UNIFEI - UNIVERSIDADE FEDERAL DE ITAJUBÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

SIN110 - ALGORITMOS E GRAFOS

RESOLUÇÃO DOS EXERCÍCIOS E13 DO DIA 20/11/2015

ITAJUBÁ

2015

Exercícios E13 – 20/11/15

Aluna: Karen Dantas **Número de matrícula:** 31243

1) O algoritmo faz 2n+1 atribuições, sendo uma realizada na linha 1 e 2n são realizadas na linha 2 e 3.

O algoritmo abaixo é uma adaptação do algoritmo dado. Nele, são realizadas 'n' atribuições as quais são realizadas na linha 4. Considerei que S é uma variável global com valor inicial igual a 0.

```
Soma (n)
1. se n=0
2.      então devolve S
3. senão
4.      S ← n + Soma(n-1)
```

---

2) O algoritmo que retorna o inverso da matriz através da regra dos cofatores deve seguir a seguinte expressão matemática:

$$A^{-1} = \frac{\left(Cof\ (A)\right)^{t}}{Det\ (A)}$$

Na qual, divide-se a transposta de sua cofatora pelo determinante da matriz dada, sendo seu determinante diferente de 0.

---

3) Algoritmo Shakesort recursivo que ordena em ordem decrescente:

```
ShakeSort_Recursivo (Vet, n)
1       i ← n
2       para j ← e ate i faça
3               se VetA[j] < Vet[j+1]
4                       então troca (Vet[j], Vet[j+1])
5       para j ← i ate e + 1 faça
6               se Vet[j-1] < Vet[j]
7                       então troca (Vet[j-1], Vet[j])
8       e ← e + 1
9       se i ≥ e
10              então ShakeSort_Recursivo (Vet, n-1)
11      senão
12              devolve Vet
```

Para se executar o algoritmo acima deve-se considerar que a variável 'e' é uma variável global com valor inicial 1. E o 'n' passado para a função deve ser n-1.

Correção:

Invariante: A cada recursão da linha 10, as variáveis contidas do lado esquerdo para o direito do vetor em A[1 ... n] serão movidas, se preciso, para seu lugar correto no vetor através do laço da linha 2. E, na linha 5, serão movidos para suas posições corretas os dados contidos do lado direito para o esquerdo do vetor em A[n-1 ... 2]. Seus espaços de percussão são determinados pelas variáveis 'i' e 'e', e que, consequentemente, se tornam cada vez menores.

Início: O invariante é válido na linha 2 quando se é verificado na linha 3 se o dado contido em A[1 ... n] está na sua posição correta e, se não estiver, é movido para sua posição correta. Na linha 5, o mesmo ocorre na linha 6, só que com dados contidos em A[n-1 ... 2].

Manutenção: O invariante é mantido através da linha 1, na qual, é decrementada a variável 'i' que tem que ser maior ou igual à variável 'e' que é incrementada na linha 8. O invariante também é mantido pelos laços das linhas 2 e 5 que são responsáveis por ordenar o vetor em ordem decrescente chamando a função troca, se necessário, e cujas condições de parada dependem dos valores de 'i' e 'e'. Logo, suas condições de parada também são válidas.

Término: Quando a condição da linha 9 não for satisfeita, o algoritmo termina sua execução. O vetor foi percorrido, em sentido bidirecional, pelos laços da linha 2 e 5 e seus dados foram ordenados decrescentemente. Portanto, o algoritmo está correto.

Complexidade do algoritmo: $O(n^2)$.

---

4) Algoritmo que confere se grafo é completo através de sua matriz de adjacência:

```
Confere (M, n)
1    para de i ← 1 até n faça
2            contador ← 0
3            para de j ← 1 até n faça
4                    se M[i, j] = 1
5                            então contador ← contador + 1
6            se contador != n-1
7                    então devolve 0
8    devolve 1
```

A função Confere recebe a matriz e um 'n' que indica o número de linhas e colunas da matriz (a matriz de adjacências possui o mesmo número de linhas e colunas). Ela percorre a matriz verificando linha por linha quantos vértices têm ligado no vértice da linha 'i', atribuindo essa quantidade à variável 'contador'. No fim do percurso de uma linha da matriz, se a variável 'contador' for diferente de n-1 (número máximo de vértices que podem ser ligados a ele) significa que o grafo não é completo, assim, a função retorna 0 (falso). Caso contrário, se for concluído que o grafo é completo, a função no fim de sua execução retorna 1 (verdadeiro).

