a -> número de subproblemas criados a cada passo

n -> tamanho do problema

n/b -> representa tamanho do subproblema

b -> indica por quanto o tamanho do problema é dividido

f(n) -> custo do passo recursivo

UNIVERSIDADE DE COIMBRA
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

Estratégias Algorítmicas
Exame de Recurso – 26 de junho de 2023

Nome: Tiago Jorge Coimbra da Silva   Nº de estudante: 2022216215

13 pontos no total, 2 horas, sem consulta.

1. Apresente a complexidade temporal do seguinte algoritmo recursivo relativamente ao número de elementos na lista A e justifique a sua resposta recorrendo ao Teorema Mestre. Assuma que A é uma lista de $n > 0$ inteiros, que o primeiro índice de A é 1 e que cada operação aritmética demora tempo constante. (2 pontos)

Function *product*(A, n)
  if $n = 1$ **then**
    return
  for $i = 1$ **to** $n/2$ **do**
    $A[i] = A[i] \times A[i + (n+1)/2]$
  product$(A, (n+1)/2)$

Teorema Mestre (versão geral):
Seja $a \geq 1$, $b > 1$, $d \geq 0$.

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

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

$T(n) = aT(n/b) + f(n)$ $f(n) = n$ $\Rightarrow T(n) = aT(n/b) + n^c,\ c = 1$

$a = 1$

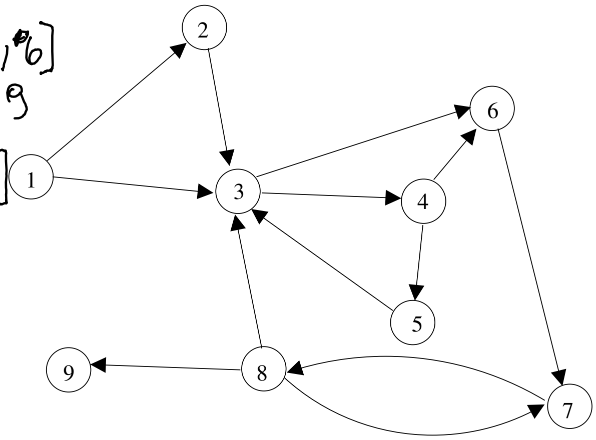
$b = 2$

$\log_2 1 < 1 \Rightarrow T(n) = O(n)$

2. Encontre as componentes fortemente conexas no seguinte grafo dirigido recorrendo ao algoritmo de Tarjan. Reporte a árvore de DFS deste algoritmo, começando no nó 3 e escolhendo os próximos nós por ordem crescente do valor nas etiquetas. Se necessitar de mais do que uma árvore de DFS, comece pelo nó que ainda não foi visitado que apresentar o menor valor de etiqueta. Indique explicitamente quais os nós que pertencem a cada componente fortemente conexa e quais os valores finais de `dfs` e `low` a cada nó. (2 pontos)

dfs: [7, 8, 0, 1, 2, 3, 4, 5, 6]
1 2 3 4 5 6 7 8 9

low: [7, 8, 0, ~~1~~ 0, ~~2~~ 0, ~~3~~ 0, ~~4~~ 0, ~~5~~ 0, 6]

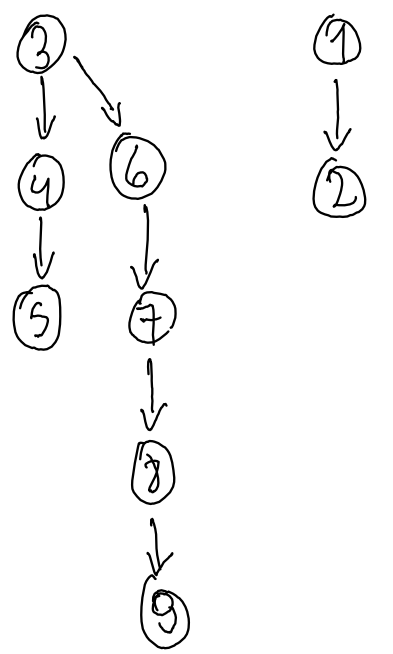


Componentes:

{3, 4, 5, 6, 7, 8}, {9}

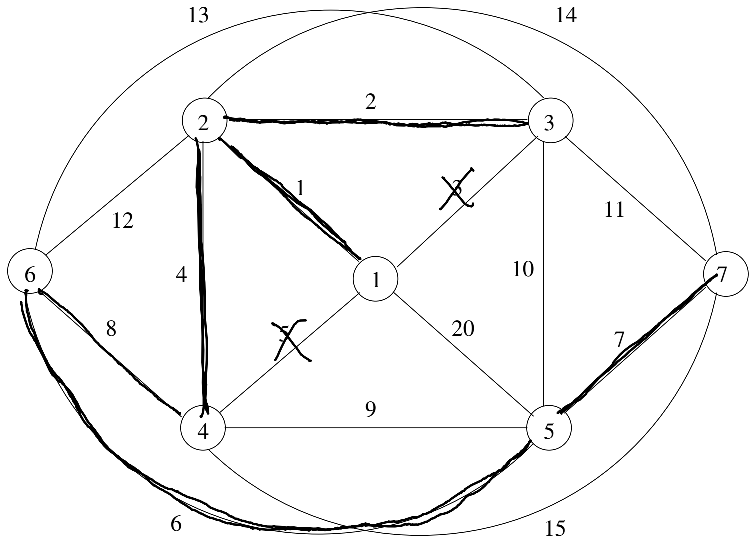
{2}, {1}

DFS Tree



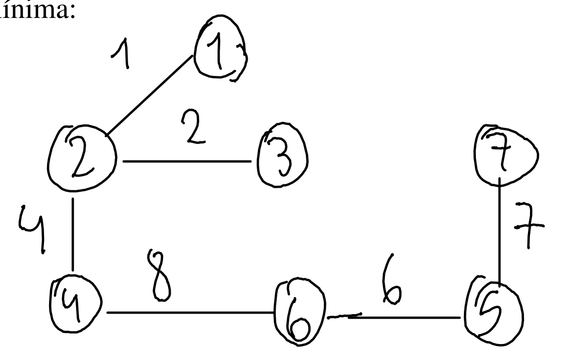
Árvore tem 6 arestas, |V|-1 vértices

3. Considere o seguinte grafo.



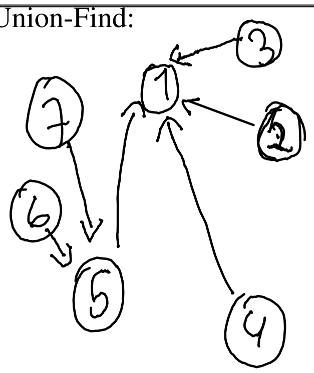
Desenhe a árvore geradora mínima e o grafo da estrutura de dados *union-find*, sem o passo de compressão de caminho, recorrendo ao algoritmo de Kruskal. Quando necessário, ligue a raiz da árvore com menor altura à raiz da árvore com maior altura e, em caso de empate, escolha, como raiz, o vértice que apresentar a etiqueta com o menor valor. (2 pontos)

Árvore Geradora Mínima:



Total Cost: 28

Estrutura de Dados Union-Find:



Rank(u[1]) > Rank(u[5])

4. Considere o seguinte algoritmo recursivo para calcular a média aritmética de $n > 0$ elementos contidos na lista L. Assuma que o primeiro índice da lista L é 1 e que os seus elementos são números reais não negativos.

Function $mean(L,n)$
  if $n = 1$ **then**
    return $L[n]$
  else
    return $L[n]/n + mean(L,n-1) \times (n-1)/n$

Mostre por indução que o algoritmo está correto, recorrendo à definição matemática da média aritmética. Apresente explicitamente o caso base, a hipótese de indução e o passo indutivo. (2 pontos)

Caso Base: para $n=1$ $mean(L,1) = L[1]$
Dado que a média aritmética é dada pela soma de $V[x]$ elementos dividido por x, $V[x]$ o valor do elemento x.

Hipótese Indutiva:

Assumir que para $k-1$ elementos:

$$Mean(L,k-1) = \frac{L[1] + L[2] + \dots + L[k-1]}{k-1}$$

Passo indutivo:

$$Mean(L,k) = \frac{L[k]}{k} + Mean(L,k-1) \times \frac{(k-1)}{k}$$

$$= \frac{L[k]}{k} + \frac{[L(1)+L(2)+\dots+L[k-1]]\cancel{(k-1)}}{k\cancel{(k-1)}}$$

$$= \frac{L(1)+L(2)+\dots+L[k-1]+L[k]}{k}$$

c.q.d

5. Considere o seguinte problema: Dada uma sequência de $n > 0$ inteiros, encontre uma subsequência contígua cuja a soma dos seus elementos é a maior. Por exemplo, para a sequência

$$(-2, 1, -3, 4, -1, 2, 1, -5, 4)$$

1 2 3 4 5 6 7 8 9

uma subsequência contígua com a maior soma é $(4, -1, 2, 1)$ com o valor 6. O seguinte algoritmo de programação dinâmica resolve o problema para uma sequência A de n elementos reportando unicamente a maior soma.

dp: [-2, 1, -2, 4, 3, 5, 6, 1, 5]

1 2 3 4 5 6 7 8 9

Function *msum*(A)
  $DP[1] = A[1]$
  for $i = 2$ **to** n **do**
    $DP[i] = \max(A[i], DP[i-1] + A[i])$
  return $\max(DP[1], \ldots, DP[n])$

Apresente o pseudo-código de um algoritmo que reconstrua a subsequência contígua com a maior soma a partir do vetor DP retornado pelo algoritmo acima. (2 pontos)

```
auto it = max_element(all(A))
int idx = it - A.begin()

Function find_subset(DP)
  sol = []
  auto it = max_element(All(DP))
  int idx = it - DP.begin()
  while DP[idx] != A[idx]
      sol.push_back(A[idx])
      idx--
  Return sol
```

6. Considere o seguinte problema: Dado um conjunto N de n inteiros positivos cuja a soma é S, determine se é possível encontrar uma partição de N em três subconjuntos disjuntos tal que a soma de cada subconjunto seja igual. Escreva o pseudo-código de uma abordagem de programação dinâmica que resolva o problema (deve retornar *True* se existe tal partição, ou *False* se não existe). Explique porque razão a sua abordagem está correta e determine a sua complexidade computacional (tempo e memória) relativamente aos parâmetros n e S do problema. Assuma que S é sempre divisível por três. (3 pontos)

$S/3 \rightarrow$ objetivo $\rightarrow T$

[4, 5, 7, 2] $\rightarrow T = 6$ False

[(4, 2), (5, 1), (3, 1, 2)]

Nota: a, b, c são arrays de inteiros ("buckets")

```
Function possible(N, a, b, c, idx)
    If idx == n:
        return a == b == c == S/3
    num = N[idx]
    if sum(a)+num <= target and possible(N, a+num, b, c, idx+1)
        return true;
            (:) same for b, c
    return False
```

Time Complexity: 3^n

Space Complexity: $n \times \left(\frac{S}{3}\right)^2$