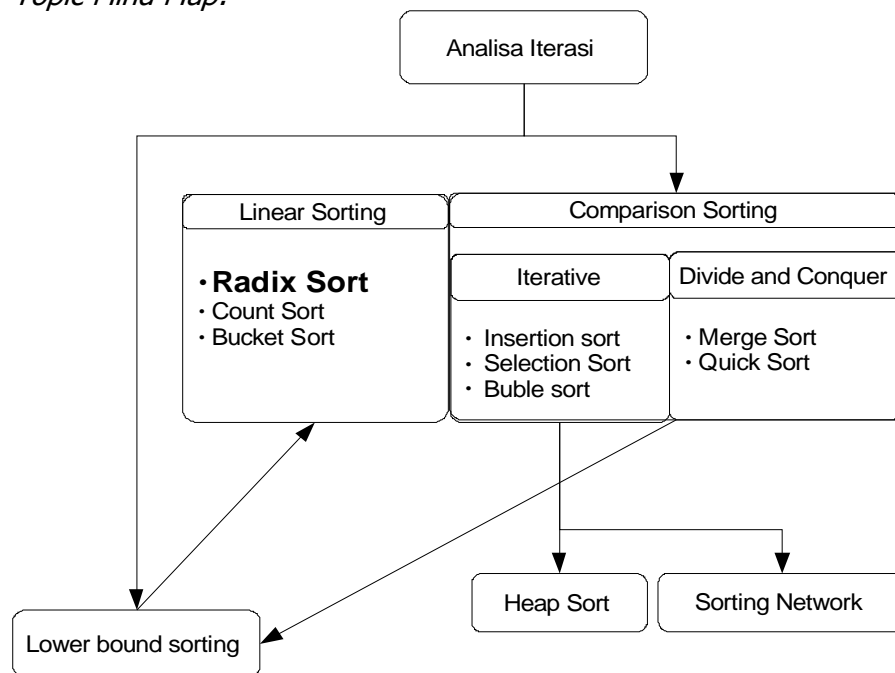
Modul 4: RadixSort

Modul ini secara spesifik mengilustrasikan cara kerja algoritma linear sorting yang disebut *RadixSort*.

Objectives:

- Mahasiswa memahami cara kerja algoritma *RadixSort*
- Mahasiswa memahami mengapa algoritma ini termasuk linear sorting, dapat melakukan sorting secara linear $O(n)$
- Mahasiswa memahami mengapa algoritma sorting untuk tiap digit haruslah menggunakan algoritma sorting yang *stable*.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

- Ilustrasikan operasi pada algoritma *Radix Sort* dalam mengurutkan kata-kata berikut ini: SAPI, KUDA, KAYU, SAPU, SAYA, TAHU, DARI, TADI, TAPI, DARA, JAKA, BARU.
- Tunjukkan bagaimana caranya mengurutkan sejumlah n bilangan yang nilainya berkisar antara 1 sampai n^2 dalam waktu $O(n)$.

Modul 5: CountSort

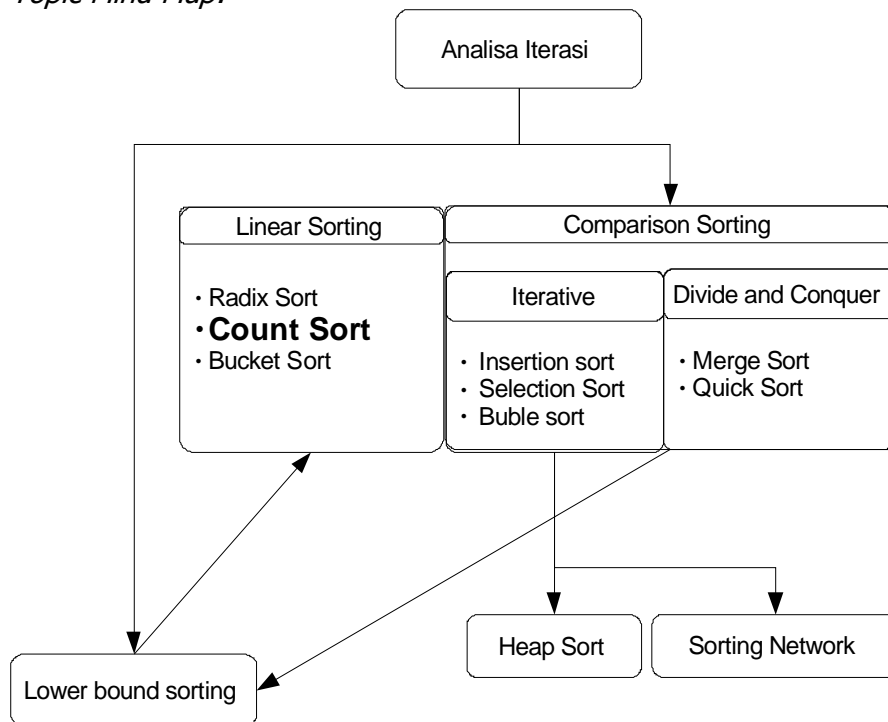
Modul ini secara spesifik mengilustrasikan cara kerja algoritma linear sorting yang disebut *CountSort*.

