**LAPORAN**

**PRAKTIKUM ANALISIS ALGORITMA**

**UTS**



**Disusun Oleh:**

Dimas Satria Prakoso (140810170007)

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**SUMEDANG**

**2019**

**LAPORAN**

**PRAKTIKUM ANALISIS ALGORITMA**

**UTS**

1. Cari masing-masing T(n), Komplesitas O, c dan n0!
2. Algoritma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Algoritma | Operasi | T(n) | Kompleksitas O | Keterangan |
| 1 | For j = 1 to n-1 | kalang |  |  |  |
| 2 | k = j | pengisian nilai | = 1 | 0(1) |  |
| 3 | For i = j + 1 to n | Kalang |  |  |  |
| 4 | if a[i] < a[k] then | perbandingan | = 1+ n(n-1)/2 | O(n2) |  |
| 5 | k=i | pengisian nilai | = 1 + 2n(n-1)/2 |  |  |
| 6 | endif |  | = 1 + n(n-1) |  |  |
| 7 | endfor |  |  |  |  |
| 8 | tm=a[j] | pengisian nilai | = 1 + n(n-1) + 1 |  |  |
| 9 | a[j] = a[k] | pengisian nilai | = 2 + n(n-1) + 1 |  |  |
| 10 | a[k] = tm | pengisian nilai | = 3 + n(n-1) + 1 |  |  |
| 11 | endfor |  | = 4(n-1)+ n(n-1)  = n2 + 3n - 4 | O(n2) |  |

Dengan n2 ≥ n dan n2 ≥ 1, maka T(n) = n2 + 3n – 4 dan n2 + 3n – 4 ≤ n2 + 3n2 - 4n2 jadi c = 1.5 dan n0 = 2. Kompleksitas O = O(n2)

1. Algoritma

|  |  |
| --- | --- |
| No | Algoritma |
| 1 | for i=0 to n-1 |
| 2 | for j=0 to n-1 |
| 3 | c[i,j] = 0 |
| 4 | for k=0 to n-1 |
| 5 | cij= d[i,k] and b[k,j] |
| 6 | c[i,j] = c[i,j] or cij |
| 7 | endfor |
| 8 | endfor |
| 9 | endfor |

Menghitung T(n). Kita mulai dari kalang terdalam

* Baris 5 dan 6 -> 2 operasi. T(n) = 2
* Baris 4 -> kalang. T(n) = 2n
* Baris 3 -> 1 operasi. T(n) = 1 + 2n
* Baris 2 -> kalang. T(n) = 2n2 + n
* Baris 1 -> kalang. T(n) = 2n3 + n2

Jadi T(n) = 2n3 + n2. Dengan n3 ≥ n2, maka 2n3 + n2 ≤ 2n3 + n3 = 3n3. Jadi c = 3 dan n0 = 1. Kompleksitas O = O(n3)

1. A. Cari masing-masing T(n), Komplesitas O, c dan n0!

Algoritma dibawah merupakan algoritma linear search. Dalam penghitungan komplesitas dibawah ini anggap kasus terburuk dimana nilai yang dicari tidak ada didalam list, maka baris code if tidak dijalankan.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Algoritma | Operasi | T(n) | Kompleksitas O | Keterangan |
| 1 | Ada = 0; | Pengisian nilai | = 1 | = O(1) |  |
| 2 | Kx=1; | Pengisian nilai | = 2 | = O(1)+O(1)  = O(1) |  |
| 3 | Input br; | Pengisian nilai | = 3 | = O(1) |  |
| 4 | For (i=1; i<n+1; i++){ | Kalang | = 3 + n | = O(n) |  |
| 5 | If(a[i] == br && (!ada)){ |  |  |  |  |
| 6 | Ada = 1; |  |  |  |  |
| 7 | kx = 1; |  |  |  |  |
| 8 | i = n+1; |  |  |  |  |
| 9 | } |  |  |  |  |
| 10 | } |  |  |  |  |

T(n) = n + 3. Dengan n ≥ 1, maka n + 3 ≤ n + 3n= 4n. jadi c = 4 dan n0 = 1. Kompleksitas O = O(n)

Algoritma dibawah merupakan algoritma binary search. Dalam penghitungan komplesitas dibawah ini anggap kasus terburuk dimana nilai yang dicari tidak ada didalam list dan lebih besar dari semua nilai didalam list, maka hanya baris 6, 7, 12 yang dijalanan pada kalang *while*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Algoritma | Operasi | T(n) | Kompleksitas O | Keterangan |
| 1 | L = 1; | Pengisian nilai | = 1 | = O(1) |  |
| 2 | R=n; | Pengisian nilai | = 2 | = O(1)+O(1)  = O(1) |  |
| 3 | Ada = 0; | Pengisian nilai | = 3 | = O(1) |  |
| 4 | Input br; | Pengisian nilai | = 4 | = O(1) |  |
| 5 | While((L<= R) && (!ada)){ | Kalang | = 4 + | = O() |  |
| 6 | M = (L+R) div 2; | Pembagian | = 4 + (1) | = O() |  |
| 7 | If (a[m] == br) | Perbandingan | = 4 + (2) | = O() |  |
| 8 | Ada = 1; |  |  |  |  |
| 9 | Else if (br<a[m]) | perbandingan | = 4 + (3) | = O() |  |
| 10 | R = m+1; |  |  |  |  |
| 11 | Else |  |  |  |  |
| 12 | L=m+1; | Pengisian nilai | = 4 + (4) | = O() |  |
| 13 | } |  |  |  |  |

Pada kalang *while* binary search, dilakukan sebanyak

Ukuran larik setiap memasuki kalang *while* ialah n, n/2, n/4, … , 1 maka

2k = n

k =

Jadi T(n) = 4 + 4. Dengan n0 = 2 dan ≥ 1, 4 + 4 ≤ 4 + 4 = 8 maka c = 8. Kompleksitas O = O()

1. Running Time

Komputer A dengan Algoritma A dan kecepatan 10-9 detik/instruksi. Jika n = 108 maka

running time = O(108) x 10-9 detik/instruksi = 108 x 10-9 = 0.1 detik

Komputer B dengan Algoritma A dan kecepatan 10-7 detik/instruksi. Jika n = 108 maka

running time = O()) x 10-7 detik/instruksi = 26.578x 10-7 = 2.6578x 10-6 detik

Terbukti **Algoritma B jauh lebih baik** dan dengan simulasi diatas kita bisa menyimpulkan bahwa perancangan Algoritma yang baik sangat penting dan berpengaruh pada kecepatan suatu program