M Reza Attthariq Kori
140810180060
Tugas 4

LAPORAN PRAKTIKUM

ANALISIS ALGORITMA

Disusun oleh

Muhammad Reza Atthariq kori

140810180060

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

SUMEDANG

2020

M Reza Atthariq Kori
140810180060
Tugas 4

Pendahuluan

PARADIGMA DIVIDE & CONQUER

Divide & Conquer merupakan teknik algoritmik dengan cara memecah input menjadi beberapa bagian, memecahkan masalah di setiap bagian secara **rekursif**, dan kemudian menggabungkan solusi untuk subproblem ini menjadi solusi keseluruhan. Menganalisis *running time* dari algoritma *divide & conquer* umumnya melibatkan penyelesaian rekurensi yang membatasi *running time* secara rekursif pada instance yang lebih kecil

PENGENALAN REKURENSI

- Rekurensi adalah persamaan atau ketidaksetaraan yang menggambarkan fungsi terkait nilainya pada input yang lebih kecil. Ini adalah fungsi yang diekspresikan secara rekursif
- Ketika suatu algoritma berisi panggilan rekursif untuk dirinya sendiri, *running time*-nya sering dapat dijelaskan dengan perulangan
- Sebagai contoh, running time worst case $T(n)$ dari algoritma merge-sort dapat dideskripsikan dengan perulangan:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

with solution $T(n) = \Theta(n \lg n)$.

BEDAH ALGORITMA MERGE-SORT

- Merupakan algoritma sorting dengan paradigma divide & conquer
- Running time worst case-nya mempunyai laju pertumbuhan yang lebih rendah dibandingkan insertion sort
- Karena kita berhadapan dengan banyak subproblem, kita notasikan setiap subproblem sebagai sorting sebuah subarray A[p..r]
- Inisialisasi, p=1 dan r=n, tetapi nilai ini berubah selama kita melakukan perulangan subproblem

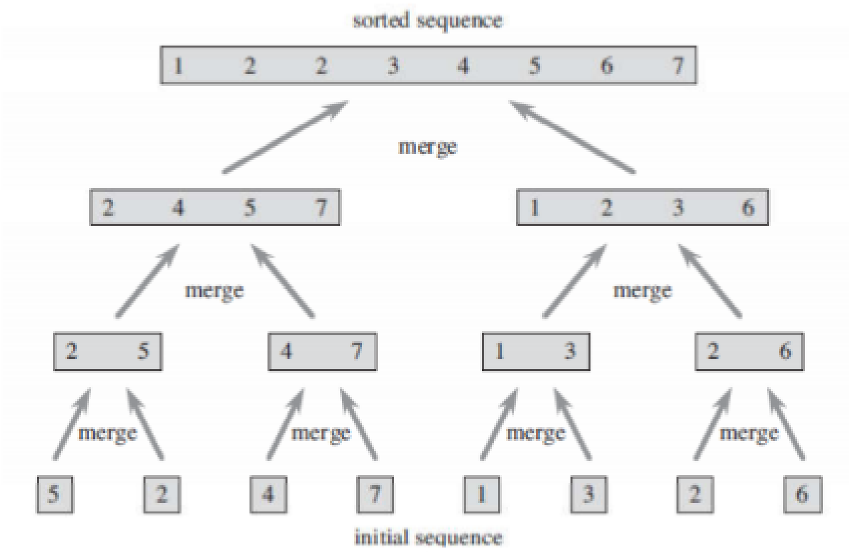
Untuk mengurutkan A[p..r]:

- **Divide** dengan membagi input menjadi 2 subarray A[p..q] dan A[q+1 .. r]
- **Conquer** dengan secara rekursif mengurutkan subarray A[p..q] dan A[q+1 .. r]
- **Combine** dengan menggabungkan 2 subarray terurut A[p..q] dan A[q+1 .. r] untuk menghasilkan 1 subarray terurut A[p..r]
- Untuk menyelesaikan langkah ini, kita membuat prosedur MERGE(A, p, q, r)
- Rekursi berhenti apabila subarray hanya memiliki 1 elemen (secara trivial terurut)

