Have You Made Full Use of the OCR Feature?

Make a scan,enhance it and save it.Are these all the features you know about CamScanner?If so,you have missed too many cool experiences. CamScanner offers you lots of features rather than scanning.What we are sharing today is the OCR(Optical Character Recognition)feature.

What can you do with OCR feature?

1.Searching

What can you do if you want to search for a document but just can't remember the names of some docs?Use this feature to recognize all the texts on your scans.Next time you just need to enter some key words in the search box and all the documents within the words will be found.

2.Text extraction

Just purchase the one-time paid version and you can enjoy the text extraction for lifetime!Ever want to edit some texts on a paper document or a PDF file?Import it into CamScanner and all texts can be extracted as.txt file after OCR!

Why wait? Follow the steps to start using OCR!

1.Sign in to CamScanner to sync all your docs->All texts will be auto recognized after syncing.

2.If you don't want to sign in,you can open one single page of any doc-> Tap the Recognize button->All recognized texts will be shown in a dialog box->Tap Share to export the texts.

Home Work 3. - Kerman de Azueta dos Santos Miranda

① Análisis de Algoritmos

N
```
public static int f1(int N){
    int x=0;
    for(int i=0; i<N; i++)
        x++;
    return x;
}
```

N³
```
public static int f2(int N){
    int x=0;
    for(int i=0; i<N; i++)
        for(int j=0; j<i; j++)
            x+=f1(j)
    return x;
}
```

N!
```
public static int f3(int N){
    if(N==0) return 1;
    int x=0;
    for(int i=0; i<N; i++)
        x+=f(3)(N-1);
    return x;
}
```

Mesh

```
public static int f4(int n){
    if(n==0) return 0;
    return f4(n/2) + f1(n) + f0(n) + f1(n) + f4(n/2)
```

3^n

```
public static int f3(int n){
    if(n==0) return 1;
    return f3(n-1) + f3(n-1) + f3(n-1)
```

$\log_2 n$

```
public static int f4(int n){
    int x=0;
    while(n>0){
        x++;
        n=n/2;
    }
    return x;
}
```

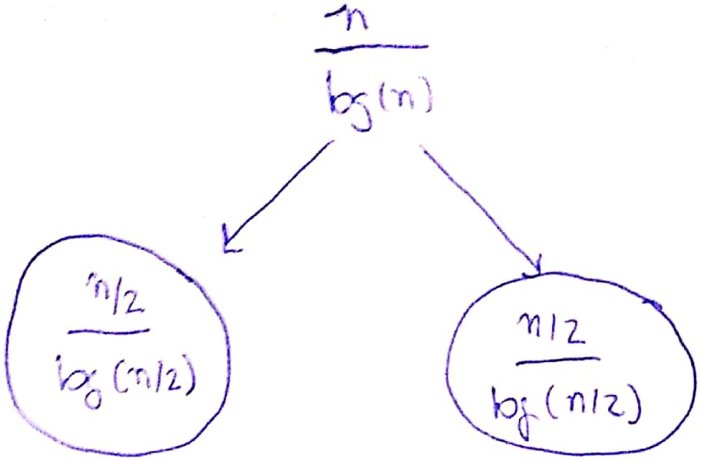
(2) Recursion

a) $T(n) = 2T(n/2) + n^k$, onde $k>0$ é uma constante.
[Use o Teorema Mestre e considere o caso dependendo de k].

$$T(n) \begin{cases} O(n^k \log n) & \text{se } 2 = 2^k \text{, ou seja, } k=1 \\ O(n^k) & \text{se } 2<2^k \text{ ou seja, } k>2. \\ O(n^{\log_b a}) & \text{se } 2>2^k \text{, ou seja, } k=0 \text{, o que é impossível} \end{cases}$$

b) $T(n) = 2T(n/2) + \frac{n}{\log n}$ [Desenhar a árvore de recursão]

Árvore de recursão.



Usando o Teorema Mestre, Temos:

Usando que $f(n) = \frac{n}{\log n}$ Temos que com $a=2, b=2$,

$\log_b a = \log_2 2 = 1$. Sabemos também que $\log n < n \ \forall n \in \mathbb{N}$.

Assim $\frac{n}{\log n} > 1$ e $n > \frac{n}{\log n}$. Assim, podemos concluir que:

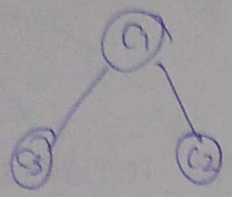
$\frac{n}{\log n} \geq c_1 \cdot n^{\varepsilon}$ ou $c_1 \cdot n \geq \frac{n}{\log n} \geq c_2 \cdot n$ ou $\frac{n}{\log n} \leq c_1 \cdot n^{-\varepsilon}$

Vega que de fato, como $c_1 \geq \frac{1}{\log n} \geq c_2$, pois $\frac{1}{\log n} \to +\infty$ quando $n \to \infty$, Temos $c_1 \geq 1$ e $c_2 < 0$. Assim, $f(n) = \Theta(n) \to$

$\to T(n) = \Theta(n \cdot \log n)$.

