TUGAS 2

ANALISIS ALGORITMA

Program Studi S-1 Teknik Informatika

Departemen Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Padjadjaran

2019

DISUSUN OLEH:

AITHRA JUNIA BOUTY

140810180054

Studi Kasus 1: Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

**Algoritma Pencarian Nilai Maksimal**

Jawaban Studi Kasus 1

Jenis-jenis operasi: Operasi assignment, operasi perbandingan, operasi penjumlahan

Kompleksitas waktu:

1. Operasi assignment:

* Baris 1) 1 kali
* Baris 2) 1 kali
* Baris 5) n-1 kali
* Baris 7) n-1 kali

T1 = 1 + 1+ n-1 + n-1 = 2n

1. Operasi perbandingan:

* Baris 4) n-1 kali

T2 = n-1

1. Operasi penjumlahan:

* Baris 7) n kali

T3 = n-1

Kompleksitas : T(n) = t1 + t2 + t3 = 2n + n-1 + n-1

= 4n – 2

procedure CariMaks(input x1, x2, …, xn: integer, output maks: integer)

{ Mencari elemen terbesar dari sekumpulan elemen larik integer x1, x2, …, xn. Elemen terbesar akan disimpan di dalam maks

Input: x1, x2, …, xn

Output: maks (nilai terbesar)

}

**Deklarasi**

i : integer

**Algoritma**

maks  x1 i  2

while i ≤ n do

if xi > maks then

maks  xi

endif

i  i + 1 endwhile

**PEMBAGIAN KOMPLEKSITAS WAKTU**

Hal lain yang harus diperhatikan dalam menghitung kompleksitas waktu suatu algoritma adalah parameter yang mencirikan ukuran input. Contoh pada algoritma pencarian, waktu yang dibutuhkan untuk melakukan pencarian tidak hanya bergantung pada ukuran larik (n) saja, tetapi juga bergantung pada nilai elemen (x) yang dicari.

Misalkan:

* + Terdapat sebuah larik dengan panjang elemen 130 dimulai dari y1, y2, … yn
  + Asumsikan elemen-elemen larik sudah terurut. Jika y1 = x, maka waktu pencariannya lebih cepat 130 kali dari pada y130 = x atau x tidak ada di dalam larik.
  + Demikian pula, jika y65 = x, maka waktu pencariannya ½ kali lebih cepat daripada

y130 = x

Oleh karena itu, kompleksitas waktu dibedakan menjadi 3 macam:

1. Tnin(n) : kompleksitas waktu untuk kasus terbaik (***best case***)

merupakan kebutuhan waktu minimum yang diperlukan algoritma sebagai fungsi dari n.

1. Tavg(n) : kompleksitas waktu untuk kasus rata-rata (***average case***)

merupakan kebutuhan waktu rata-rata yang diperlukan algoritma sebagai fungsi dari n. Biasanya pada kasus ini dibuat asumsi bahwa semua barisan input bersifat sama. Contoh pada kasus *searching* diandaikan data yang dicari mempunyai peluang yang sama untuk tertarik dari larik.

1. Tnas(n) : kompleksitas waktu untuk kasus terburuk (***worst case***)

merupakan kebutuhan waktu maksimum yang diperlukan algoritma sebagai fungsi dari n.

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulan x1, x2, … xn yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata- rata dari algoritma pencarian beruntun (*sequential search*). Algoritma *sequential search* berikut

menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

procedure SequentialSearch(input x1, x2, … xn : integer, y : integer, output idx : integer)

{ Mencari y di dalam elemen x1, x2, … xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.

Jika y tidak ditemukan, makai idx diisi dengan 0. Input: x1, x2, … xn

Output: idx

}

**Deklarasi**

i : integer

found : boolean {bernilai true jika y ditemukan atau false jika y tidak ditemukan}

**Algoritma**

i  1

found  false

while (i ≤ n) and (not found) do if xi = y then

found  true else

i  i + 1 endif

endwhile

{*i < n or found*}

If found then {*y ditemukan*}

idx  i

else

idx  0 {y tidak ditemukan}

endif

Jawaban Studi Kasus 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Best** | **Avarage** | **Worst** |
| **Assignment** |  |  |  |
| Baris 1 | 1 | 1 | 1 |
| Baris 2 | 1 | 1 | 1 |
| Baris 5 | 1 | 1 | found: 1  !found: n |
| Baris 7 |  | (n+1)/2 | Found: (n-1)  !found: n |
| Baris 12 | 1 | 1 | found: 1  !found: n |
| Baris 14 |  |  | Found: 0  !found: 1 |
| **Perbandingan** |  |  |  |
| Baris 4 | 1 | (n+1)/2 | N |
| **Penjumlahan** |  |  |  |
| Baris 7 |  | (n+1)/2 | Found: (n-1)  !found: n |
|  |  |  |  |
| T(n) | 5 | 4 + (n+1) + (n+1)/2 | Found: 4 + n + 2(n+1)  !found: 3+ 3n |

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulan x1, x2, … xn yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata- rata dari algoritma pencarian bagi dua (*binary search*). Algoritma *binary search* berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

if xmid < y then

i  mid + 1 else

j  mid – 1 endif

endif endwhile

{*found or i > j* }

{*mencari di bagian kanan*}

{*mencari di bagian kiri*}

If found then

Idx  mid

else

Idx  0

endif

Jawaban Studi Kasus 3

procedure BinarySearch(input x1, x2, … xn : integer, x : integer, output : idx : integer)

{ Mencari y di dalam elemen x1, x2, … xn. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.

Jika y tidak ditemukan makai dx diisi dengan 0.

