Instituto Superior Técnico

Análise e Síntese de Algoritmos

Ano Lectivo 2020/2021 Repescagem do 1º Teste - versão A

RESOLUÇÃO

I. (2.5 + 2.5 + 2.5 + 2.5 = 10 val.)

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

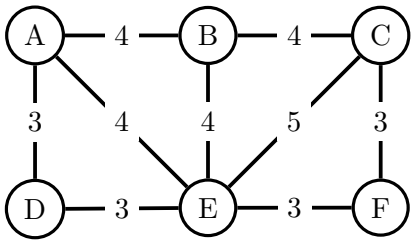
```
1 for i = 0 to 9 do
2     Make-Set (x_i)
3 for i = 0 to 4 do
4     Union (x_{2*i}, x_{2*i+1})
5 for i = 0 to 1 do
6     Union (x_{2*i}, x_{2*i+4})
7 Union (x_2, x_8)
8 Union (x_8, x_4)
9 Find-Set(x_0)
```

Use a estrutura em árvore para representação de conjuntos disjuntos com a aplicação das heuristicas de união por categoria e compressão de caminhos. Para cada elemento x_i ($0 \leq i \leq 9$) indique os valores de categoria ($rank[x_i]$) e o valor do seu pai na árvore que representa os conjuntos ($p[x_i]$).

Nota: Na operação $Make\text{-}Set(x)$, 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.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$rank[x_i]$	0	1	0	1	0	3	0	2	0	1
$p[x_i]$	5	5	7	7	5	5	7	5	7	7

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

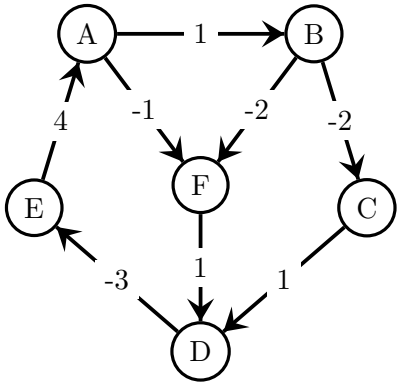


Aplique o algoritmo de Prim ao grafo, considerando o vértice **D** como origem.
Para cada vértice, indique qual o valor da sua chave (k) e o de seu antecessor (π), quando o vértice é removido da fila de prioridade. Em caso de empate, considere os vértices por ordem lexicográfica.
Indique ainda o peso da árvore abrangente de menor custo encontrada, e o número árvores distintas com esse mesmo custo.

Ordem vértices	1	2	3	4	5	6
v	**D**	**A**	**E**	**F**	**C**	**B**
$key[v]$	**0**	**3**	**3**	**3**	**3**	**4**
$\pi[v]$	**NIL**	**D**	**D**	**E**	**F**	**A**

Peso MST:	**16**
Nº de MSTs:	**3**

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

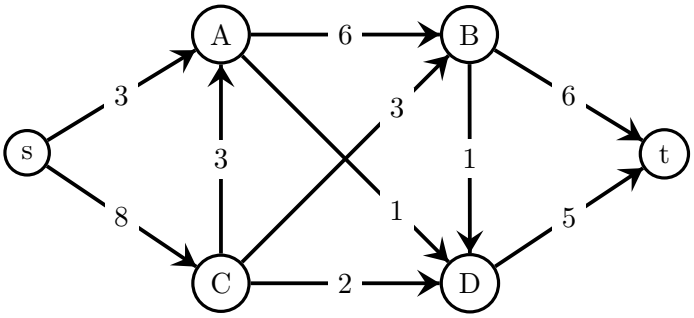


Calcule os valores de $h(u)$ para todos os vértices $u \in V$ do grafo. Calcule também os pesos de todos os arcos após a repesagem.

	A	B	C	D	E	F
$h()$	**0**	**0**	**-2**	**-1**	**-4**	**-2**

$\widehat{w}(A,B)$	$\widehat{w}(A,F)$	$\widehat{w}(B,C)$	$\widehat{w}(B,F)$	$\widehat{w}(C,D)$	$\widehat{w}(D,E)$	$\widehat{w}(E,A)$	$\widehat{w}(F,D)$
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**

I.d) Aplique o algoritmo de Edmonds-Karp na seguinte rede de fluxo, onde s e t são respectivamente os vértices fonte e destino na rede.