PSEUDOCODE MERGE-SORT

```
➢ MERGE-SORT(A, p, r)
  //sorts the elements in the subarray A[p..r]
  if p < r
     then q ← ⌊(p + r)/2⌋
          MERGE-SORT(A, p, q)
          MERGE-SORT(A, q + 1, r)
          MERGE(A, p, q, r)
```

M Reza Atthariq Kori
140810180060
Tugas 4



Gambar 1. Ilustrasi algoritma merge-sort

PROSEDUR MERGE

- Prosedur merge berikut mengasumsikan bahwa subarray A[p..q] dan A[q+1 .. r] berada pada kondisi terurut. Prosedur merge menggabungkan kedua subarray untuk membentuk 1 subarray terurut yang menggantikan array saat ini A[p..r] (input).
- Ini membutuhkan waktu Θ(n), dimana n = r-p+1 adalah jumlah yang digabungkan
- Untuk menyederhanakan code, digunakanlah elemen sentinel (dengan nilai ∞) untuk menghindari keharusan memeriksa apakah subarray kosong di setiap langkah dasar.

PSEUDOCODE PROSEDUR MERGE

```
MERGE(A, p, q, r)
1.  n1 ← q - p + 1;    n2 ← r - q
2.  //create arrays L[1 .. n1 + 1] and R[1 .. n2 + 1]
3.  for i ← 1 to n1  do  L[i] ← A[p + i - 1]
4.  for j ← 1 to n2  do  R[j] ← A[q + j]
5.  L[n1 + 1] ← ∞; R[n2 + 1] ← ∞
6.  i ← 1; j ← 1
7.  for k ← p to r
8.      do  if L[i] ≤ R[j]
9.              then  A[k] ← L[i]
10.                   i ← i + 1
11.             else  A[k] ← R[j]
12.                   j ← j + 1
```

RUNNING TIME MERGE

Untuk melihat running time prosedur MERGE berjalan di Θ(n), dimana n-p+1, perhatikan perulangan for pada baris ke 3 dan 4,