Objectives:

- Mahasiswa memahami cara kerja algoritma *CountSort*.
- Mahasiswa memahami asumsi yang dibutuhkan agar algoritma *CountSort* dapat melakukan sorting dalam waktu linear
- Mahasiswa memahami mengapa algoritma ini termasuk linear sorting, dapat melakukan sorting secara linear $O(n)$

- Mahasiswa memahami mengapa algoritma ini termasuk *stable*.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

- Ilustrasikan operasi algoritma *CountSort* pada input array berikut ini: 8, 4, 8, 6, 7, 4, 7, 7, 9, 8, 1, 2, 7, 4, 2, 6.

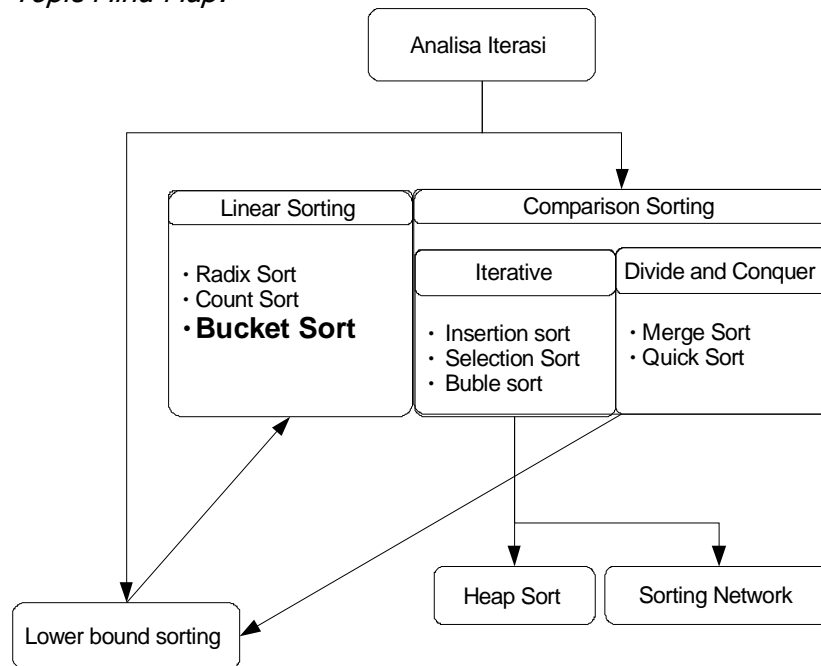
Modul 6: BucketSort

Modul ini secara spesifik mengilustrasikan cara kerja algoritma linear sorting yang disebut *BucketSort*.

Objectives:

- Mahasiswa memahami cara kerja algoritma *BucketSort*
- Mahasiswa memahami mengapa algoritma ini termasuk linear sorting, dapat melakukan sorting secara linear $O(n)$
- Mahasiswa memahami asumsi yang dibutuhkan agar algoritma *BucketSort* dapat melakukan sorting dalam waktu linear.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

- Ilustrasikan operasi algoritma *BucketSort* pada input array berikut ini: 0.8, 0.4, 0.8, 0.6, 0.75, 0.4, 0.7, 0.75, 0.9, 0.8, 0.15, 0.2, 0.7, 0.45, 0.2, 0.6.
- Tunjukkan perbedaan antara *CountSort* dengan *BucketSort*.