**Input:** x1, x2, … xn

**Output: idx**

}

**Deklarasi**

i, j, mid : integer found : Boolean

**Algoritma**

i  1 j  n

found  false

while (not found) and ( i ≤ j) do mid  (i + j) div 2

if xmid = y then found  true

else

Jawaban Studi Kasus 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Best** | **Avarage** | **Worst** | **Operasi** |
| Baris 1 | 1 | 1 | 1 | Assignment |
| Baris 2 | 1 | 1 | 1 | Assignment |
| Baris 3 | 1 | 1 | 1 | Assignment |
| Baris 4 |  |  |  |  |
| Baris 5 | 1  1  1  **3** | (n+1)/2  (n+1)/2  (n+1)/2  **3(n+1)/2** | N  N  N  **3n** | Assignment  Penjumlahan  Pembagian |
| Baris 6 |  |  |  |  |
| Baris 7 | 1 | (n+1)/2 | Found: 1  !found: 0 | Assignment |
| Baris 8 |  |  |  |  |
| Baris 9 |  |  |  |  |
| Baris 10 |  | If: n/4  If: n/4  **n/2** | If:  Found: 2(n-1)  Else: 2n | Assignment  Penjumlahan |
| Baris 11 |  |  |  |  |
| Baris 12 |  | Else: n/4  Else: n/4  **n/2** | Else:  Found: 2(n-1)  Else: 2n | Assignment  pengurangan |
| Baris 13 |  |  |  |  |
| Baris 14 |  |  |  |  |
| Baris 15 |  |  |  |  |
| Baris 16 |  |  |  |  |
| Baris 17 |  |  |  |  |
| Baris 18 |  |  |  |  |
| Baris 19 | 1 | 1 | Found: 1 | assignment |
| Baris 20 |  |  |  |  |
| Baris 21 |  |  |  |  |
| Baris 22 |  |  | !found: 1 | assignment |
| T(n) | 8 | 3 + 3(n+1)/2 + (n+1)/2 + n/2  **4 + n/2 + 4(n+1)/2** | Found: 3 + 5n  !found: 4 + 5n |  |

# Studi Kasus 4: Insertion Sort

1. Buatlah program insertion sort dengan menggunakan bahasa C++
2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort.
3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

procedure InsertionSort(input/output x1, x2, … xn : integer)

{ Mengurutkan elemen-elemen x1, x2, … xn dengan metode insertion sort.

Input: x1, x2, … xn

OutputL x1, x2, … xn (sudah terurut menaik)

}

**Deklarasi**

i, j, insert : integer

**Algoritma**

for i  2 to n do

insert  xi j  i

while (j < i) and (x[j-i] > insert) do x[j] x[j-1]

jj-1 endwhile x[j] = insert

endfor

Jawaban Studi Kasus 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Best** | **Avarage** | **Worst** | **Operasi** |
| Baris 1 |  |  |  |  |
| Baris 2 | n-1 | n-1 | n-1 | Assignment |
| Baris 3 | n-1 | n-1 | n-1 | Assignment |
| Baris 4 |  |  |  |  |
| Baris 5 | n-1 | (((n+1)/2)(n-1)) | n2 - n | Assignment |
| Baris 6 | n-1 | (((n+1)/2)(n-1)) | n2 - n | Assignment |
| Baris 7 |  |  |  |  |
| Baris 8 | n-1 | n-1 |  | Perbandingan |
| Baris 9 |  |  |  |  |
| T(n) | 5n - 5 | **3n – 3 + 2(((n+1)/2)(n-1))** | 2n2 + n - 3 |  |

# Studi Kasus 5: Selection Sort

1. Buatlah program selection sort dengan menggunakan bahasa C++
2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma selection sort.
3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

procedure SelectionSort(input/output x1, x2, … xn : integer)

{ Mengurutkan elemen-elemen x1, x2, … xn dengan metode selection sort.

Input: x1, x2, … xn

OutputL x1, x2, … xn (sudah terurut menaik)

}

**Deklarasi**

i, j, imaks, temp : integer

**Algoritma**

for i  n downto 2 do {*pass sebanyak n-1 kali*} imaks  1

for j  2 to i do

if xj > ximaks then imaks  j

endif endfor

{pertukarkan ximaks dengan xi} temp  xi

xi  ximaks ximaks  temp

endfor

Jawaban Studi Kasus 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Best** | **Avarage** | **Worst** | **Operasi** |
| Baris 1 |  |  |  |  |
| Baris 2 | n-1 | n-1 | n-1 | Assignment |
| Baris 3 |  |  |  |  |
| Baris 4 |  |  |  |  |
| Baris 5 | n-1 | (((n+1)/2)(n-1)) | n2-n | assigment |
| Baris 6 |  |  |  |  |
| Baris 7 |  |  |  |  |
| Baris 8 |  |  |  |  |
| Baris 9 | n-1 | n-1 | n-1 | Assigment |
| Baris 10 | n-1 | n-1 | n-1 | Assigment |
| Baris 11 | n-1 | n-1 | n-1 | assigment |
| Baris 12 |  |  |  |  |
| T(n) | 5n - 5 | **4n – 4 + (((n+1)/2)(n-1))** | n2 + 3n – 4 |  |

## Teknik Pengumpulan

* Lakukan push ke github/gitlab untuk semua program dan laporan hasil analisa yang berisi jawaban dari pertanyaan-pertanyaan yang diajukan. Silahkan sepakati dengan asisten praktikum.

## Penutup

* Ingat, berdasarkan Peraturan Rektor No 46 Tahun 2016 tentang Penyelenggaraan Pendidikan, mahasiswa wajib mengikuti praktikum 100%
* Apabila tidak hadir pada salah satu kegiatan praktikum segeralah minta tugas pengganti ke asisten praktikum
* Kurangnya kehadiran Anda di praktikum, memungkinkan nilai praktikum Anda tidak akan dimasukkan ke nilai mata kuliah.