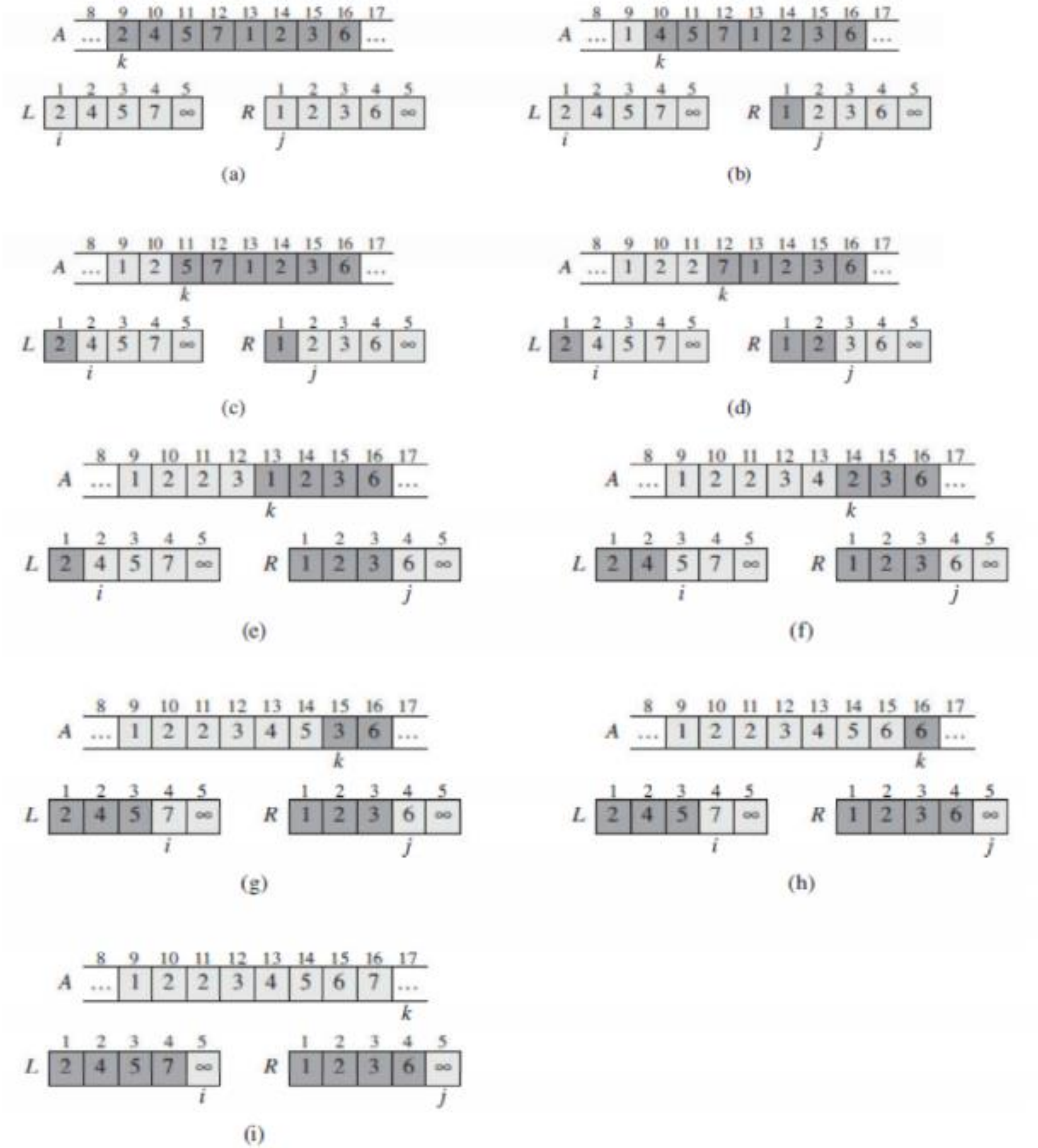
$$\Theta(n1 + n2) = \Theta(n)$$

dan ada sejumlah n iterasi pada baris ke 8-12 yang membutuhkan waktu konstan.

M Reza Atthariq Kori
140810180060
Tugas 4

CONTOH SOAL MERGE-SORT

MERGE(A, 9, 12, 16), dimana subarray A[9 .. 16] mengandung sekuen (2,4,5,7,1,2,3,6)



Algoritma merge-sort sangat mengikuti paradigma divide & conquer:

- **Divide** problem besar ke dalam beberapa subproblem
- **Conquer** subproblem dengan menyelesaikannya secara **rekursif**. Namun, apabila subproblem berukuran kecil, diselesaikan saja secara langsung.
- **Combine** solusi untuk subproblem ke dalam solusi untuk original problem

Gunakan sebuah persamaan rekurensi (umumnya sebuah perulangan) untuk mendeskripsikan running time dari algoritma berparadigma divide & conquer.

T(n) = running time dari sebuah algoritma berukuran n

- Jika ukuran problem cukup kecil (misalkan $n \le c$, untuk nilai c konstan), kita mempunyai *best case*. Solusi brute-force membutuhkan waktu konstan $\Theta(1)$
- Sebailknya, kita membagi input ke dalam sejumlah a subproblem, setiap $(1/b)$ dari ukuran

M Reza Atthariq Kori
140810180060
Tugas 4

original problem (Pada merge sort $a = b = 2$)

- Misalkan waktu yang dibutuhkan untuk membagi ke dalam n-ukuran problem adalah $D(n)$
- Ada sebanyak a subproblem yang harus diselesaikan, setiap subproblem (n/b) → setiap subproblem membutuhkan waktu T(n/b) sehingga kita menghabiskan $aT(n/b)$
- Waktu untuk **combine** solusi kita misalkan $C(n)$
- Maka persamaan **rekurensinya untuk divide & conquer** adalah:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

Setelah mendapatkan rekurensi dari sebuah algoritma divide & conquer, selanjutnya rekurensi harus diselesaikan untuk dapat menentukan kompleksitas waktu asimptotiknya. Penyelesaian rekurensi dapat menggunakan 3 cara yaitu, **metode subtitusi, metode recursion-tree dan metode master**. Ketiga metode ini dapat dilihat pada slide yang diberikan.