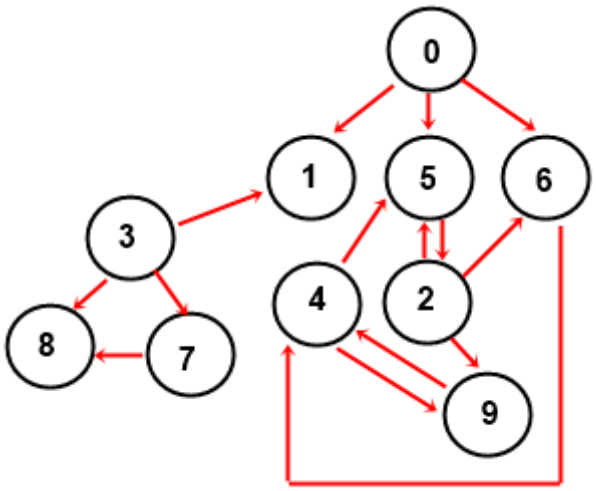
---

5) Lista de Adjacências:

0 → 1 → 5 → 6
1
2 → 9 → 5 → 6
3 → 7 → 1 → 8

4 → 5 → 9
5 → 2
6 → 4
7 → 8
8
9 → 4

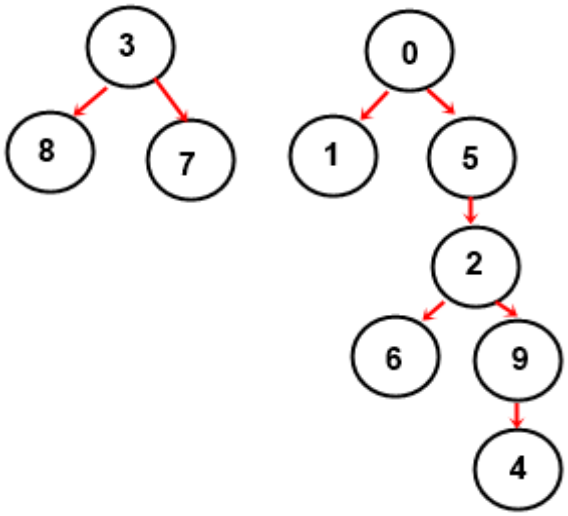
Dígrafo:



Busca em profundidade - DFS:

Vértice	Cor(u)	Predecessor(u)	Descoberta (u)	Fim (u)
0	~~b~~ /~~c~~ /~~p~~	-	1	14
1	~~b~~ /~~c~~ /~~p~~	0	2	3
2	~~b~~ /~~c~~ /~~p~~	5	5	12
3	~~b~~ /~~c~~ /~~p~~	-	15	20
4	~~b~~ /~~c~~ /~~p~~	9	7	8
5	~~b~~ /~~c~~ /~~p~~	0	4	13
6	~~b~~ /~~c~~ /~~p~~	2	10	11
7	~~b~~ /~~c~~ /~~p~~	3	18	19
8	~~b~~ /~~c~~ /~~p~~	3	16	17
9	~~b~~ /~~c~~ /~~p~~	2	6	9

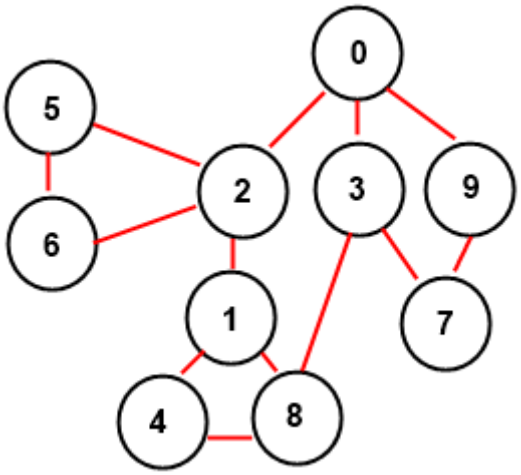
Floresta de arborescências:



6) Lista de Adjacências:

0 - 2 – 3 - 9
1 – 4 – 8 - 2
2 - 5
3 - 7 - 8
4 - 8
5 - 6
6 - 2
7 - 9
8
9 - 7

Grafo:



Ao aplicar os algoritmos modificados da busca em profundidade obtêm-se que:

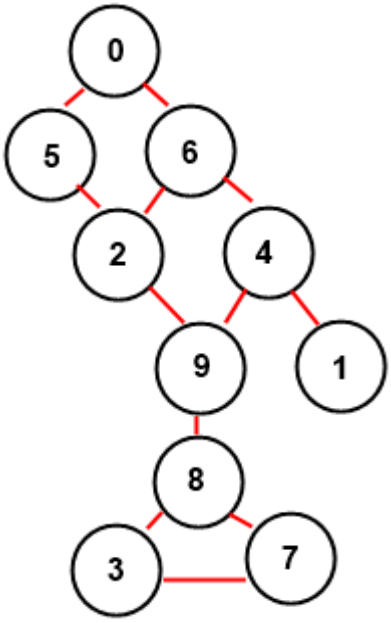
a) O número de componentes é 1.
b) Há ciclos no grafo. Observando-se o grafo pode-se notar que são eles: 5-6-2-5; 1-4-8-1; 0-3-7-9-0; 0-3-8-1-2-0.
c) Não há pontes.

---

7) Matriz de adjacências:

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	1	1	0	0	1
3	0	0	0	0	0	0	0	1	1	0
4	0	1	0	0	0	0	1	0	0	1
5	1	0	1	0	0	0	0	0	0	0
6	1	0	1	0	1	0	0	0	0	0
7	0	0	0	1	0	0	0	0	1	0
8	0	0	0	1	0	0	0	1	0	1
9	0	0	1	0	1	0	0	0	1	0

Grafo:



Distância do vértice 0 a cada um dos vértices do grafo usando-se busca em largura a partir do vértice 0:

Vértice	Cor(u)	Predecessor(u)	Dist(u)
0	~~b~~ /~~c~~ /~~p~~	-	0
1	~~b~~ /~~c~~ /~~p~~	4	3
2	~~b~~ /~~c~~ /~~p~~	5	2
3	~~b~~ /~~c~~ /~~p~~	8	5
4	~~b~~ /~~c~~ /~~p~~	6	2
5	~~b~~ /~~c~~ /~~p~~	0	1
6	~~b~~ /~~c~~ /~~p~~	0	1
7	~~b~~ /~~c~~ /~~p~~	8	5
8	~~b~~ /~~c~~ /~~p~~	9	4
9	~~b~~ /~~c~~ /~~p~~	2	3

Filas:

u = 0 0-5-6

u = 5 5-6-2

u = 6 6-2-4

u = 2 2-4-9

u = 4 4-9-1

u = 9 9-1-8

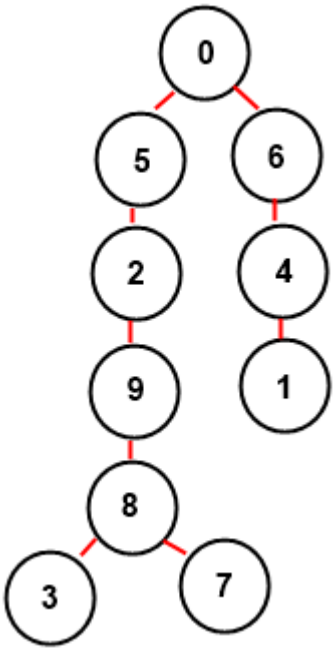
u = 1 1-8

u = 8 8-3-7

u = 3 3-7

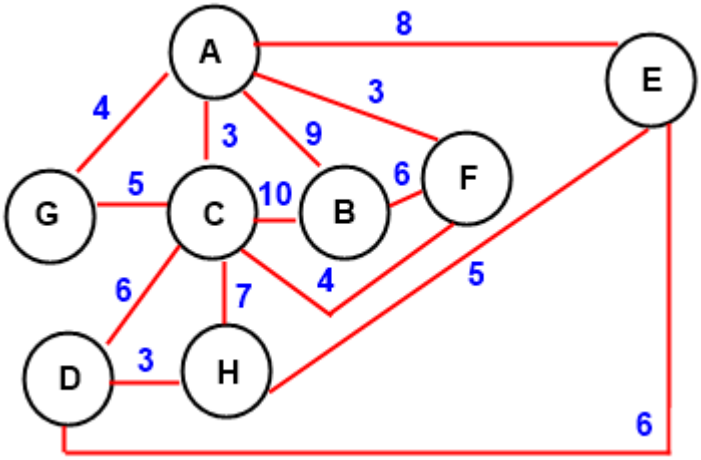
u = 7 7

Arborescência:



---

8) Grafo:



Solução através do algoritmo Prim:

Filas: A – B - C – D – E - F – G – H

C - F – G – E - B – D – H

F – G – D – H - E - B

G – D – B - H - E

D – B - H – E

H - B – E

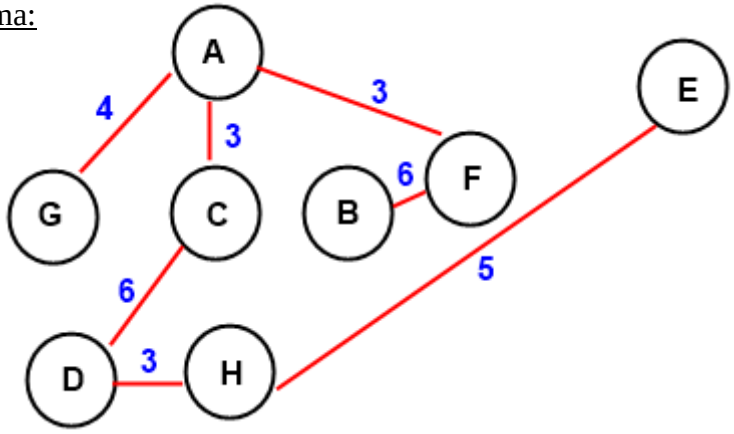
E – B

B

Vértice	Cor(u)	Predecessor(u)	Chave(u)
A	~~B~~/P	A	0
B	~~B~~/P	~~A~~ / F	~~9~~ / 6
C	~~B~~/P	A	3
D	~~B~~/P	C	6
E	~~B~~/P	~~A~~ / ~~D~~ / H	~~8~~ / ~~6~~ / 5
F	~~B~~/P	A	3
G	~~B~~/P	A	4
H	~~B~~/P	~~C~~ / D	~~7~~ / 3

Total= 0+6+3+6+5+3+4+3 = 30

Árvore geradora mínima:



Na árvore acima está representado o sistema de conexão o qual apresenta o menor risco total possível totalizando-se 30.

---

9) Algoritmo:

```
Confere_num_repetido (Vet, n)
1   aux1 ← Vet[1]
2   aux2 ← Vet[n]
3   i ← 2
4   j ← n-1
5   enquanto (i != (n/2) +1) e (j != (n/2))
6           se Vet[i] = aux1 ou Vet[j] = aux2
7                   então devolve 1
8           i ← i+1
9           j ← j-1
10          aux1 ← Vet[i]
11          aux2 ← Vet[j]
12  devolve 0
```

Para se executar o algoritmo acima, o vetor deve estar ordenado.

---

10) Ideia de solução: Criar dois vetores, um para a máquina A e outro para a máquina B. Para cada ‘i’, colocar no vetor de A a unidade de tempo que seria gasto para executar a ‘i’ tarefa. E o mesmo ocorre para o vetor de B.

Para cada ‘i’, comparar a unidade de tempo na posição ‘i’ do vetor de A com a do vetor de B. Se $A[i] \geq B[i]$, a tarefa ‘i’ será executada pela máquina B. Senão, se $A[i] < B[i]$, a tarefa ‘i’ será executada pela máquina A.

---

11) A heurística 1 não irá devolver o menor número possível de latas, pois, por exemplo, se um objeto avaliado tiver peso 0,8 e a lata que tem mais espaço livre tenha um peso 0,1, a lata ficará com um total de peso de 0,9. Pode-se notar que não foi avaliado o peso dos outros objetos e se houvesse um objeto com peso igual a 0,9, a lata ficaria com um peso total igual a 1 que é o máximo permitido e nenhum espaço seria desperdiçado. Logo, essa heurística pode não obter o número mínimo possível de latas.

Já a heurística 2 irá resolver o problema, pois, sempre pegará o objeto de maior peso e o colocará na lata com maior espaço livre ou pegando um nova lata caso não seja possível inseri-lo. Logo, essa heurística em conjunto com a heurística 1 irão devolver o número mínimo possível de latas.
