Analisa Efisiensi Algoritma Non-Rekursif

Elsa Elvira Awal, M.Kom

Preview - Rekursif vs Non-Rekursif

```
while condition do
    solve it
end while
return solution
```

```
if base case
        solve it
else
        redefine the problem
        using recusion case
```

Manakah yang menggunakan algoritma rekursif?

- Tujuan recusion case memecah suatu permasalahan yang kompleks, sehingga dapat ditemukan bentuk base case.
- Non-rekusrif biasa disebut dengan algoritma iterative (perulangan).

Preview - Rekursif vs Non-Rekursif

```
def  factorial(number):
      product = 1
      for i in range(number):
            product = product * (i + 1)
      return product
```

Factorial

i	product*(i+1)	product
0	1*(0+1)	1
1	1*(0+1)	2
2	2*(0+1)	6

factorial (3) ... 6
3 * factorial (2)
2 * factorial (1)

```
def factorial(number):
      if number <= 1: #base case
            return 1
      else
            return number*factorial(number-1)
```

Analisis Algoritma

- Analisis algoritma bertujuan memeriksa efisiensi algoritma dari dua segi: waktu eksekusi dan penggunaan memori.

- Efisiensi waktu seberapa cepat algoritma dieksekusi
- Efisiensi memori berapa banyak memori yang dibutuhkan untuk menjalankan algoritma

- Untuk melakukan analisa efisiensi waktu algoritma harus diestimasi dulu waktu eksekusi algoritma.

Jelaskan Algoritma Berikut

Algoritma Sequential Search $(A[0 \ldots n-1], K)$

// Mencari nilai tertentu dalam array dengan algoritma Sequential Search

// Input: Array $A[0 \ldots n-1]$ dan search key K

// Output: Indeks elemen pertama A yang cocok dengan K

// atau -1 jika tidak ada elemen yang cocok

$i \leftarrow 0$

while $i < n$ **and** $A[i] \neq K$ **do**

$i \leftarrow i + 1$

if $i < n$ **return** i

else return -1

Sequential Search

$i \leftarrow 0$	1x
while $i < n$ **and** $A[i] \neq K$ **do**	2x
$i \leftarrow i + 1$	1x
if $i < n$ **return** i	2x
else return -1	1x

time = nLoop x tLoop

- time: estimasi waktu esksekusi algoritma untuk input tertentu.
- nLoop: berapa kali loop dieksekusi
- tLoop: waktu yang diperlukan untuk mengeksekusi loop 1 kali. Biasanya ditentukan 1 satuan waktu tanpa dispesifikasikan berapa nilainya.

- Asumsikan array A terdiri atas n elemen.

- Best case: k ditemukan di elemen pertama array A. $time = 1\ x\ 1$ satuan waktu.
- Worst case: k ditemukan di elemen paling akhir array A. $time = n\ x\ 1$ satuan waktu.
- Average case: k ditemukan diposisi mana saja dengan probabilitas sama.

Langkah-langkah Umum untuk Menganalisa Efisiensi Waktu Algoritma Nonrekursif

- Tentukkan parameter yang mengindikasikan ukuran input.
- Identifikasi basic operation algoritma.
- Tentukan apakah untuk ukuran input yang sama, banyaknya eksekusi basic operation bisa berbeda.
- Tentukan rumus deret yang menunjukkan berapa kali basic operation dieksekusi.
- Selesaikan rumus deret untuk menghitung banyaknya ekseskusi basic operation.

Step 1: Tentukan Parameter yang Mengindikasikan Ukuran Input

- Sesuatu pada input yang jika nilainya bertambah akan menyebabkan banyaknya eksekusi loop bertambah
- Contoh, algoritma untuk menghitung X^n menggunakan cara $X^n = X \times X \times X \times \cdots \times X$ sebanyak n kali. Parameter ukuran inputnya adalah nilai n, karena jika n makin besar, maka banyaknya eksekusi loop bertambah.
- Bagaimana dengan nilai X?
- Untuk algoritma sequential search, parameter ukuran inputnya adalah banyaknya elemen array (n)

Step 2: Identifikasi Basic Operation Algoritma

- Waktu yang diperlukan untuk mengeksekusi loop 1 kali
- Dapat diwakili oleh sebuah operasi pada loop paling dalam.
- Operasi yang dipilih adalah operasi yang selalu dilakukan ketika loop dieksekusi
- Untuk algoritma sequential search, basic operationnya dapat digunakan $i < n$
- $i < n$ dieksekusi 1 kali setiap loop dieksekusi.