③ Binary search Tree.

Para três nós teremos 1 dado três chaves distintas, por tricotomia, 1 uma TQ.



$c_3 < c_1 < c_2$. Logo, c_1 deve ocupar a posição do topo. [ou seja, existe somente uma maneira de disposição válida].

Suponha que dado k chaves, nossa árvore é uma BST. Com k+1 chaves, vamos dividi-las em três grupos: O grupo x TQ $x < d_1$, com d_1 sendo uma chave, e o grupo $y > d_1$. Assim, a única chave que pode ocupar o Topo é d_1, pois [estamos assumindo que essa árvore tem "x" "y" elementos à esquerda e y à direita, em relação ao primeiro nó], pois se esse nó for diferente de d_1, suponha $d_2 < d_1$, então ~~sobrará~~ faltará um lugar do lado direito para d_1, uma vez que agora temos $x-1 < d_2$ e $y+1 > d_2$, mas somente y lugares à direita do primeiro nó. ■

④ Pesquisa em grafos:

a) M W Y I P S E B O [Pesquisa do Coelho] em ordem inversa.

Resolução

O B E S P I Y W M. usando DFS [busca profunda]

~~Coelho~~
Coelho

b) [redacted]

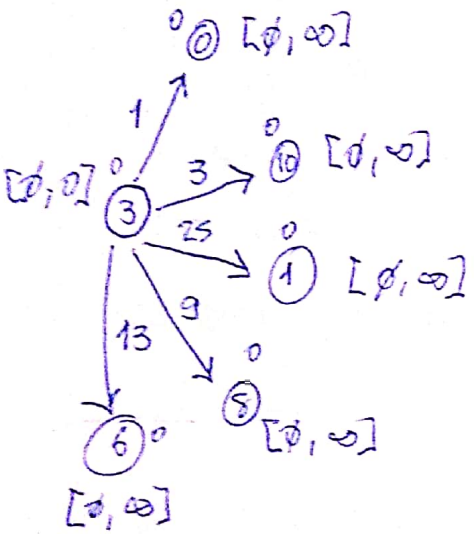
A = [[O], [B, E, Y], [S, W], [M, P, I]] usando BFS

[illegible]

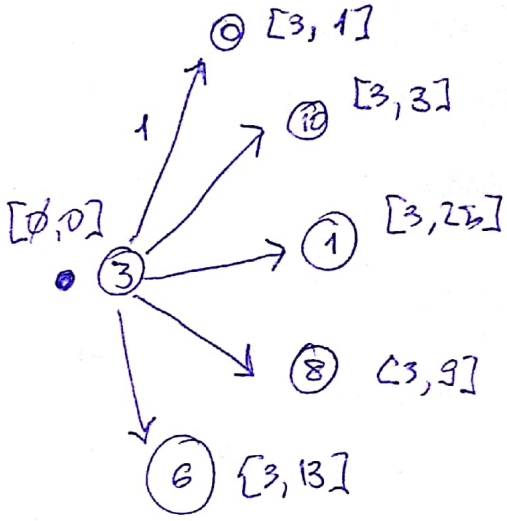
c) Não existe Topological sort do grafo dado, pois por definição, E deveria vir antes de S e S antes de E. Absurdo, Assim, não tem como existir esse Topological ~~order~~ sort.

(5) DIJKSTRA

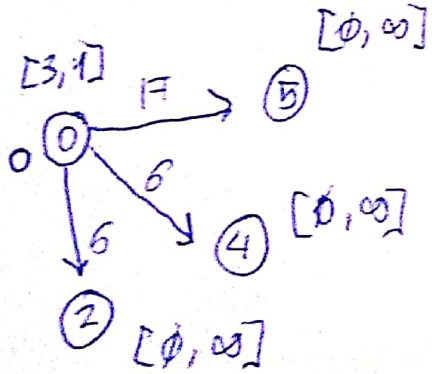
a) Queremos a ordem dos 5 primeiros caminhos. Note inicialmente que a lista é iniciada em 3, pois seu DISTTO[3] é zero e seu EDGETO[3] é null. Assim temos analisar primeiramente os vizinhos de 3.

