Instituto Superior Técnico

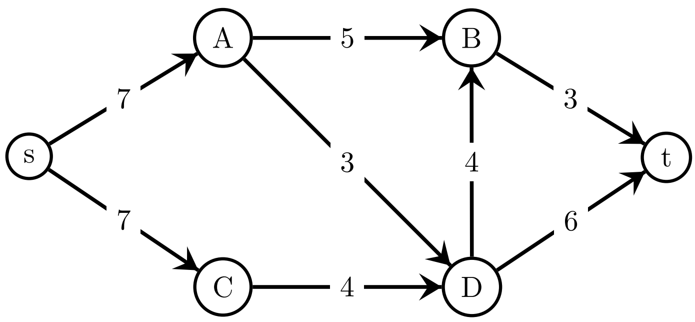
Análise e Síntese de Algoritmos

Ano Lectivo 2020/2021 — 1º Teste

RESOLUÇÃO

I. (2,5 + 2,5 + 2,5 + 2,5 = 10,0 val.)

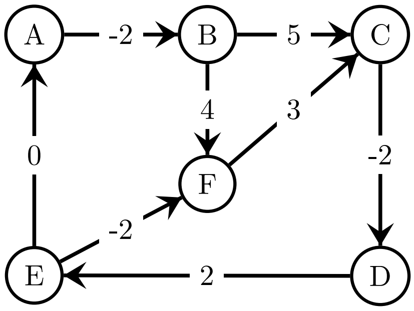
I.a) Considere a rede de fluxo da figura onde s e t são respectivamente os vértices fonte e destino na rede. Aplique o algoritmo Relabel-To-Front na rede de fluxo. Considere que as listas de vizinhos dos vértices intermédios são as seguintes:
$N[A] = \langle B, D, s\rangle$
$N[B] = \langle t, A, D\rangle$
$N[C] = \langle D, s\rangle$
$N[D] = \langle t, B, A, C\rangle$
e que a lista de vértices inicial é $L = \langle C, A, B, D\rangle$.



Indique o valor do fluxo a chegar a t e a altura de cada vértice após a 4ª reconfiguração de L antes de executar qualquer operação de `discharge()`. Se o algoritmo terminar antes, considere os valores finais das alturas e fluxo. Indique também a sequência de <u>diferentes</u> configurações de L, sem incluir a 1ª.

	s	A	B	C	D	t
$h()$						
$L : \langle C, A, B, D\rangle$					$f(V,t) =$	

I.b) Considere a aplicação do algoritmo de Johnson ao grafo dirigido e pesado da figura.

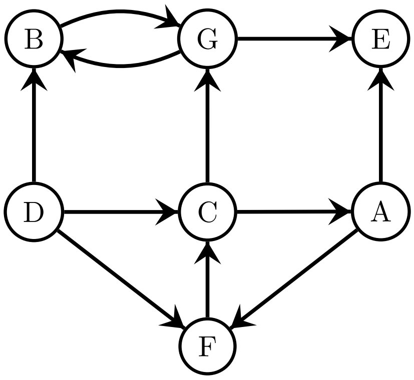


Calcule os valores de $h(u)$ para todos os vértices $u \in V$ do grafo. Calcule também os pesos de todos os arcos após a repesagem.

	A	B	C	D	E	F
$h()$						

$\widehat{w}(A,B)$	$\widehat{w}(B,C)$	$\widehat{w}(B,F)$	$\widehat{w}(C,D)$	$\widehat{w}(D,E)$	$\widehat{w}(E,A)$	$\widehat{w}(E,F)$	$\widehat{w}(F,C)$

I.c) Considere o grafo dirigido da figura.