Studi Kasus

Studi Kasus 1: MERGE SORT

Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan Hal berikut:

1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

Jawaban :

Kompleksitas waktu merge sort = O(n log n). Jika input pada merge sortnya sebanyak 20 buah, maka

running timenya dapat ditulis sebagai berikut:

big-O -> T(20 $\log_{10}$ 20) = 26

Studi Kasus 2: SELECTION SORT

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma selection sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma selection sort
- Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

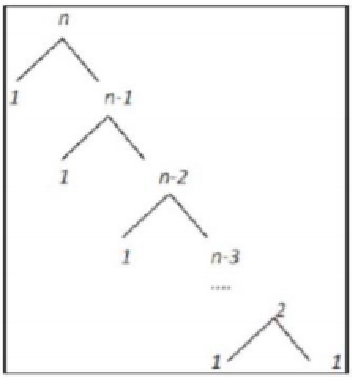
- Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

M Reza Atthariq Kori
140810180060
Tugas 4

Jawaban :

```
for i ← n downto 2 do {pass sebanyak n-1 kali}
        imaks ← 1
        for j ← 2 to i do
          if x_j > x_imaks then
             imaks ← j
          endif
        endfor
        {pertukarkan x_imaks dengan x_i}
        temp ← x_i
        x_i ← x_imaks
        x_imaks ← temp
endfor
```

$T(n) = \{\Theta(1)\ T(n-1) + \Theta(n)$



Subproblem =1,
Masalah setiap subproblem = n-1,
Waktu proses pembagian = n,
Waktu proses penggabungan = n

$$
\begin{aligned}
T(n) &= cn + cn\text{-}c + cn\text{-}2c + \ldots + 2c + cn \\
&= c((n-1)(n-2)/2) + cn \\
&= c((n^2-3n+2)/2) + cn \\
&= c(n^2/2)-(3n/2)+ 1 + cn \\
&= O(n^2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= cn + cn\text{-}c + cn\text{-}2c + \ldots + 2c + cn \\
&= c((n-1)(n-2)/2) + cn \\
&= c((n^2-3n+2)/2) + cn \\
&= c(n^2/2)-(3n/2)+ 1 + cn \\
&= \Omega(n^2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= cn^2 \\
&= \Theta(n^2)
\end{aligned}
$$

Studi Kasus 3: INSERTION SORT

Insertion sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma insertion sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma insertion sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode subtitusi** untuk mendapatkan

M Reza Atthariq Kori
140810180060
Tugas 4

kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

- Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++

Jawaban :

T(n) = { Θ(1)T(n-1) + Θ(n)
T(n) = cn + cn-c + cn-2c + … + 2c + cn <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + cn <= $2cn^2 + cn^2$
= c((n^2-3n+2/2) + cn <= $2cn^2 + cn^2$
= c(n^2/2) – c(3n/2) + c + cn <= $2cn^2 + cn^2$
= O(n^2)

T(n) = cn <= cn
= Ω(n)

T(n) = (cn +cn^2)/n
= Θ(n)

Studi Kasus 4: BUBBLE SORT

Bubble sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma bubble sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma bubble sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++

Jawaban :

Subproblem = 1
Masalah setiap subproblem = n-1
Waktu proses pembagian = n
Waktu proses penggabungan = n

T(n) = { Θ(1)T(n-1)+ Θ(n)
T(n) = cn + cn-c + cn-2c + … + 2c + c <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + c <= $2cn^2 + cn^2$
= c((n^2-3n+2/2) + c <= $2cn^2 + cn^2$
= c(n^2/2) – c(3n/2) + 2c <= $2cn^2 + cn^2$
= O(n^2)

M Reza Atthariq Kori
140810180060
Tugas 4

T(n) $= cn + cn\text{-}c + cn\text{-}2c + \ldots + 2c + c \leq 2cn^2 + cn^2$
$= c((n-1)(n-2)/2) + c \leq 2cn^2 + cn^2$
$= c((n^2-3n+2/2) + c \leq 2cn^2 + cn^2$
$= c(n^2/2) - c(3n/2) + 2c \leq 2cn^2 + cn^2$
$= \Omega(n^2)$

T(n) $= cn^2 + cn^2$
$= \Theta(n^{2)}$