Indique um corte mínimo da rede, o valor do fluxo máximo, o número de caminhos de aumento, e o fluxo de cada arco após a aplicação do algoritmo. Nota: Na selecção do caminho de aumento, em caso de empate (caminhos de aumento com o mesmo comprimento), escolha o menor caminho de aumento por ordem lexicográfica.

f(s,A)	f(s,C)	f(A,B)	f(A,D)	f(B,D)	f(B,t)	f(C,A)	f(C,B)	f(C,D)	f(D,t)
3	**7**	**4**	**1**	**1**	**6**	**2**	**3**	**2**	**4**

Corte:	$\{s, A, B, C\}/\{D, t\}$	f(S,T) =	**10**
Numero de caminhos de aumento :		**5**	

II. (2.5 + 2.5 + 2.5 + 2.5 = 10 val.)

II.a) Considere a função recursiva:

```
int f(int n) {

  int i = 0, j=0, z=0;
  while (j + z < n) {  // Loop 1
    z += 1;
    j += i;
    i += 2;
  }

  int r = 0;
  if (n > 0) r = 3*f(n/2)

  j = 1; z = 0;
  while (j<n) {  // Loop 2
    j *= 2;
    z += 1;
  }

  return r+i+z;
}
```

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. Consideramos os dois loops separadamente.
   - *Loop 1*: Definimos $z(k)$, $i(k)$ e $j(k)$ como sendo os valores das variáveis z, i e j no final da k-ésima iteração do loop 1:
     - $z(k) = k$
     - $i(k) = 2 * k$
     - $j(k) = j(k-1) + i(k-1) = j(k-1) + 2 * (k-1)$
       $j(k) = 2 * ((k-1) + (k-2) + ... + 1) = 2 * \frac{k*(k-1)}{2} = k * (k-1)$

     Resolvemos a condição de paragem em função de k:

     $$j(k) + z(k) = n \Leftrightarrow k * (k-1) + k = n \Leftrightarrow k = \sqrt{n}$$

     Concluímos que o majorante assimptótico para o Loop 1 é $O(\sqrt{n})$.
   - *Loop 2*: Definimos $j(k)$ e $z(k)$ como sendo os valores das variáveis j e z no final da k-ésima iteração do loop 2:
     - $j(k) = 2^k$
     - $z(k) = k$

Resolvemos a condição de paragem em função de k:

$$j(k) = n \Leftrightarrow 2^k = n \Leftrightarrow k = log\ n$$

Concluímos que o majorante assimptótico para o Loop 2 é $O(log\ n)$.

2. Observando que $log\ n \in O(\sqrt{n})$, obtemos a equação do tempo:

$$T(n) = T(n/2) + O(\sqrt{n})$$

Aplicando o Teorema Mestre, concluímos que $T(n) = O(\sqrt{n})$. Parâmetros: $a = 1$, $b = 2$ e $d = 1/2$. Notamos que $log_b\ a = 0 < 1/2$.

II.b) Foi pedido ao prof. Caracol que coordenasse a re-estruturação da licenciatura em Eng. Informática da Universidade Técnica de Caracolândia. A actual coordenação da licenciatura forneceu ao prof. Caracol a lista de unidades curriculares (UCs) da mesma, bem como as dependências que estas têm entre si. Por exemplo, a UC Cálculo II depende da UC Cálculo I.

1. Proponha um algoritmo para determinar se existem dependências circulares entre unidades curriculares e identifique a complexidade assimptótica do mesmo.

2. Admitindo que cada UC tem a duração de um semestre e que os alunos podem fazer um número ilimitado de UCs em cada semestre, proponha um algoritmo para determinar o número mínimo de semestres necessários para completar o curso de Eng. Informática. Identifique a complexidade do algoritmo proposto.

Nota: Não é necessário apresentar o pseudo-código dos algoritmos propostos; é suficiente explicar como estes seriam obtidos a partir dos algoritmos estudados nas aulas.

Solução:

1. Seja $G = (V, E)$ o grafo que representa as dependências entre as unidades curriculares; V é o conjunto de unidades curriculares e E é o conjunto de dependências entres UCs, isto é, $(UC_1, UC_2) \in E$ sse UC_2 depende de UC_1.

   Para determinar se existem dependências circulares entre UCs, temos de verificar se G é acíclico. Para tal, basta efectuar uma DFS em G, verificando que não são encontrados arcos para trás; isto é, que nunca se atinge um vértice cinzento a partir de um vértice branco durante a aplicação da DFS. Complexidade assimptótica: $O(V + E) = O(n + m)$, onde n é o número de UCs e m o número de dependências entre UCs .

