LATIHAN MENGANALISI ALGORITMA

1. Perhatikan pseudocode berikut:

|  |
| --- |
| Function **tileWithTrominoes**(n, m)  if n == 2  return  tileWithTrominoes (n/2, m1)  tileWithTrominoes (n/2, m2)  tileWithTrominoes (n/2, m3)  tileWithTrominoes (n/2, m4) |

1. Nyatakan algoritma tersebut ke dalam recurrence relation!

T(n)=4T(n/2​)+O(1)

1. Hitunglah running time dari algoritma tersebut!

T(n)=aT(n/b​)+f(n)

di mana a=4a = 4a=4, b=2b = 2b=2, dan f(n)=O(1)f(n) = O(1)f(n)=O(1).  
a=4  
b=2b = 2b=2  
logb a = log2 4 =2   
f(n)=O(1), di mana O(1)=O(n0)O(1) = O(n^0)O(1)=O(n0) dan 0<20 < 20<2.  
di mana c<logb a. Jadi, T(n) = Θ(n logb a) = Θ(n2)).

1. Nyatakan running time algoritma tersebut ke dalam notasi asimtotic!

T(n) = Θ(n2)).

1. Perhatikan pseudocode berikut:

|  |
| --- |
| **Alg(n)**   1. **for** p 🡨 2 **to** n **do** A[p] 🡨 p 2. **for** p 🡨 2 **to**  **do** 3. **if** A[p]  0 4. j 🡨 p \* p 6. i 🡨 0 7. **for** p 🡨 2 **to** n **do** 8. **if** A[p]0 9. L[i] 🡨 A[p] 10. i 🡨 i + 1 11. **return** L |

1. Tentukan basic operation dari algoritma tersebut!

* A[p] <- p
* if A[p] ≠ 0
* A[j] <- 0
* j <- j + p
* L[i] <- A[p]
* i <- 0  
  i <- i + 1

1. Hitunglah running time dari algoritma tersebut dengan pohon rekursi!

for p <- 2 to n do A[p] <- p

* Loop ini berjalan dari 2 hingga n, sehingga kompleksitasnya adalah O(n).

for p <- 2 to sqrt(n) do

if A[p] ≠ 0

j <- p \* p

while j ≤ n do

A[j] <- 0

j <- j + p

* Loop luar berjalan dari 2 hingga \sqrt{n}, jadi kompleksitasnya adalah
* Di dalam loop, jika A[p] ≠ 0, kita memiliki loop while yang berjalan dengan langkah p dari p\*p hingga n. Pada kasus terburuk, loop ini akan berjalan sekitar kali.
* Jadi, total kompleksitas untuk bagian ini adalah (dengan menggunakan analisis Sieve of Eratosthenes).

i <- 0

for p <- 2 to n do

if A[p] ≠ 0

L[i] <- A[p]

i <- i + 1

* Loop ini berjalan dari 2 hingga n, jadi kompleksitasnya adalah O(n).

1. Nyatakan running time algoritma tersebut ke dalam notasi asimtotic!

Dari analisis di atas, kita memiliki dua bagian utama dari algoritma:

1. Inisialisasi array A dengan kompleksitas O(n).
2. Menandai bilangan non-prime dengan kompleksitas .
3. Mengumpulkan bilangan prime ke dalam list L dengan kompleksitas O(n).

Karena mendominasi O(n), kompleksitas waktu asimptotik dari algoritma ini adalah:

T(n)=

1. Perhatikan pseudocode di bawah ini :

|  |
| --- |
| **Algoritma** **ABC** (a,n)  p = 1  i = n  **while** (i>0) do  p = p\*a  i = i+1  **end while** |

1. Untuk apakah algoritma tersebut !

Algoritma ini tampaknya ditujukan untuk menghitung nilai dari a^n, yaitu menghitung hasil dari aaa dipangkatkan dengan nnn.

1. Tentukan loop invarian dari pseudocode tersebut!

Sebelum setiap iterasi loop, nilai p sama dengan a dipangkatkan dengan nilai n− i.

1. Analisalah kebenaran dari algoritma tersebut, apakah benar atau tidak dan beri alasan dari jawaban anda!

Algoritma tersebut benar dalam konteks menghitung a^n, dengan menjaga invariant loop yang sesuai dan berakhir ketika seharusnya.

1. Perhatikan pseudocode berikut:

|  |
| --- |
| int maxCrossingSum(int arr[], int l, int m, int h)  {          int sum = 0;      int left\_sum = INT\_MIN;      for (int i = m; i >= l; i--)      {          sum = sum + arr[i];          if (sum > left\_sum)            left\_sum = sum;      }       sum = 0;      int right\_sum = INT\_MIN;      for (int i = m+1; i <= h; i++)      {          sum = sum + arr[i];          if (sum > right\_sum)            right\_sum = sum;      }        return left\_sum + right\_sum;  }  int maxSubArraySum(int arr[], int l, int h)  {     if (l == h)       return arr[l];     int m = (l + h)/3;       return max(maxSubArraySum(arr, l, m),                maxSubArraySum(arr, m+1, 2m),  maxSubArraySum(arr, 2m+1, h),                maxCrossingSum(arr, l, m, h));  } |

1. Tentukan recurrence dari algoritma tersebut!
2. Tentukan running time recurrence tersebut dengan master theorema!

Untuk rekurens :

 Jika maka

 Jika maka

 Jika maka

Di sini, a=3, b=2, dan d=1 (karena O(n) dari maxCrossingSum).

* .
* Karena d=1 dan (karena ), maka kita berada pada kasus kedua dari Master Theorem.

1. Nyatakan running time yang anda dapatkan pada soal b dalam bentuk notasi asimtotic!