Soal Divide and Conquer

Sebuah larik A berisi deretan bilangan integer yang tidak terurut. Tuliskan algoritma dengan pendekatan **Divide and Conquer** dalam bentuk rekursif untuk mencari banyaknya elemen yang memiliki rentang nilai antara (dan termasuk) min dan mak. Sebagai contoh, jika A = 10, 29, 89, 50, 34, 91, 39, 66, 20 dengan nilai yang dicari dalam rentang antara min=20 dan mak=40, maka banyak elemen dengan rentang tersebut adalah 4 (empat). Lalu tuliskan analisis algoritma untuk mendapatkan kompleksitas dari perbandingan elemen yang dilakukan.

Untuk algoritma, gunakan nama dan urutan parameter masukan berikut:

CountRange (**Input** *A*: larik integer;
low, *high*: integer; // batas indeks bawah dan atas larik A.
min, *mak*: integer). // batas bawah dan atas rentang nilai yang dihitung.

Output CountRange adalah banyaknya elemen antara (dan termasuk) min dan mak.

Dalam penulisan untuk penuruan kompleksitas gunakan symbol '^' untuk menyatakan pangkat, sebagai contoh: "n pangkat 2" ditulis dengan "n^2".

Jawaban

Algoritma:

```
if low = high then
    if A[low] ≥ min AND A[low] ≤ mak then
        Return 1
    else
        Return 0
    endif
else
    mid = (low + high) div 2
    Return CountRange (A, low, mid, min, mak) + CountRange (A, mid+1, high, min, mak)
Endif
```

Analisis kompleksitas:

$$T(n) = \begin{cases} 2 & ,n = 1 \\ 2\,T(\frac{n}{2}) & ,n > 1 \end{cases}$$

$$T(n) = 2\,T\left(\frac{n}{2}\right)$$

Missal $n = 2^k$

$$T(n) = 2T\left(\frac{2^k}{2}\right)$$

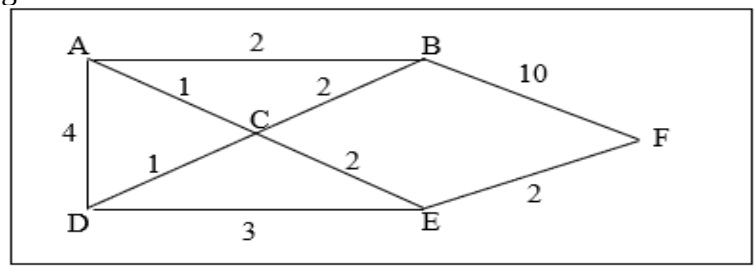
$$= 2\ T(2^{k-1}) = \ 2^2\ T(2^{k-2}) = \ 2^3\ T(2^{k-3})$$
$$= 2^i\ T\left(2^{k-i}\right)$$

Saat i = k, maka

$$T(n) = 2^k\ T(2^0) = n\ T(1) = 2\,n = O(n)$$

Soal BFS/DFS

Terdapat sebuah graf tidak berarah berikut ini. Simpul merepresentasikan kota, dan bilangan yang terdapat pada sisi adalah jarak antara dua kota.
Persoalan yang ingin diselesaikan adalah mencari jalur **dari kota A ke kota F, dan diharapkan dapat memberikan jalur dengan jarak total minimum**.
(i) Jika digunakan pendekatan Brute Force, tuliskan langkah penyelesaiannya secara garis besar, kemudian tuliskan jalur hasil penyelesaiannya serta jarak total jalur hasil tersebut.
(ii) Jika digunakan pendekatan BFS, tuliskan proses pencarian dengan cara seperti pada gambar. (iii) Jika digunakan pendekatan DFS, tuliskan proses pencarian dengan cara seperti pada gambar. Catatan: urutan prioritas simpul sesuai dengan urutan abjad. Perhatikan urutan penulisan simpul hidup jika terdapat simpul baru yang ditambahkan pada daftar simpul hidup, sesuai dengan teknik BFS atau DFS.



Gambar 1.