2. Para determinar o número mínimo de semestres temos de calcular o tamanho do caminho mais longo no grafo G. Como G é acíclico, o tamanho do caminho mais longo pode ser encontrado em tempo linear usando uma DFS. Primeiro, há que identificar os vértices source do grafo (isto é, vértices sem arcos incidentes), o que pode ser feito em tempo $O(V)$ (admitindo que G é representado com base em listas de adjacências). Depois há que invocar o procedimento DFS_Visit em cada vértice source.

   O procedimento DFS_Visit tem ser modificado por forma a retornar o comprimento da cadeia mais longa com origem no vértice dado como input, de acordo com a equação:

   $$l(u) = \mathbf{max}\{l(v) + 1 \mid v \in G.Adj[u]\}$$

   A alteração sugerida ao algoritmo DFS não altera a complexidade do mesmo, que permanece linear ($O(V + E)$).

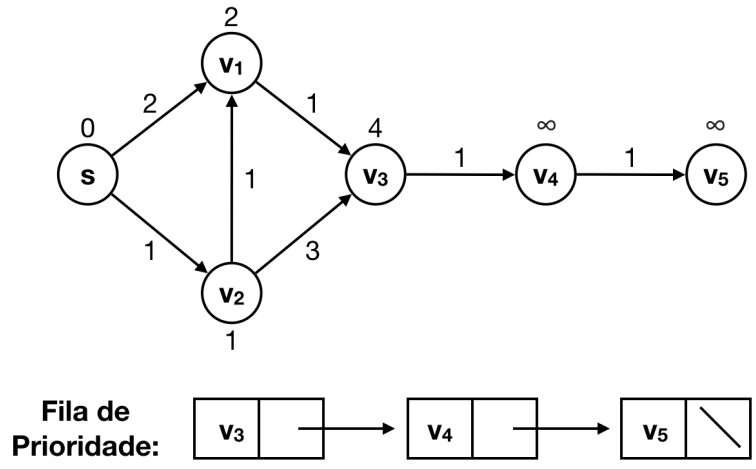
II.c) O aluno X implementou o algoritmo de Dijkstra como forma de preparação para o teste de ASA.

1. Na sua implementação do algoritmo de Dijkstra, o aluno X usou uma fila de prioridade baseada em listas simplesmente ligadas ao invés de utilizar as filas de prioridade baseadas em amontoados estudadas na disciplina. Em particular, a estrutura de dados utilizada pelo aluno X suporta as seguintes operações com as complexidades assimptóticas indicadas:

   - `buildMinHeap`: $O(n.log\, n)$
   - `decreaseKey`: $O(n)$
   - `removeKey`: $O(n)$

   Onde n denota o número de elementos do amontado. Tendo em conta a complexidade assimptótica das operações indicadas, indique a complexidade assimptótica da implementação do algoritmo de Dijkstra produzida pelo aluno X. Deve justificar a resposta.

2. Durante o processo de implementação do algoritmo de Dijkstra, o aluno X reparou que a sua implementação produzia resultados incorrectos. Em baixo ilustra-se um estágio intermédio da execução do algoritmo quando aplicado ao grafo da figura. Cada nó está anotado com a sua estimativa de distância ao nó fonte, s.



   Considerando a etapa em que o algoritmo se encontra, indique os nós para os quais a estimativa de distância está incorrectamente calculada, bem como a estimativa de distância correcta. Justifique a sua resposta apelando ao invariante do algoritmo de Dijkstra.

Solução:

1. Analisamos os dois loops do algoritmo de Dijkstra separadamente:

   - Loop exterior (o loop que retira os vértices da fila de prioridade): $O(V^2)$, $|V|$ iterações, sendo que em cada iteração é efectuada uma operação `removeKey`.
   - Loop interior (o loop que relaxa os arcos que unem o vértice actual aos seus vizinhos): $O(E.V)$, $|E|$ iterações, sendo que em cada iteração pode ser efectuada uma operação `decreaseKey`.

Concluímos que a complexidade assimptótica da implementação é $O(V^2 + E.V)$.

2. A estimativa de distância está incorrectamente calculada no vértice v_3. O valor correcto seria 3. Como o vértice v_1 já não se encontra na fila de prioridade, sabemos que o arco (v_1, v_3) já foi relaxado, pelo que a estimativa de distância de v_3 deveria ser 3 e não 4.

