$\langle 1, 6 \rangle$

$\langle 1, 3 \rangle \langle 3, 6 \rangle$

$\langle 1, 2 \rangle \langle 2, 3 \rangle$

· U C ·

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

Laboratório de Programação Avançada

Retake Exam – July 9 2019

Name: Tiago Jorge Coimbra da Silva Student ID: 2022216215

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

1. Derive the computational time complexity of the following recursive algorithm to compute the maximum and the minimum value of a sequence $S = (S[1], S[2], \ldots, S[n])$ with $n > 0$ values. Justify your answer with the Master Theorem. Assume that each arithmetic operation takes a constant amount of time. Note that the recursive function returns a pair of numbers and that the first recursive call is $mm(1, n, S)$. (1 g.p.)

Function $mm(i, j, S)$
  if $j - i \leq 1$ **then**
    $a = \max(S[i], S[j])$
    $b = \min(S[i], S[j])$
  else
    $(c, d) = mm(i, \lfloor (i+j)/2 \rfloor, S)$
    $(e, f) = mm(\lceil (i+j)/2 \rceil, j, S)$
    $a = max(c, e)$
    $b = min(d, f)$
  return (a, b)

n decreasing factor

Master Theorem (general version):
Let $a \geq 1$, $b > 1$, $d \geq 0$. → Base Case complexity

Num of subproblems created

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

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

Pelo Teorema de Mestre:

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

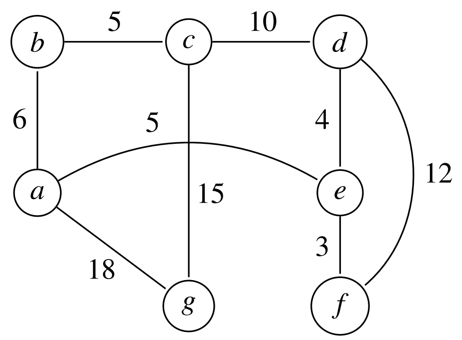
$a = 2$ $d = 1$
$b = 2$
$c = 0$

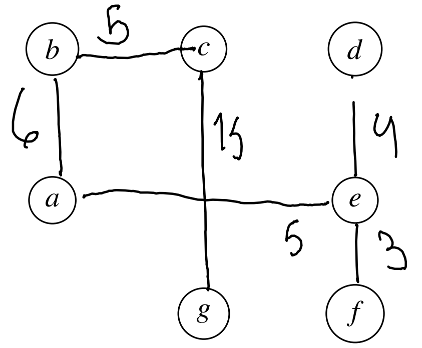
Pelo corolário, como $\log_b a = \log_2 2 = 1 > 0$ então

$T(n) = O(n)$

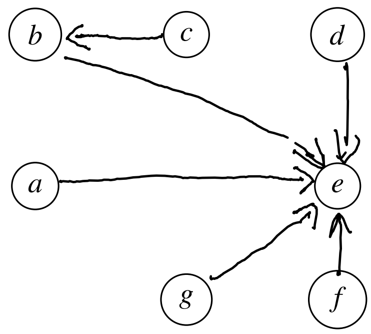
2. Given the graph below, draw its minimum spanning tree (left) as well as the graph of the union-find data structure (right), without path compression, using Kruskal algorithm. Always connect the root of the tree with the smallest height to the root of the tree with largest height and, in case of a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

3 {e,f} ✓ {d,f} 12 ✗
4 {e,d} ✓ {c,g} 15 ✓
5 {a,e} ✓ {a,g} 18 ✗
5 {b,c} ✓
6 {b,a} ✓
10 {c,d} ✗



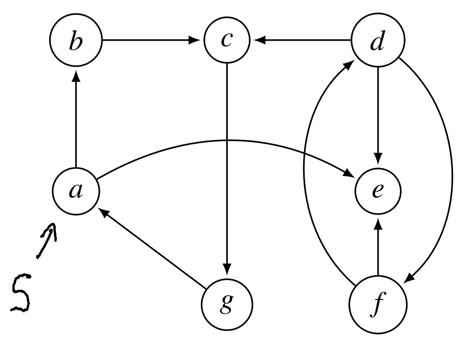


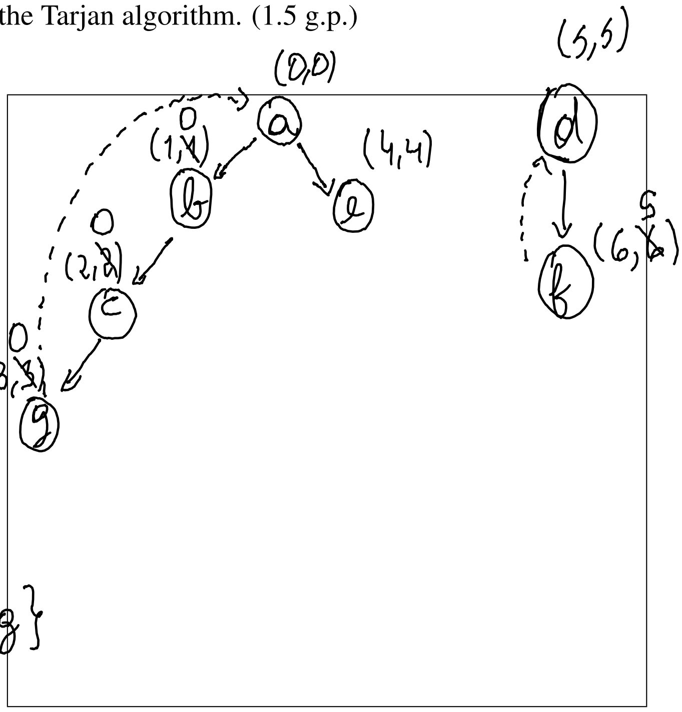
(minimum spanning tree)