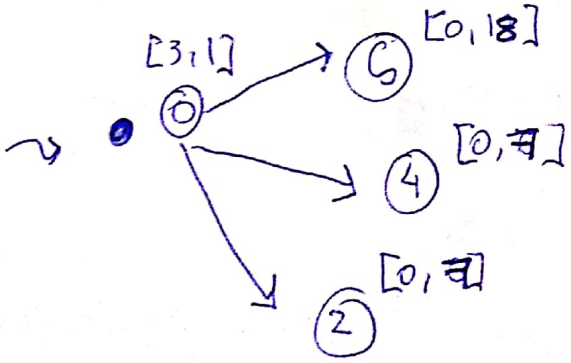


escolhendo o nó de menor distância e o relaxando

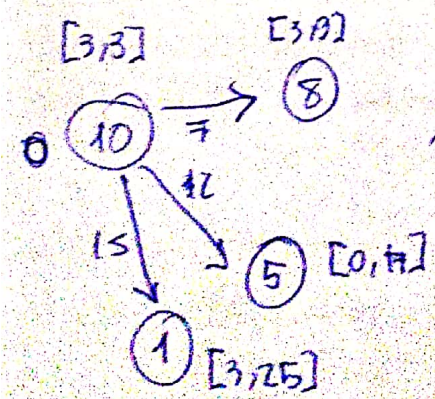


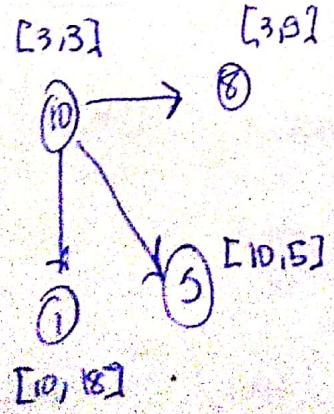
Agora devemos escolher o 0, devido a sua distância mínima em relação aos outros nós.



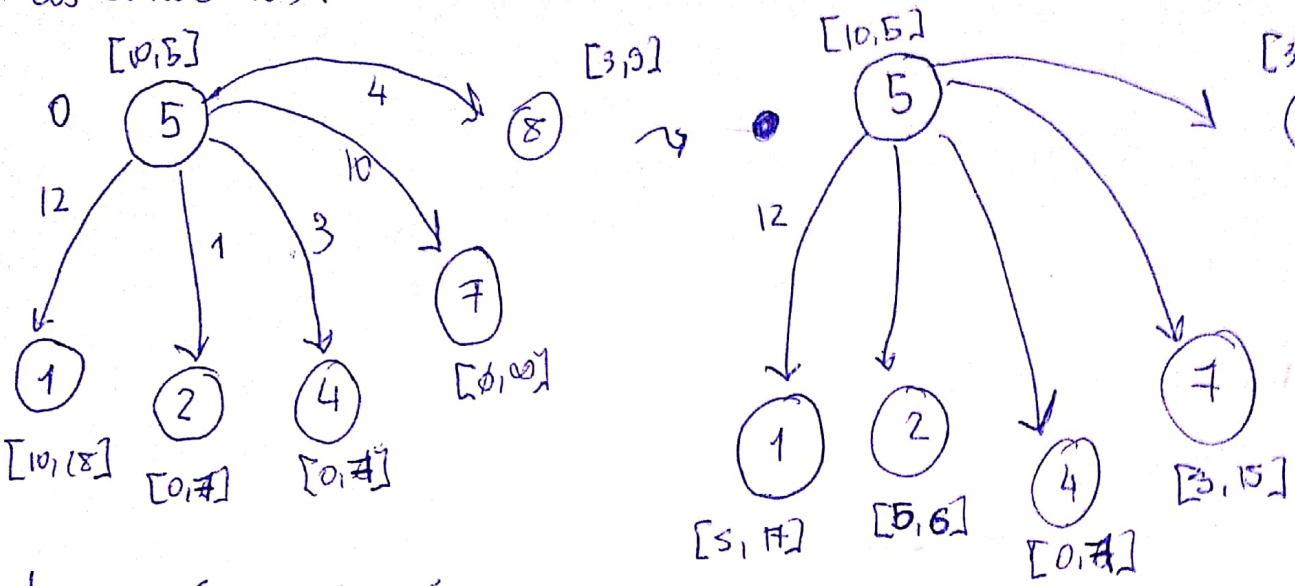


O próximo nó será o 10.





