INSTITUTO SUPERIOR TÉCNICO

Análise e Síntese de Algoritmos

Ano Lectivo 2018/2019 1º Teste - versão A

RESOLUÇÃO

I. (2,5 + 2,5 + 2,5 + 2,5 + 2,5 + 2,5 = 15,0 val.)

I.a)

	x_4	x_8	x_{10}	x_{11}	x_{12}
$rank[x_i]$	1	1	3	0	2
$p[x_i]$	x_{10}	x_8	x_{10}	x_{12}	x_{10}

I.b)

Ordenação topológica:	A, C, F, B, D, G, E, H
Nº Ordenações topológicas:	6

I.c)

Ordem vértices	1	2	3	4	5	6	7	8	9
v	A	B	D	E	H	I	F	G	C
$key[v]$	0	1	3	2	1	1	2	3	4

I.d)

	A	B	C	D	E	F	G	H
$h()$	-2X	-X	0	-2X-Y	-3X-Y	-X	0	-4X-Y

$\widehat{w}(A,B)$	$\widehat{w}(B,E)$	$\widehat{w}(C,E)$	$\widehat{w}(E,G)$	$\widehat{w}(G,H)$
0	X+Y	4X+Y	3Y-3X	5X+Y

I.e)

Expressão	$T(n) = 2 * T(n/2) + O(n)$
Majorante	$O(n\ lg\ n)$

I.f)

	A	B	C	D	E
$h()$	1	1	8	1	9

Corte :	$\{s,C,E\}/\{A,B,D,t\}$	$f(S,T) =$	15

II. (2,0 + 3,0 = 5,0 val.)

II.a) $\langle$ XXX $\rangle$ **II.b)** $\langle$ XXX $\rangle$

I. (2,5 + 2,5 + 2,5 + 2,5 + 2,5 + 2,5 = 15,0 val.)

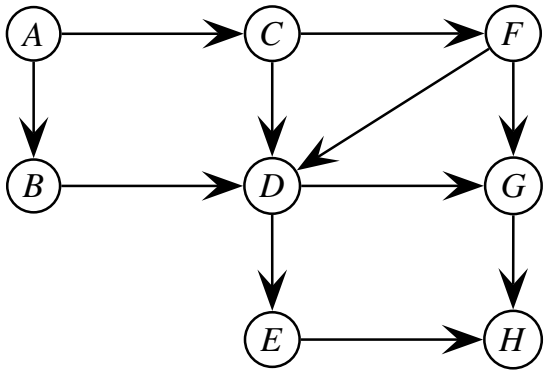
I.a) Considere o seguinte conjunto de operações sobre conjuntos disjuntos:

```
1 for i = 1 to 12 do
2     Make-Set (x_i)
3 for i = 1 to 6 do
4     Union (x_{2i-1}, x_{2i})
5 Union (x_4, x_11)
6 Union (x_2, x_9)
7 Union (x_1, x_6)
8 Union (x_11, x_5)
9 return Find-Set(x_3)
```

Use a estrutura em árvore para representação de conjuntos disjuntos com a aplicação das heuristicas de união por categoria e compressão de caminhos. Para os elementos $x_4, x_8, x_{10}, x_{11}, x_{12}$, indique os valores de categoria ($rank[x_i]$) e o valor do seu pai na árvore ($p[x_i]$).

Nota: Na operação $Make-Set(x)$, o valor da categoria de x é inicializado a 0. Na operação de $Union(x, y)$, em caso de empate, considere que o representante de y é que fica na raíz.

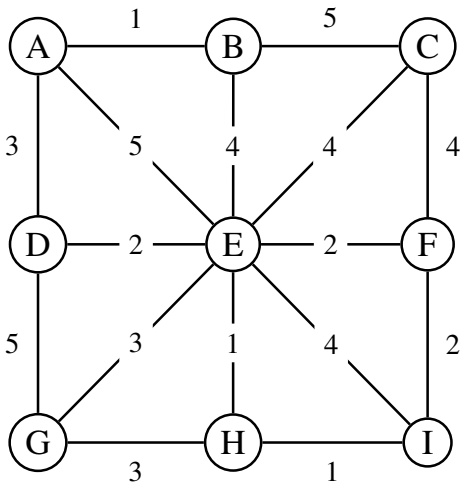
I.b) Considere o grafo dirigido:



Considere o grafo dirigido e acíclico da figura. Aplique o algoritmo de travessia em profundidade primeiro (DFS) para encontrar uma ordenação topológica no grafo. Inicie a travessia no vértice A. Durante a aplicação do algoritmo considere que os vértices adjacentes são analisados por ordem lexicográfica (ou seja, A, B, C...).