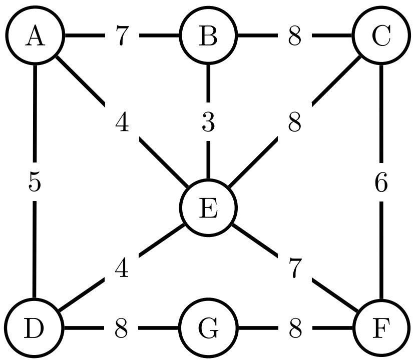
Aplique o algoritmo de Tarjan para identificar os componentes fortemente ligados, considerando o vértice G como inicial. Durante a aplicação do algoritmo considere que tanto a escolha dos vértices a visitar, como a pesquisa dos vértices adjacentes são feitas por ordem lexicográfica (ou seja, A, B, C, ...).

Indique os componentes fortemente ligados do grafo pela ordem segundo a qual são identificados/fechados pelo algoritmo e o valor *low* calculado para cada vértice.

Considere que o tempo de descoberta d começa em 1.

	A	B	C	D	E	F	G
low()							
SCCs por ordem							

I.d) Considere o grafo não dirigido e pesado da figura.



Considere a execução do algoritmo de Kruskal para determinar árvores abrangentes de menor custo, até processar arcos de peso 6, inclusivé.
Indique a soma do peso dos arcos seguros seleccionados, bem como quais os conjuntos disjuntos existentes nessa fase do algoritmo.

Soma pesos:	
Conjuntos disjuntos:	

Indique ainda o custo da MST calculada.

Custo MST:	

II. (2,5 + 2,5 + 2,5 + 2,5 = 10 val.)

II.a) Considere a função recursiva:

```
int f(int n) {
  int sum = 0;

  for (int j = n; j>0; j/=2) {
    for (int k=0; k<j; k+=1) { // Loop 1
      sum += 1;
    }
  }

  for (int i=1; i<n; i*=2) {
    for (int k=n; k>0; k/=2) { // Loop 2
      sum += 1;
    }
  }

  return sum+4*f(n/2);
}
```

1. Determine o menor majorante assimptótico medido em função do parâmetro n para o número total de iterações dos loops **1** e **2** por cada chamada à função f.
2. Determine o menor majorante assimptótico da função f, em função do parâmetro n, utilizando os métodos que conhece.

Solução:

1. Analisamos cada loop separadamente:

   - *Loop 1:* Contamos o número de iterações do *loop 1* por cada iteração do loop exterior que o contém. Para tal, precisamos de determinar o valor da variável j em função da iteração do loop exterior.
     - Iteração 0: $j = n = n/2^0$. *Loop 1:* $n/2^0$.
     - Iteração 1: $j = n/2 = n/2^1$. *Loop 1:* $n/2^1$.
     - Iteração 2: $j = n/4 = n/2^2$. *Loop 1:* $n/2^2$.
     - ...
     - Iteração k: $j = n/2^k$. *Loop 1:* $n/2^k$.

     Dado que o loop exterior é executado $log\ n$ vezes, concluímos que o número total de iterações é do loop 1 é dado por:

     $$\sum_{k=0}^{log\ n} \frac{n}{2^k} = n. \sum_{k=0}^{log\ n} \frac{1}{2^k} \leq 2.n.log\ n = O(n.log\ n)$$

   - *Loop 2:* O *loop 2* é executado $log\ n$ vezes por cada iteração do loop exterior que o contém. Por sua vez, o loop exterior é também executado $log\ n$ vezes. Pelo que concluímos que o *loop 2* é executado $O((log\ n)^2)$ vezes.

2. Observamos que:

   $$T(n) = T(n/2) + O(n.log\ n)$$

   Podemos aplicar o Teorema Mestre na forma geral notando que $a = 1$, $b = 2$ e $log_b\ a = 0$. Notamos ainda que $n.log\ n = \Omega(n^{(0+\epsilon)}$, de onde concluímos que $T(n) = O(n.log\ n)$.

