Random: Time Complexity of Strassen's theorem

$a = 7$

$b = 2$

$c > 0$

$d = 1$

$\log_b a = \log_2 7 = [2,(n)] > c$ $\quad \Theta = n^{\log_2 7}$

· U C ·

UNIVERSIDADE DE COIMBRA
Faculty of Science and Technology
Department of Informatics Engineering

Laboratório de Programação Avançada
Written Test – June 19 2019

Name: Tiago Jorge Coimbra da Silva Student ID: 2022216219

10 grade points in total, 2h 30m, closed books.

1. Derive the computational time complexity of the following recursive algorithm to compute the value of a^n, $a \geq 0$, $n > 0$, and justify your answer with the Master Theorem. Assume that each arithmetic operation takes a constant amount of time. (1 g.p.)

```
Function exp(a,n)
  if n = 1 then
    return a
  else
    x = exp(a, ⌊n/2⌋)
    if n is even then
      return x·x
    else
      return x·x·a
```

Master Theorem (general version):
Let $a \geq 1$, $b > 1$, $d \geq 0$. → Fator de redução; → Cost of base case

Número de subproblemas criados (a)

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases} \Rightarrow$$

$$T(n) = \begin{cases} \Theta(n^c) & \text{if } \log_b a < c \\ \Theta(n^c \log n) & \text{if } \log_b a = c \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > c \end{cases}$$

$a = 1$

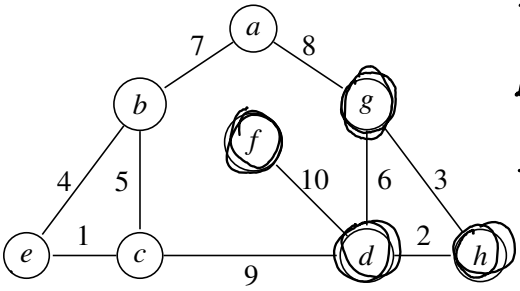
$b = 2$

$c = 0$

$\log_2 1 = 0 = c$

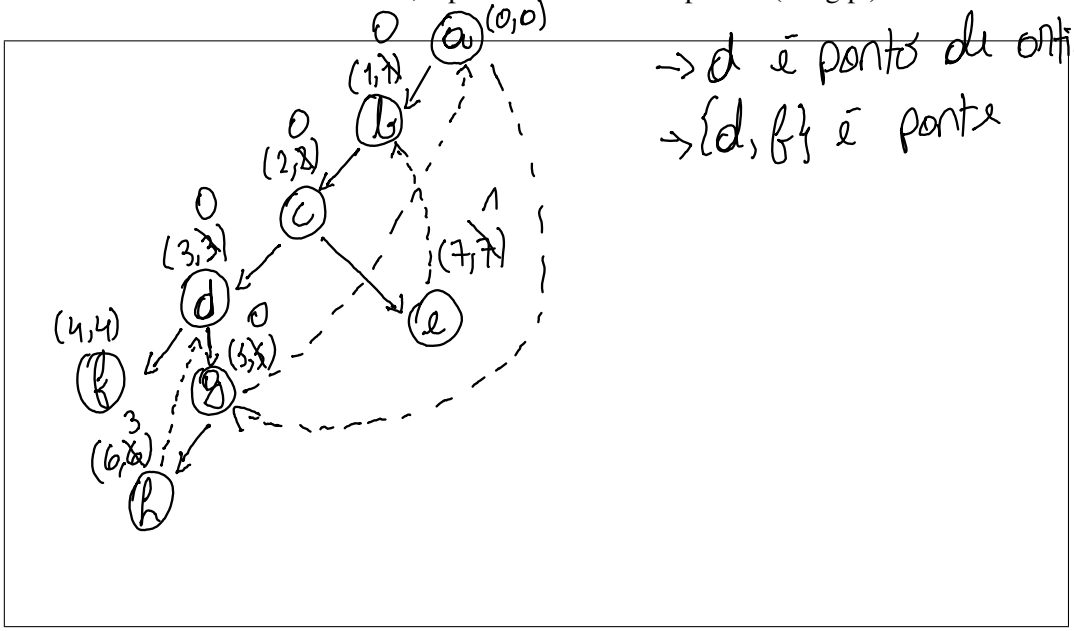
Assim o algoritmo tem complexidade $O(\log(n))$