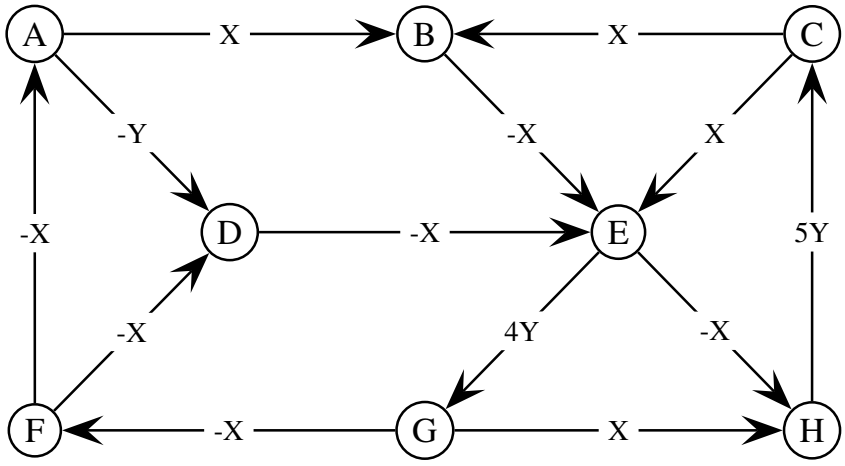
Indique a ordenação topológica que obtém aplicando este algoritmo. Indique o número de ordenações topológicas diferentes que se consegue definir neste grafo.

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



Aplique o algoritmo de Prim ao grafo, considerando o vértice A como origem. Indique a ordem pela qual os vértices são removidos da fila de prioridade durante a execução do algoritmo. Em caso de empate, considere os vértices por ordem lexicográfica.
Para cada vértice, indique também qual o valor da sua chave quando são removidos da fila de prioridade.

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



Sabendo que $Y > X > 0$, calcule os valores de $h(u)$ para todos os vértices $u \in V$ do grafo em função de X e Y. Calcule também os pesos dos seguintes arcos após a repesagem: (A,B), (B,E), (C,E), (E,G), (G,H).

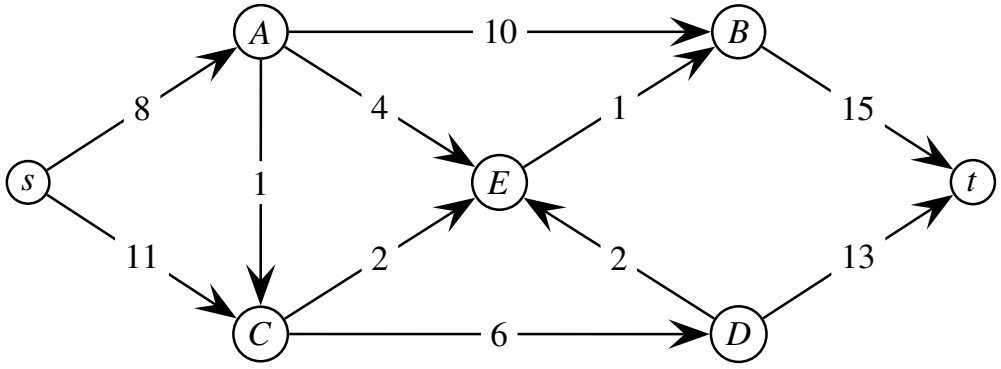
I.e) Considere a função recursiva:

```
int f(int n)
{
  int i = 0, j = n*n;
  while(j > 0) {
    i++;
    j = j / 2;
  }

  if(n > 1)
    i = i * f(n/2) + n * f(n/2);

  while (j < n) {
    i = i + 2;
    j++;
  }
  return i;
}
```

Indique a expressão (recursiva) que descreve o tempo de execução da função em termos do número n, e de seguida, utilizando os métodos que conhece, determine o menor majorante assimptótico.

I.f) 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 a lista de vértices é inicializada por ordem alfabética e que os vizinhos de cada vértice também estão ordenados alfabeticamente. Assim, as listas de vizinhos dos vértices intermédios são as seguintes:
$N[A] = < B,C,E,s >$ $N[B] = < A,E,t >$ $N[C] = < A,D,E,s >$ $N[D] = < C,E,t >$ $N[E] = < A,B,C,D >$



Indique a altura final de cada vértice. Indique ainda o corte mínimo da rede e o valor do fluxo máximo.