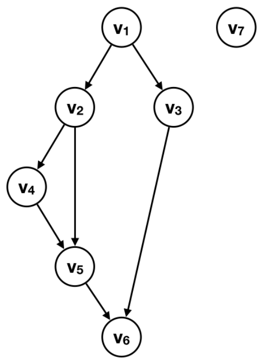
II.b) Considere a seguinte implementação iterativa do algoritmo DFS.

```
function DFS(G)
st ← MkStack()
for each vertex v ∈ sources(G) do
   st.push(v)
end for
for each vertex v ∈ G.V do
   v.color ← white
end for
while !(st.isEmpty()) do
   u ← st.head()
   if (u.color == white) then
      for each vertex w ∈ G.Adj[u] do
         if (w.color == white) then
            st.push(w)
         end if
      end for
      u.color ← gray
   else if (u.color == gray) then
      u.color ← black
      st.pop()
   else
      st.pop()
```

```
    end if
end while
```

Onde a função $mkStack()$ retorna uma nova pilha e a função $sources(G)$ retorna um vector com os vértices de G que não possuem arcos incidentes por ordem crescente de índice. As funções $isEmpty()$, $push()$ e $pop()$ são usadas, respectivamente, para: testar se a pilha está ou não vazia, adicionar um novo elemento ao topo da pilha e remover o elemento que se encontra no topo da pilha.

1. Admitindo que as listas de vizinhos de cada nós estão organizadas por ordem crescente de índice, indique a sequência de todas as pilhas produzidas aquando da aplicação do algoritmo ao grafo que se ilustra em baixo. Deve indicar **apenas** o estado da pilha no início de cada iteração do ciclo *while*.



2. Qual a complexidade do algoritmo sugerido. Justifique a resposta apresentando um limite superior para o número iterações do ciclo *while* e do seu ciclo *for* interior.

3. Explique como poderia modificar a implementação fornecida para adicionalmente calcular os tempos de início e de fim de cada nó.

Solução:

1. As cinco primeiras stacks são as seguintes:

   - $\langle 1, 7 \rangle$
   - $\langle 1, 7 \rangle$
   - $\langle 1 \rangle$
   - $\langle 1, 2, 3 \rangle$
   - $\langle 1, 2, 3, 6 \rangle$
   - $\langle 1, 2, 3, 6 \rangle$
   - $\langle 1, 2, 3 \rangle$
   - $\langle 1, 2 \rangle$
   - $\langle 1, 2, 4, 5 \rangle$
   - $\langle 1, 2, 4, 5 \rangle$
   - $\langle 1, 2, 4 \rangle$
   - $\langle 1, 2 \rangle$
   - $\langle 1 \rangle$
   - $\langle \rangle$

2. Complexidade: $O(E+V)$.
   - Ciclo *while:* $O(E)$.
   - Ciclo *for:* $O(E)$.
   - Inicialicação: $O(V)$.
3. O tempo de início é registado quando o nó é pintado de cinzento. O tempo de fim é registado quando o nó é pintado de preto.

II.c) O Eng. Caracol quer deslocar-se entre as cidades A e B em Caracolândia. Para tal, dispõe de um mapa de Caracolândia que modelou como um grafo dirigido $G = (V, E)$, cujos vértices correspondem às cidades de Caracolândia e os arcos às estradas que as unem. O Eng. Caracol anotou cada arco $(u, v) \in E$ com a quantidade de combustível $w(u, v)$ que o seu automável necessitaria para percorrer a respectiva estrada, arco (u,v), sem abastecer. O Eng. Carocol depara-se no entanto com um problema: em Caracolândia todos os postos de abastecimento se encontram em cidades, não havendo nenhum entre cidades. Além disso, o automóvel do Eng. Caracol tem um depósito de combustível com capacidade W.

1. Proponha um algoritmo linear para determinar se o Eng. Caracol consegue deslocar-se no seu automóvel da cidade A até à cidade B.
2. O Eng. Caracol vai comprar um novo automóvel. Proponha um algoritmo para determinar a capacidade mínima do depósito W' do novo automóvel, de forma a garantir que o Eng. Caracol se consegue deslocar da cidade A até à cidade B, gastando o mínimo combustível possível. Indique a complexidade do algoritmo proposto. Nota: assuma que todos os caminhos têm pesos diferentes.