II.d) O presidente do Departamento de Engenharia Informática da Universidade Técnica de Caracolândia está neste momento a fazer a distribuição do serviço docente para o próximo ano lectivo, que será constituído por quatro períodos em vez dos habituais dois semestres. O departamento é integrado por n docentes $D_1, ..., D_n$ e oferece m unidades curriculares $C_1, ..., C_m$, divididas por quatro períodos. Cada unidade curricular C_i está associada ao período $\mathbf{period}(i)$ no qual é leccionada.

Adicionalmente, cada unidade curricular C_i está associada a um inteiro $\mathbf{uc\_need}(i)$ que denota o número de docentes necessários para sua leccionação. Analogamente, cada docente D_j está associado a um inteiro $\mathbf{service}(j)$, que denota o o número máximo de unidades curriculares que o docente pode leccionar, e a um conjunto $\mathbf{competent}(j)$ que contém as unidades curriculares que o docente D_j tem competência para leccionar.

Este ano o presidente do departamento estabeleceu que, dada a nova organização em períodos, cada docente só poderá leccionar uma única unidade curricular em cada período.

Na resolução do exercício pode assumir que: $\sum_{i=1}^{m} \mathbf{uc\_need}(i) \leq 4.n$.

1. Modele o problema de atribuição de docentes a unidades curriculares como um problema de fluxo máximo. A resposta deve incluir o procedimento utilizado para determinar a atribuição de docentes a unidades curriculares.

2. Indique o algoritmo que utilizaria para a calcular o fluxo máximo, bem como a respectiva complexidade assimptótica medida em função dos parâmetros do problema (número de docentes, n, e número de unidades curriculares, m). De entre os algoritmos de fluxo estudados nas aulas deve escolher aquele que garanta a complexidade assimptótica mais baixa para o problema em questão.
   Nota: A resposta deverá necessariamente incluir as expressões que definem o número de vértices e de arcos da rede de fluxo proposta ($|V|$ e $|E|$, respectivamente) em função dos parâmetros do problema.

Solução:

1. *Construção da rede de fluxo:* $G = (V, E, c, s, t)$. Na construção da rede de fluxo consideramos um vértice por unidade curricular, cinco vértices por docente e dois vértices adicionais s e t, respectivamente a fonte e o sumidouro. Associamos a cada docente D_j cinco vértices, respectivamente designados por: D_j^0, D_j^1, D_j^2, D_j^3 e D_j^4. Intuitivamente, o vértice D_j^0 serve para seleccionar a carga lectiva total atribuída ao docente D_j, enquanto os vértices D_j^k, com $1 \leq k \leq 4$, servem para seleccionar a UC que o docente D_j irá leccionar no período k. Formalmente:

   - $V = \{s, t\} \cup \{D_j^k \mid 1 \leq i \leq n \ \wedge \ 0 \leq k \leq 4\} \cup \{C_i \mid 1 \leq i \leq m\}$
   - $$\begin{aligned} E = \quad & \{(s, D_j^0, \mathbf{service}(j)) \mid 1 \leq j \leq n\} \\ & \quad D_j \text{ pode dar no máximo } \mathbf{service}(j) \text{ UCs} \\ & \cup \{(D_j, D_j^k, 1) \mid 1 \leq j \leq n \ \wedge \ 1 \leq k \leq 4\} \\ & \quad D_j \text{ pode dar uma UC por período} \\ & \cup \{(D_j^k, C_i, 1) \mid \mathbf{period}(i) = k \ \wedge \ i \in \mathbf{competent}(j)\} \\ & \quad D_j \text{ quer dar a UC } C_i \\ & \cup \{(C_i, t, \mathbf{uc\_need}(i)) \mid 1 \leq i \leq m\} \\ & \quad C_i \text{ requer } \mathbf{uc\_need}(i) \text{ docentes} \end{aligned}$$

   A atribuição é satisfazível se o fluxo máximo for $\sum_{i=1}^{m} \mathbf{uc\_need}(i)$. O docente j dará a UC i no período k se $f^*(D_j^k, C_i) = 1$.

   - $|V| = m + 5.n + 2 = O(n + m)$

- $|E| \leq n + 4.n + n.m + m = O(n.m)$
- $|f^*| \leq \sum_{i=1}^{m} \textbf{uc\_need}(i) \leq 4.n = O(n)$
- Edmonds Karp (upper bound de FF): $O(|f^*|.E) = O(n.n.m) = O(n^2.m)$
- Edmonds Karp (upper bound EK): $O(E^2.V) = O(n^2.m^2.(n+m)) = O(n^3.m^2 + m^3.n^2)$
- Relabel-To-Front: $O((n+m)^3) = O(n^3 + m^3)$

Utilizaria o algoritmo de Edmonds Karp.