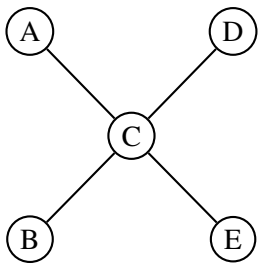
II. (2,0 + 3,0 = 5,0 val.)

II.a) Considere um grafo $G = (V,E)$ não dirigido e ligado. Suponha que existe um vértice $u \in V$ que é ponto de articulação no grafo. Recorde que um vértice é um ponto de articulação se removido faz com que vértices anteriormente ligados por um caminho deixam de estar ligados.

Considere uma lista L com n novos arcos a adicionar ao grafo. Supondo que os arcos de L são adicionados ao grafo de forma sequencial pela ordem que ocorrem na lista, proponha um algoritmo que recebe o grafo G, o ponto de articulação u e uma lista de arcos L e determina o número de arcos da lista L a partir do qual o vértice u deixa de ser um ponto de articulação.

Apenas as soluções mais eficientes serão consideradas. Tem que indicar a complexidade do algoritmo proposto.

Por exemplo, no grafo da figura, o vértice C é ponto de articulação do grafo. Se a lista L fosse $< (A,B),(A,D),(B,E),(B,D) >$, então o seu algoritmo deve retornar 3 porque a partir do momento que os três primeiros arcos são adicionados ao grafo, então o vértice C deixa de ser ponto de articulação. O seu algoritmo deve retornar -1 caso após todos os vértices de L sejam adicionados ao grafo, o vértice ainda permanece como ponto de articulação.



Solução:

Considerando que o grafo é não dirigido e ligado, então um vértice u será ponto de articulação se ao remover o vértice u (e os respectivos arcos) o grafo não ficar ligado.

Comecemos por identificar os componentes do grafo se o vértice u fosse removido. Para isso, podemos usar estruturas de dados para conjuntos disjuntos. Inicialmente, temos $|V|-1$ conjuntos onde cada vértice $v \in V \setminus \{u\}$ define um conjunto. De seguida, para cada arco $(x,y) \in E$ não incidente em u (ou seja, $x \neq u$ e $y \neq u$), se x e y estiverem em conjuntos diferentes fazemos a união dos conjuntos a que pertencem os vértices.

Finalmente, percorremos a lista L de forma sequencial e aplicamos a mesmo procedimento que aos restantes arcos. Se após análise do arco k da sequência ficamos com apenas um conjunto de vértices, então o algoritmo retorna k. Caso contrário, continua para o arco seguinte na lista L. Se não houver mais arcos na lista, o algoritmo termina devolvendo -1.

A complexidade do algoritmo proposto é $O((E+L)\alpha(V))$.

II.b) Considere um grafo $G = (V, E)$ dirigido e pesado. Suponha que a função de pesos $w : E \to Z$ admite que alguns arcos do grafo $(u, v) \in E$ tenham pesos negativos (i.e. $w(u, v) < 0$) tal que seja possível formar pelo menos um ciclo de peso negativo.

Recorde que um ciclo de peso negativo é um caminho $p =< v_0, v_1 \ldots v_k >$ tal que $v_0 = v_k$ e o peso total do caminho seja negativo (i.e. $w(p) < 0$).

Proponha um algoritmo que permite identificar o **menor ciclo negativo no grafo em termos do número de arcos**, ou seja, devolve o menor número de arcos que compõem um ciclo negativo no grafo. Indique a complexidade do algoritmo proposto.

Sugestão: Pode supor que o grafo é representado através de uma matriz de adjacência.

Solução:

Seja W a matriz de adjacência do grafo G. Esta matriz representa todos os caminhos mais curtos entre todos os pares de vértices, se consideramos apenas caminhos com 1 arco. Seja $D^{(k)}$ a matriz que representa todos os caminhos mais curtos considerando caminhos com um máximo de k arcos.

O problema proposto pode ser resolvido calculando as sucessivas matrizes $D^{(k)}$ até que haja alguma entrada $d_{ii}^{(k)} < 0$. Sendo $D^{(1)} = W$, e sabendo que $D^{(k)} = \text{Extend-Shortest-Paths}(D^{(k-1)}, W)$, então conseguimos resolver o problema aplicando sucessivamente esta função até obtermos a condição de paragem.

No pior caso, este algoritmo é $O(V^4)$. No máximo temos $O(V)$ chamadas à função Extend-Shortest-Paths, sendo que cada chamada é $O(V^3)$.

Há algoritmos mais eficientes para resolver o problema, mas esta solução seria suficiente para obtenção da cotação total.