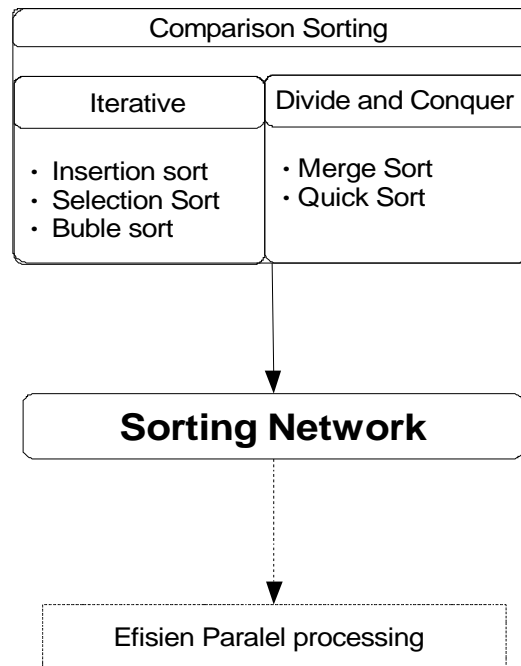
Modul 7: Sorting Network

Modul ini diharapkan dapat memberikan gambaran lebih jelas kepada mahasiswa tentang *Sorting network*. Animasi akan menunjukkan bagaimana beberapa comparator bekerja secara independent dan concurrent sehingga dapat mengurutkan elemen.

Objectives:

- Mahasiswa memahami bahwa beberapa operasi independent dapat dilakukan secara bersamaan
- Mahasiswa memahami bahwa tahapan melakukan sorting bisa disusun sebagai sebuah network comparator. Mahasiswa memahami bahwa comparator tersebut berkerja masing-masing
- Mahasiswa memahami bahwa susunan dari network penting untuk menjamin sehingga proses sorting berlangsung dengan efisien dan benar.

Topic Mind Map:

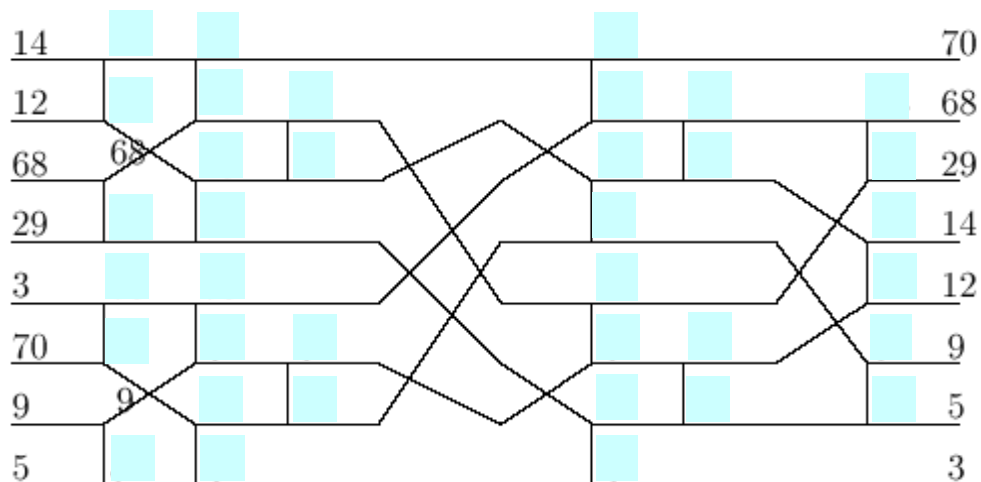


Soal Evaluasi Diri Pemelajaran Mandiri:

Gunakan sorting network ukuran 8 untuk mengurutkan bilangan-bilangan berikut:

14, 12, 68, 29, 3, 70, 9, 5.

Lengkapi gambar network berikut:



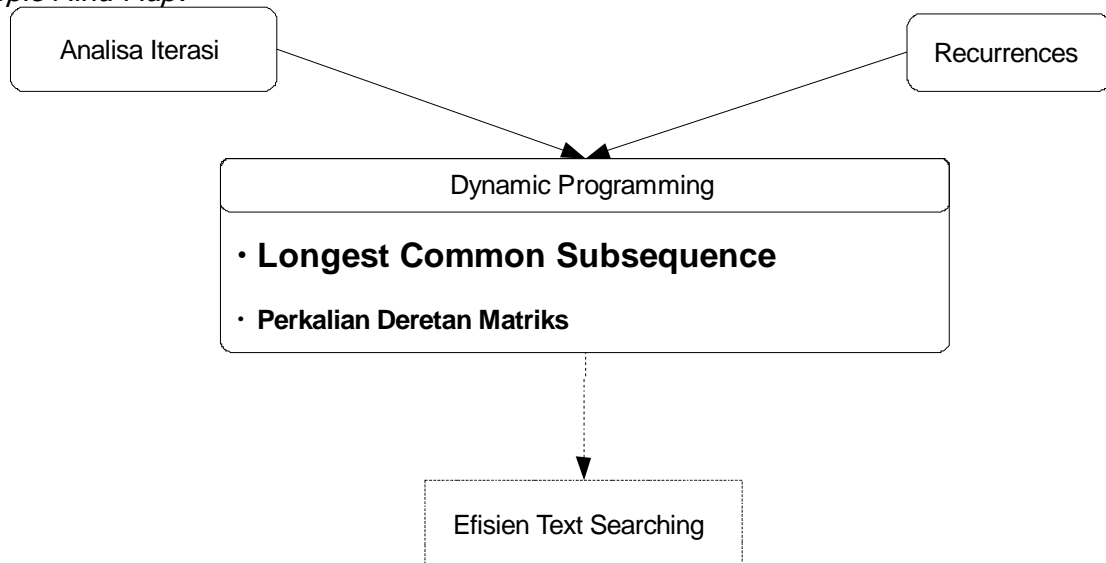
Modul 8: Longest Common Subsequence

Pokok bahasan *Dynamic Programming* termasuk pokok bahasan yang sulit dipahami oleh mahasiswa. Oleh karena itu dibutuhkan beberapa contoh dan alat bantu untuk menjelaskan contoh tersebut. Modul 4 memberikan animasi yang menjelaskan permasalahan Longest Common Subsequence yang dapat diselesaikan secara efisien dengan pendekatan *Dynamic Programming*.

Objectives:

- Mahasiswa memahami salah satu contoh permasalahan yang dapat diselesaikan dengan *Dynamic Programming* yaitu: *Longest Common Subsequence*
- Mahasiswa memahami bagaimana solusi untuk sebuah problem disusun dari solusi sub-problem pada kasus *Longest Common Subsequence*
- Mahasiswa memahami bagaimana cara 'mencatat' hasil sementara dalam 'tabel' dan menggunakannya sehingga mendapat hasil yang optimal pada kasus *Longest Common Subsequence*.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Gunakan *Dynamic Programming* untuk mencari *longest common subsequence* dari dua deretan bit berikut ini: (1,0,0,1,0,1,1,1) dan (0,1,0,1,1,0,1,1,0).

Modul 9: Perkalian Deretan Matriks

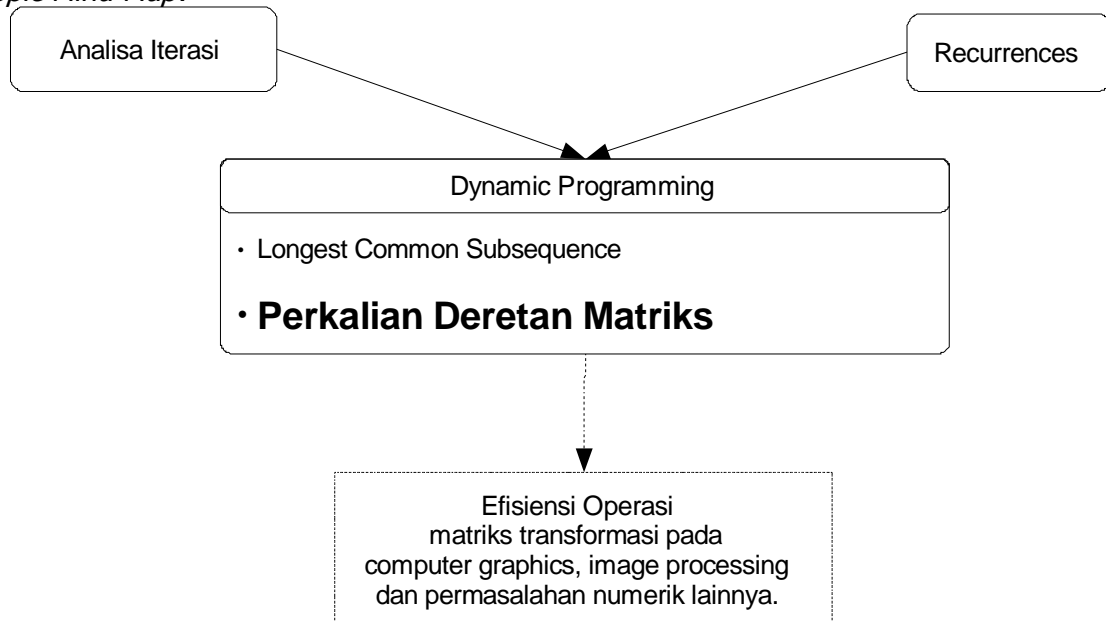
Modul 5 memberikan animasi penerapan teknik *Dynamic Programming* untuk dapat menyelesaikan permasalahan Perkalian Deretan Matriks dengan efisien. Mahasiswa dapat melihat dari animasi ini bagaimana cara kerja algoritma sehingga dapat menentukan pengelompokan matriks dengan biaya operasi perkalian yang paling efisien.