(union-find data structure)

3. Find the strongly connected components of the following graph using Tarjan algorithm. Report the DFS tree(s) starting from vertex a and traversing the graph following the alphabetic order of the vertices labels. In addition, report the strongly connected components on the box below, ordered by the time their are found in the Tarjan algorithm. (1.5 g.p.)





SCC's: {e}, {d,f}, {a,b,c,g}

Assumption: S^* is optimal for P, $l \leq k$ objects with v^* value
Negation: S^* contains a suboptimal solution S' for subproblem P', $l \leq k-1$ objects, with optimal value $v^* - v$ for the set $S^* \setminus \{s\}$

Consequence: then it exists R for P that contains R' meaning its better then S^*

Contradiction: Therefore it must exist a set S^k with $s \in S^k$ wit value: $V' + V_s > V^*$ e $l \leq k$ which contradicts (1) so 2 isn't possible

4. Consider the following problem: Given a set $S = \{s_1, \ldots, s_n\}$ of $n > 0$ objects, each of which with a given value, find the subset of at most k objects that maximizes the total value. } P

(a) Let S^* be the optimal subset for the problem above using $\ell \leq k$ objects and let v^* be its optimal value. Prove that, if you remove an object s from S^*, with value v, the remaining subset of objects is an optimal solution for the same problem but considering set $S \setminus \{s\}$ of objects and a constraint of at most $k-1$ objects. (for simplification, consider that set $S^* \setminus \{s\} \neq \emptyset$.) (1.5 g.p.)

S^* → the optimal subset
l → number of objects used (At most k)
v^* → the optimal value

Prove that if we remove s with value v the subset will be optimal for $S \setminus \{s\}$ and k-1 objects

1. Assumption: S^* is optimal for P
2. Negation: S^* contains suboptimal solution S' for subproblem P'. Then it exists R' for P'
3. Consequence: Its possible to construct R to P that contains R' and its better than S^*
4. Contradiction: S cannot be optimal to P so S must contain R'.

(b) Write the pseudo-code of an algorithm that solves the problem. Identify the algorithm paradigm that you are considering and discuss its correctness and its space and time complexity. The grade to this answer depends on the efficiency (time complexity) of your approach. (1.5 g.p.)

Greedy

```
Function SS(n,k)
  sort(S), ans=0
  for i from 1 to k
    ans += S[i]
  Return ans
```

Dynamic Programming

dp[i][j] → max value for i objects
$0 \leq i \leq k \leq n$

```
Function SS(n,k,i)
  if i >= n then
    return 0
  if k=0 then return 0
  if dp[i][j] != -1 Return dp[i][i]
  dp[i][j] = max (SS(n,k,i+1), S[i]+SS(n,k-1,i+1))
  return dp[i][j]
```

5. Let D be a two-dimensional matrix of size $n \times n$. For a given n, we define $T(i,j)$, $1 \le i \le n$, $1 \le j \le n$, with the following recurrence relation:

$$T(i,j) = \begin{cases} +\infty & \text{if } i = 1 \\ \max\limits_{j \le \ell < n-i+2} \{\min\{D[j,\ell], T(i-1,\ell)\}\} & \text{if } i > 1 \end{cases}$$

(a) For a positive integer $k \le n$, give the pseudo-code of a top-down dynamic programming algorithm that explores the recurrence above to find the value for $\max\limits_{1 \le j \le n} T(k,j)$. Explicitly give the first call. (1.5 g.p.)

```
Function T(i,j)
  If i = 1
    Return ∞
  If (dp[i][j] != -1) Return dp[i][j]
  dp[i][j] = 0, ans = 0
  for l from j to n-i+2 do
      ans = min(D[j,l], T(i-1,l))
  return dp[i][j] = max(dp[i][j], ans)
```

First Call:

T(k,1)

(b) For a positive integer k, give the pseudo-code of a bottom-up dynamic programming algorithm that explores the recurrence above to find the value for $\max\limits_{1 \le j \le n} T(k,j)$. Discuss its time complexity. (1.5 g.p.)

```
Function T(k)
  for i from 1 to n do
       dp[1][i] = ∞
  for i from 2 to k
    for j from 1 to n
        dp[i][j] = 0, ans = 0
        for l from j to n-i+2
            ans = min(D[j,l], dp(i-1,l))
        dp[i][j] = max(dp[i][j], ans)
  return dp[k, n]
```

6