Solução:

1. Algoritmo para decidir se o Eng. Carocal se consegue deslocar entre as cidades A e B:
   - Construir um novo grafo $G' = (V, E')$ onde:

     $$E' = \{(u, v) \mid (u, v) \in E \ \wedge \ w(u, v) \leq W\}$$

     Complexidade: $O(V + E)$.
   - Determinar se B é atingível a partir de A no grafo G' usando uma DFS. Complexidade: $O(V + E)$.
   - Complexidade total: $O(V + E)$.
2. Algoritmo para determinar a capacidade mínima do novo depósito, W':
   - Determinar o caminho de peso mínimo que liga a cidade A à cidade B no grafo G, p, usando o algoritmo de Dijkstra. Complexidade: $O(E.log\ V)$.
   - W' corresponde ao peso do arco mais pesado de p:

     $$W' = max\{w(u, v) \mid (u, v) \in p\}$$

     Complexidade: $O(V)$.
   - Complexidade total: $O(E.log\ V)$.

II.d) O governo de Caracolândia tem n residentes $\{R_1, ..., R_n\}$; m clubes $\{C_1, ..., C_m\}$; e k partidos políticos $\{P_1, ..., P_k\}$, e decidiu estabelecer uma comissão para financiamento de eventos lúdicos. A comissão deve ser constituída por um residente indicado por cada clube. Contudo, de modo a garantir o peso relativo dos vários partidos políticos, exige-se ainda que a comissão seja integrada por exactamente c_i membros de cada partido P_i. Cada residente pode pertencer a vários clubes mas apenas a um único partido político.

O governo de Caracolândia pretende agora determinar se é possível constituir uma comissão que satisfaça as restrições estabelecidas.

1. Modele o problema da constituição da comissão como um problema de fluxo máximo.
2. Admitindo que o número total de residentes n é muito superior ao número de partidos, número de clubes, e número total de elementos da comissão, indique a complexidade assimptótica, medida em função dos parâmetros do problema, do algoritmo Relabel-To-Front e do algoritmo do Edmonds-Karp. Indique que algoritmo utilizaria para calcular o fluxo máximo.

1. *Construção da rede de fluxo:* $G = (V, E, c, s, t)$. Na construção da rede de fluxo consideramos um vértice por residente, um vértice por partido político, e um vértice por comissão, e dois vértices adicionais s e t, respectivamente a fonte e o sumidouro. Formalmente:
   - $V = \{s, t\} \cup \{R_i \mid 1 \leq i \leq n\} \cup \{C_i \mid 1 \leq i \leq m\} \cup \{P_i \mid 1 \leq i \leq n\}$
   - $$\begin{array}{rll} E = & \{(s, C_i, 1) \mid 1 \leq i \leq n\} & C_i \text{ pode nomear } \underline{\text{um}} \text{ representante} \\ & \cup\, \{(C_i, R_j, 1) \mid R_j \text{ é membro do clube } C_i\} & \\ & \cup\, \{(R_i, P_j, 1) \mid R_i \text{ é membro do partido } P_j\} & \\ & \cup\, \{(P_i, t, c_i) \mid 1 \leq i \leq k\} & P_i \text{ pode nomear } c_i \text{ representantes} \end{array}$$
2. *Complexidade:*
   - $|V| = n + m + k + 2 = O(n)$
   - $|E| \leq n + n.m + 2k = O(n.m)$
   - $|f^*| \leq \sum_{i=1}^{k} c_k \leq n = O(n)$
   - Edmonds Karp (upper bound de FF): $O(|f^*|.E) = O(n.n.m) = O(n^2.m)$
   - Edmonds Karp (upper bound EK): $O(E^2.V) = O(n^2.m^2.n) = O(n^3.m^2)$
   - Relabel-To-Front: $O(n^3)$

   Dado que $n > m$, o algoritmo a utilizar é o algoritmo de Edmonds-Karp.