Objectives:

- Mahasiswa memahami salah satu contoh permasalahan yang dapat diselesaikan dengan *Dynamic Programming* yaitu: Perkalian Deretan Matriks dengan jumlah operasi perkalian yang optimal

- Mahasiswa memahami bahwa mengalikan sederetan matriks dapat dilakukan dengan banyak cara, masing-masingnya butuh jumlah operasi dasar yang berbeda, namun menghasilkan satu matriks yang sama
- Mahasiswa memahami bagaimana solusi untuk sebuah problem disusun dari solusi sub-problem pada kasus Perkalian Deretan Matriks
- Mahasiswa memahami bagaimana cara 'mencatat' hasil sementara dalam 'tabel' dan menggunakannya sehingga mendapat hasil yang optimal pada kasus Perkalian Deretan Matriks.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Buatlah tabel penerapan *Dynamic Programming* dari perkalian deretan matriks dari matriks dengan ukuran terurut berikut : 30, 35, 15, 5, 10, 20, 25.

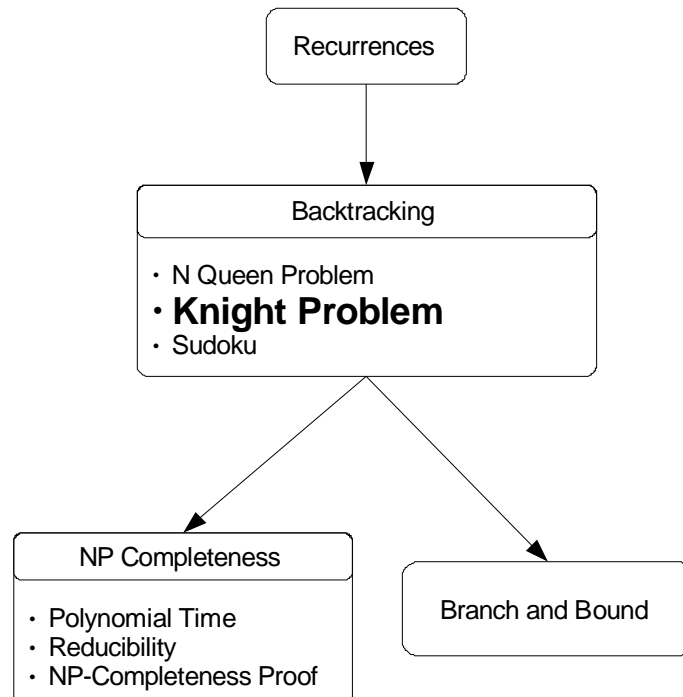
Modul 10: Knight Problem

Modul ini memberikan gambaran dan animasi dari algoritma yang melakukan *backtracking* saat mencari solusi pada Knight Problem.

Objectives:

- Mahasiswa memahami proses *backtracking* dalam proses pencarian solusi secara rekursif
- Mahasiswa memahami proses eksekusi yang terjadi saat *backtracking* dilakukan dan dapat menggambarkannya dalam bentuk *tree*
- Mahasiswa memahami penerapan rekursif dalam permasalahan *Knight Problem*.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Dalam sebuah papan catur, tentukan langkah-langkah yang dibutuhkan oleh kuda untuk berpindah dari satu posisi ke posisi lain. Tentukan algoritma yang digunakan. Gambarkan proses eksekusinya dalam bentuk tree pada proses perpindahan dari posisi *c1* ke *e8*. Jelaskan kapan *backtracking* terjadi.