BFS
Iterasi 1:
Simpul Ekspan (Simpul yang diperiksa) = A
Simpul Hidup = B[A] C[A] D[A] (catatan: C[AB] artinya simpul C dari jalur AB)
Iterasi 2:
Simpul Ekspan (Simpul yang diperiksa) = [tuliskan simpul yang diperiksa berikutnya]
Simpul Hidup = [tuliskan semua simpul tetangga termasuk dari iterasi sebelumnya yang belum diperiksa]
Dst…. (Catatan: pencarian dihentikan ketika simpul yang diperiksa = F)
Jalur hasil pencarian dengan BFS =
Jarak jalur hasil pencarian dengan BFS=

DFS
Iterasi 1:
Simpul Ekspan (Simpul yang diperiksa) = A
Simpul Hidup = B[A] C[A] D[A] (catatan: C[AB] artinya simpul C dari jalur AB)
Iterasi 2:
Simpul Ekspan (Simpul yang diperiksa) = [tuliskan simpul yang diperiksa berikutnya]
Simpul Hidup = [tuliskan semua simpul tetangga termasuk dari iterasi sebelumnya yang belum diperiksa]
Dst…. (Catatan: pencarian dihentikan ketika simpul yang diperiksa = F)
Jalur hasil pencarian dengan DFS =
Jarak jalur hasil pencarian dengan DFS =

Solusi:
(i) Brute Force
Periksa semua kemungkinan jalur, kemudian pilih 1 jalur yang memberikan hasil jarak paling minimum. Hasilnya adalah: A-C-E-F dengan jarak 5. Nilai 5
Brute Force (Exhaustive Search) akan memeriksa semua kemungkinan, bukan permutasi dari simpul yang ada, karena mencari jalur dari A ke F tidak harus melewati seluruh simpul.

(ii) BFS
Ingat bahwa BFS memiliki prinsip serupa FIFO, penambahan simpul hidup dilakukan di akhir, dengan urutan sesuai urutan abjad (sesuai permintaan soal).
Iterasi 1:
Simpul Ekspan (Simpul yang diperiksa) = A
Simpul Hidup = B_A C_A D_A
Iterasi 2:
Simpul Ekspan (Simpul yang diperiksa) = B_A
Simpul Hidup = C_A D_A C_{BA} F_{BA}
Catatan: Ingat bahwa simpul C_A belum pernah diperiksa, oleh karena itu simpul C_{BA} tetap dimasukkan ke dalam simpul hidup. Simpul C_{BA} tidak akan dimasukkan ke simpul hidup jika C_A sudah pernah menjadi simpul ekspan sebelumnya.
Iterasi 3:
Simpul Ekspan (Simpul yang diperiksa) = C_A
Simpul Hidup = D_A C_{BA} F_{BA} D_{CA} E_{CA}
Iterasi 4:
Simpul Ekspan (Simpul yang diperiksa) = D_A
Simpul Hidup = C_{BA} F_{BA} D_{CA} E_{CA} E_{DA}
Iterasi 5:
Simpul Ekspan (Simpul yang diperiksa) = C_{BA}
Simpul Hidup = F_{BA} D_{CA} E_{CA} E_{DA} E_{CBA}
Iterasi 6:

Simpul Ekspan (Simpul yang diperiksa) = F_{BA}; sudah simpul tujuan, pencarian dihentikan.
Jalur hasil pencarian dengan BFS = A-B-F
Jarak jalur hasil pencarian dengan BFS= 12

(iii) DFS
Iterasi 1:
Simpul Ekspan (Simpul yang diperiksa) = A
Simpul Hidup = B_A C_A D_A
Iterasi 2:
Simpul Ekspan (Simpul yang diperiksa) = B_A
Simpul Hidup = C_{BA} F_{BA} C_A D_A
Iterasi 3:
Simpul Ekspan (Simpul yang diperiksa) = C_{BA}
Simpul Hidup = D_{CBA} E_{CBA} F_{BA} C_A D_A
Iterasi 4:
Simpul Ekspan (Simpul yang diperiksa) = D_{CBA}
Simpul Hidup = E_{DCBA} E_{CBA} F_{BA} C_A D_A
Iterasi 5:
Simpul Ekspan (Simpul yang diperiksa) = E_{DCBA}
Simpul Hidup = F_{EDCBA} E_{CBA} F_{BA} C_A D_A
Iterasi 6:
Simpul Ekspan (Simpul yang diperiksa) = F_{EDCBA}; sudah simpul tujuan, pencarian dihentikan.
Jalur hasil pencarian dengan DFS = A-B-C-D-E-F
Jarak jalur hasil pencarian dengan DFS= 10