2. Consider the following undirected graph.



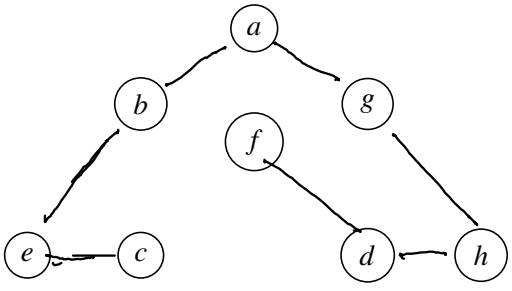
{e,c}✓ {b,a}✓
{d,h}✓ {a,g}✓
{g,h}✓ {c,d}✗
{b,e}✓ {b,d}
{b,c}✗
{g,d}✗

(a) Find the articulation points of the graph above (ignore the weights in the edges). Justify your answer by reporting the DFS tree starting from vertex *a*, choosing the vertices for traversal in alphabetic order of the labels, and by explicitly writing the final values for `dfs` and `low` at each vertex. In addition, report the articulation points. (1.5 g.p.)

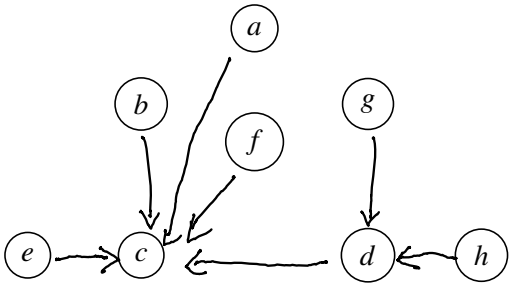


→ d é ponto de articulação
→ {d, f} é ponte

(b) Given the graph above, draw its minimum spanning tree (left) as well as the graph of the union-find data structure (right), without path compression, using Kruskal algorithm. Always connect the root of the tree with the smallest height to the root of the tree with largest height and, in case of a tie, choose as root the node with the smallest label. (1.5 g.p.)



(minimum spanning tree)

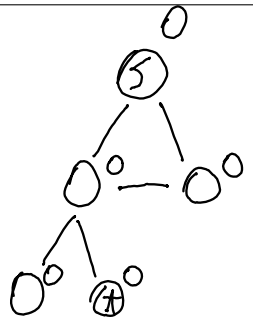


(union-find data structure)

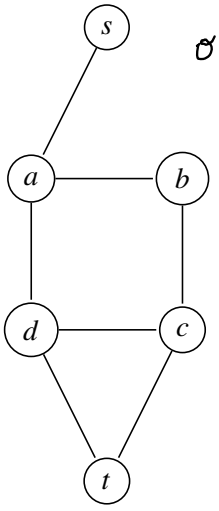
3. Consider the following problem: Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges in G, and two vertices s and t in V, find the longest simple path (with respect to the number of edges traversed by the path) from s to t. A simple path does not contain repeated vertices.

   (a) Write the pseudo-code of a recursive algorithm that computes the length (the number of edges) of the longest simple path between two given vertices of a graph. Assume that there is always a path between any two vertices of the graph. (1.5 g.p.)

```
Function DFS (v):
    visited [v] = 1
    if v = s then
        return ∞
    l = -∞
    for each nbr ∈ adj[v] do
        if !visited [nbr]
            l = max (l, DFS (nbr) + 1)
    return l
```



   (b) Show that optimal substructure *does not* apply to this problem if you consider the following definition of sub-problem: *find the longest simple path from s to v, for v in V*. Hint: Construct a counter-example using the graph below (1.5 g.p.).



A ideia da subestrutura ótima falha porque o caminho ótimo entre s e t não é composto por subcaminhos ótimos:

s → a → b → c → d → t porém o caminho mais longo entre s e b é:

s → a → d → t → c → b

4. Let p be a three-dimensional matrix of positive integers of size $n \times n \times n$. For a given n, we define $M(i,j)$, $1 \leq i \leq n$, $1 \leq j \leq n$, with the following recurrence relation:

$$M(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \leq k < j} \{M(i,k) + M(k+1,j) + p[i,k,j]\} & \text{if } i < j \end{cases}$$

(a) Give the pseudo-code of a top-down dynamic programming algorithm that explores the recurrence above to find the value for $M(1,n)$, for a given n. (1.5 g.p.)

```
Function M(i,j)
    if i=j then
    if return 0
    if memo then return
    elif i<j:
        dp[i][j] = ∞ , msf = ∞
        for k=i to j:
            msf = min(msf, M(i,k)+M(k+1,j)+p[i,k,j])
        dp[i][j] = msf
    return dp[i][j]
```

(b) Give the pseudo-code of a bottom-up dynamic programming algorithm that explores the recurrence above to find the value for $M(1,n)$, for a given n. (1.5 g.p.)

```
Function M(n):
    for i=0...n
        dp[i][i] = 0

    for i=1...n
        for j from 1 to n-i:
            dp[i][j] = ∞
            for k=i to j:
                dp[i][j] = min(dp[i][j] , ...)

    return dp[1,n]
```