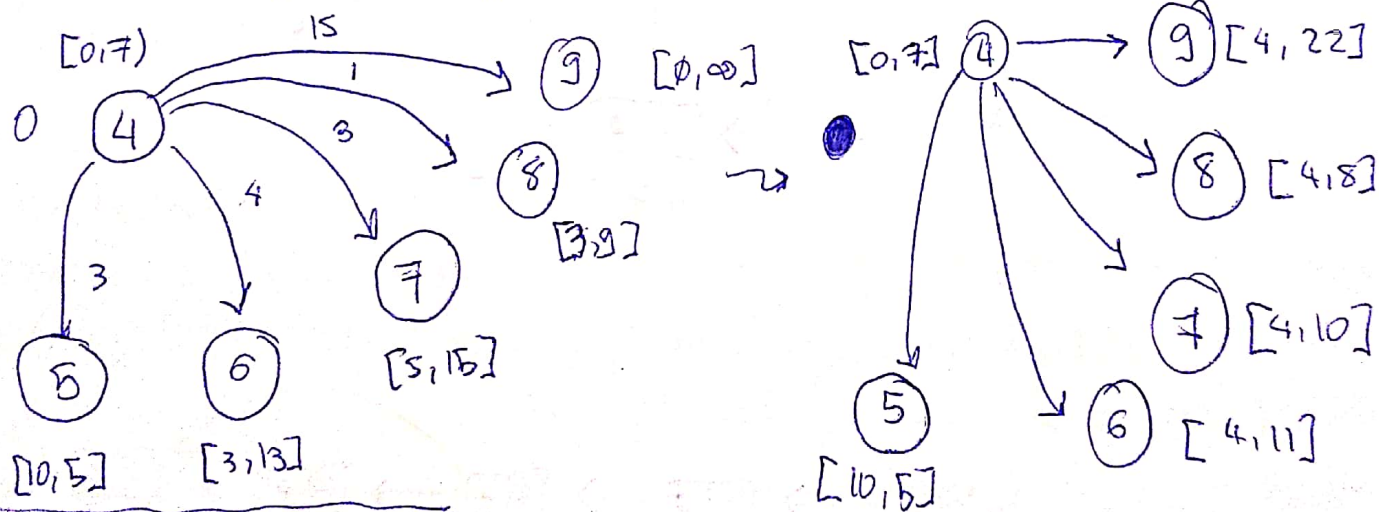
O próximo nó será o 5, devido sua distância mínima em relação aos outros nós.



logo, o próximo nó será o 2.

É assim, a sequência pedida é dada por $\boxed{3-0-10-5-2}$

b) Temos que identificar o próximo vértice a ser relaxado. Note que 4 deve ser esse número, uma vez que temo menor dist[], dentre os que não foram relaxados. Assim, vamos relaxar:



v	DISTO []	EDGTO
0	1	3→0
1	17	5→1
2	6	5→2
3	0	null
4	7	0→4
5	5	10→5
6	11	4→6
7	10	4→7
8	8	4→8
9	22	4→9
10	3	3→10

(7) Desenho de algoritmos

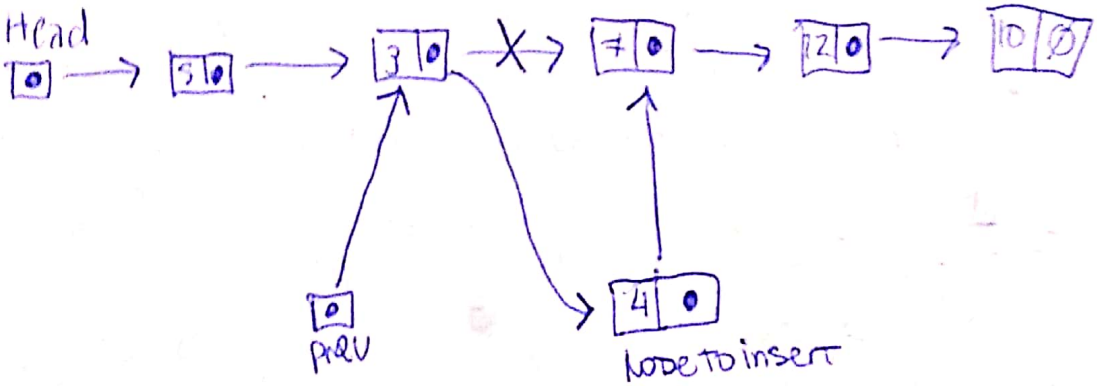
```
verifica(A)
    n = size(A)

    if A(⌊n/2⌋) = ⌊n/2⌋
        return A(⌊n/2⌋)

    else
        if A(⌊n/2⌋) < ⌊n/2⌋
            verifica(A[⌈⌊n/2⌋⌉, :])
        else
            verifica(A[:, ⌊n/2⌋])
```

(6) Algoritmos Aleatórios

(C) Armazenar Todo o Hi ocupa muito espaço.

(8) Linked Lists

a)



b) (Prev → Item) = 3

c)