Step 3: Tentukan Apakah untuk Ukuran Input yang Sama, Banyaknya Eksekusi Basic Operation Bisa Berbeda

- Pada sequential search, parameter untuk ukuran input adalah n atau banyaknya elemen array.
- Untuk n tertentu, apakah banyaknya eksekusi basic operation/banyaknya loop bisa berbeda?
- Jika elemen pertama array input A bernilai K, maka banyaknya eksekusi basic operation untuk n tertentu $C(n) = 1$. Nilai K bisa ditemukan di mana saja pada array.
- Jika K ditemukan di elemen terakhir, maka $C(n) = n$
- Perlu diadakan analisa best case, worst case, dan average case.

Step 4: Tentukan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- C(n) = banyaknya ekseskusi basic operation untuk input ukuran n
- Untuk sequential search best casenya

$$C(n) = \sum_{i=1}^{1} 1$$

- Best case terjadi jika elemen pertama A bernilai K

Step 4: Tentukan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Untuk sequential search worst casenya

$$C(n) = \sum_{i=1}^{n} 1$$

- Worst case terjadi jika elemen A yang bernilai K merupakan elemen terakhir atau tidak ada elemen yang bernilai K.

Step 4: Tentukan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Averrage case pada sequential search

- Asumsikan
  - Data K memang ada di A
  - Probabilitas K terletak di elemen tertentu A terdistribusi merata.
  - Probabilitas K terletak di elemen ke $i = \frac{1}{n}$

Step 4: Tentukan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Bentuk umum : $i * \frac{1}{n}$

Posisi K ditemukan	Banyaknya eksekusi basic opration	Probabilitas terjadi	Kontribusi pada C(n)
1	1	1/n	1*1/n
2	2	1/n	2*1/n
…	…	…	… * …
…	…	…	… * …
n	n	n	n*n

Step 4: Tentukan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Average case pada sequential search

$$C(n) = \sum_{i=1}^{n} i \times \frac{1}{n}$$

Step 5: Selesaikan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Best case untuk sequential search

$$C(n) = \sum_{i=1}^{1} 1$$

$$C(n) = 1$$

- Best case pada sequential search $C(n) = 1$
- Untuk input berukuran n, basic operation dilakukan 1 kali

Step 5: Selesaikan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Worst case untuk sequential search

$$C(n) = \sum_{i=1}^{n} 1$$

$$C(n) = n$$

- Worst case pada sequential search $C(n) = n$
- Untuk input berukuran n, basic operation dilakukan n kali

Step 5: Selesaikan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Average case pada sequential search

$$C(n) = \sum_{i=1}^{n} i \frac{1}{n}$$

$$C(n) = \frac{1}{n} \sum_{i=1}^{n} i$$

$$C(n) = \frac{1}{n} \times \frac{1}{2} n(1+n) = \frac{1}{2}(1+n)$$

Step 5: Selesaikan Rumus Deret yang Menunjukkan Berapa Kali Basic Operation Dieksekusi

- Pada sequential search, average casenya

$$C(n) = \frac{(n+1)}{2}$$

- Untuk $n = 10, C(n) = 5{,}5$

Estimasi Waktu Running Algoritma Sequential Search

$$T(n) = C_{op} \times C(n)$$

- $T(n)$: waktu yang diperlukan untuk mengeksekusi algoritma dengan input berukuran n
- C_{op}: waktu untuk mengeksekusi basic operation 1 kali. Biasanya ditentukan 1 satuan waktu.

Thank You