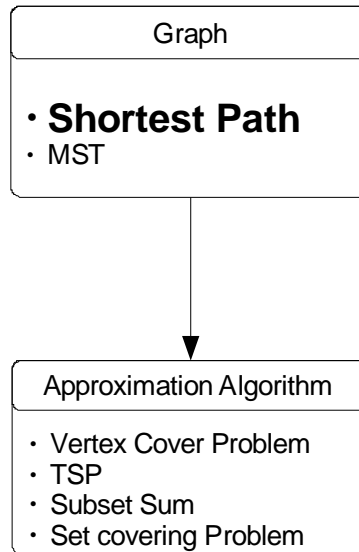
Modul 11: Shortest Path

Konsep Graph merupakan konsep yang sederhana namun tidak umum bagi mahasiswa ilmu komputer terutama karena dalam programming konsep graph tersebut masih harus dipetakan dalam struktur data konkret yang tersedia dalam bahasa yang digunakan. Modul ini memberikan gambaran bagaimana sebuah permasalahan pencarian jalan terpendek dapat dipecahkan dengan menggunakan algoritma pada Graph.

Objectives:

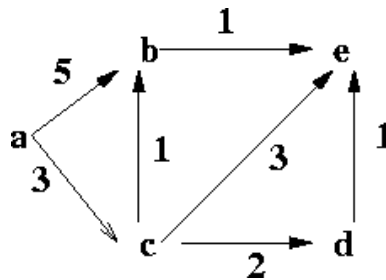
- Mahasiswa memahami bahwa penggunaan algoritma graph yang standar dapat mempermudah penyelesaian permasalahan dengan cara yang efisien
- Mahasiswa dapat merepresentasikan permasalahan dalam graph
- Mahasiswa dapat menerapkan algoritma shortest path Dijkstra untuk mencari shortest path dalam graph.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Terapkan algoritma shortest path untuk mencari jalan terpendek dari node *a* ke node *e*. Tuliskan langkah-langkahnya dengan teratur.



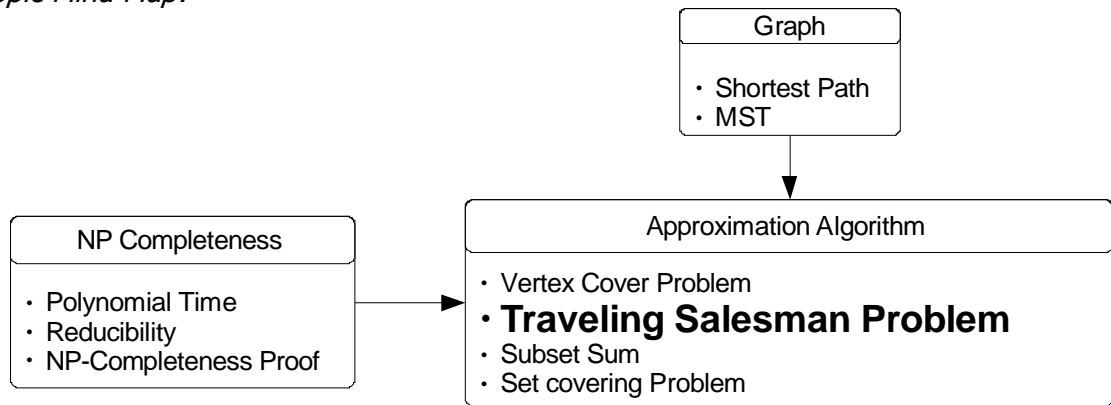
Modul 12: Traveling Salesman Problem

Modul ini memberikan contoh permasalahan yang memiliki kompleksitas *NP-Complete* dan coba dipecahkan dengan cara aproksimasi.

Objectives:

- Mahasiswa memahami bahwa beberapa permasalahan yang *NP-Complete* dapat dibuat algoritma aproksimasi (pendekatan) yang efisien namun tidak selalu benar
- Mahasiswa memahami bahwa solusi aproksimasi walaupun tidak menjamin solusi yang optimal namun dalam prakteknya hal ini dapat diterima dan cukup memuaskan karena algoritma aproksimasi bisa jauh lebih cepat dibanding algoritma pencarian solusi optimal
- Mahasiswa memahami contoh permasalahan Traveling Salesman Problem (TSP) sebagai permasalahan *NP-Hard* yang memiliki solusi algoritma aproksimasi.

Topic Mind Map:



Soal Evaluasi Diri Pemelajaran Mandiri:

Terapkan algoritma Traveling Salesman Problem pada gambar di bawah. Coba cari dan pilih beberapa algoritma aproksimasi lainnya dan bandingkan hasilnya, masing-masingnya dengan solusi optimal. Dalam kasus ini, solusi algoritma aproksimasi mana yang paling mendekati solusi optimal. Berdasarkan kompleksitas, algoritma mana yang paling efisien?

