

MAC0323 Algoritmos de Estruturas de Dados II

Prova 3 – 25 de junho de 2019

Nome: _____

Assinatura: _____

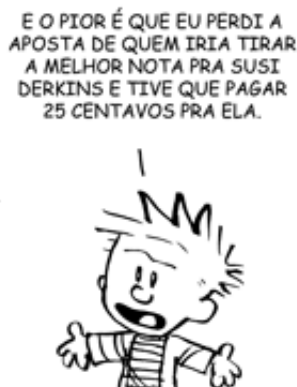
Nº USP: _____

Instruções:

1. Não destaque as folhas deste caderno. A prova pode ser feita a lápis.
2. A prova consta de 10 questões. Verifique antes de começar a prova se o seu caderno está completo.
3. Cuidado com a legibilidade.
4. Não é permitido o uso de folhas avulsas para rascunho, a consulta a livros, apontamentos, colegas ou equipamentos eletrônicos. Desligue o seu celular e qualquer equipamento que possa perturbar o andamento da prova.

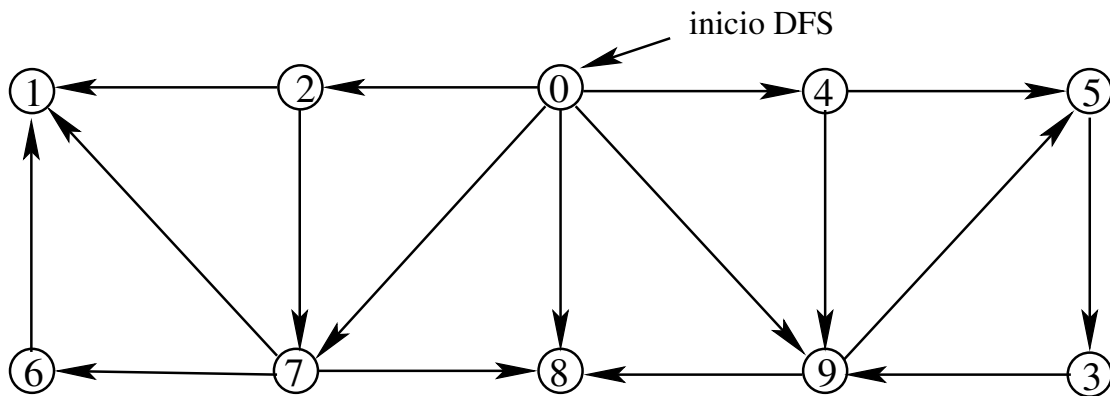
DURAÇÃO DA PROVA: 100 minutos

Questão	Valor	Nota
1	1,0	
2	1,0	
3	1,0	
4	1,0	
5	1,0	
6	1,0	
7	1,0	
8	1,0	
9	1,0	
10	1,0	
Total	10,0	



1. DFS (1 ponto)

Considere o digrafo a seguir. Suponha que no vetor de listas de adjacência que representa o digrafo as listas estão ordenadas. Assim, por exemplo, quando **examinando o vértice 7**, considere o arco $7 \rightarrow 1$ antes de $7 \rightarrow 6$ ou $7 \rightarrow 8$.



(a) Execute uma busca em profundidade no digrafo a partir do vértice 0.

Liste os vértices em **pré-ordem**:

Liste os vértices em **pós-ordem**:

Liste os **arcos da arborescência**:

Liste os **arcos de retorno**:

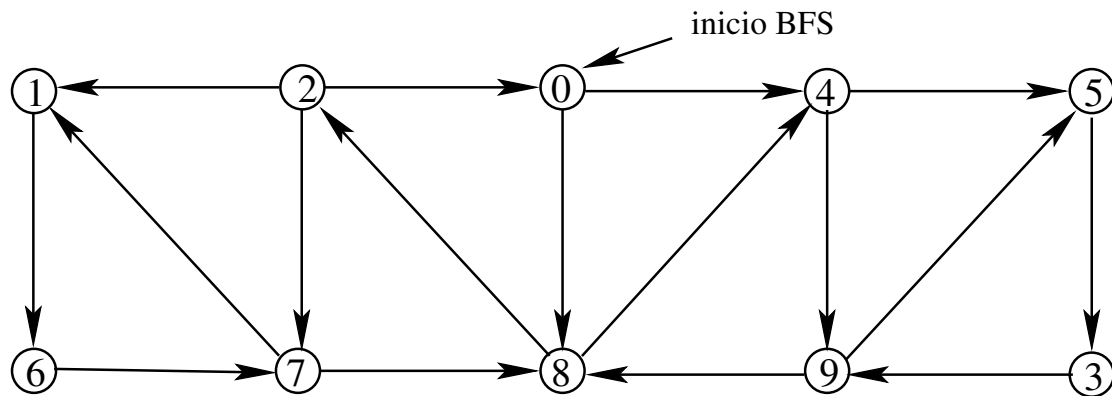
Liste os **arcos descendentes**:

Liste os **arcos cruzados**:

(b) Esse digrafo tem uma ordenação topológica? Se tem, escreva uma ordenação topológica do digrafo. Se não tem, explique sucintamente a razão.

2. **BFS** (1 ponto)

Considere o digrafo a seguir. Suponha que no vetor de listas de adjacência que representa o digrafo as listas estão ordenadas. Assim, por exemplo, quando **examinando o vértice 2**, considere o arco $2 \rightarrow 0$ antes de $2 \rightarrow 1$ ou $2 \rightarrow 7$.

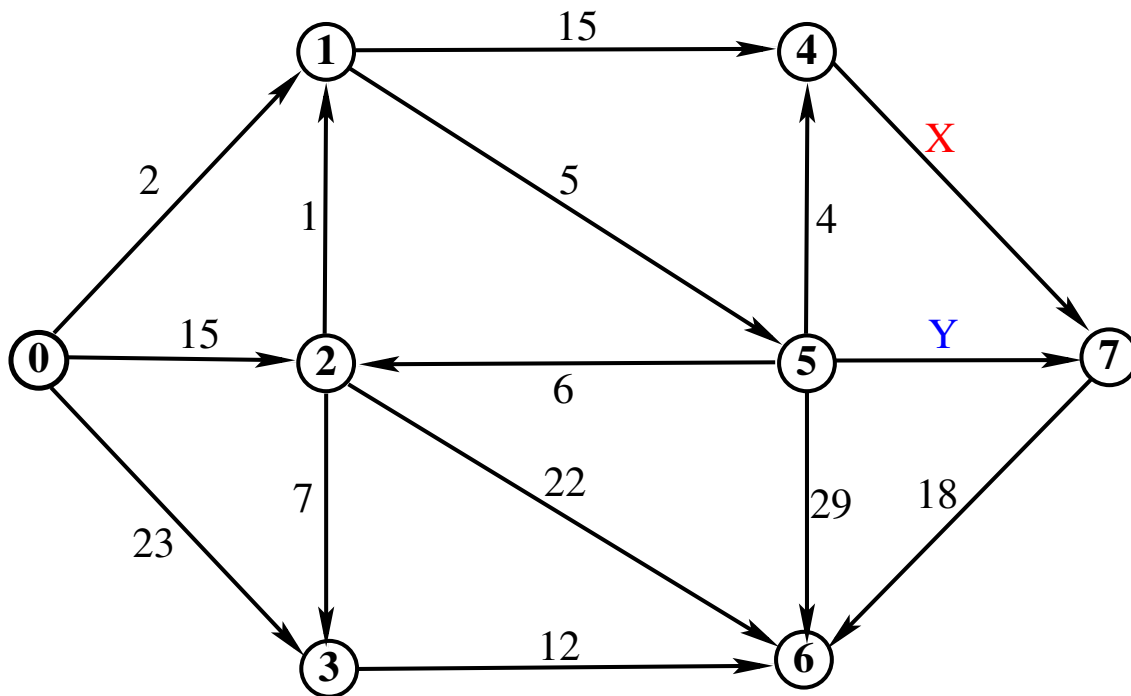


Execute uma busca em largura no digrafo a partir do vértice 0.

Liste os vértices na ordem em que são **inseridos na fila**: _____

3. Dijkstra (1 ponto)

Suponha que você está executando o algoritmo de **Dijkstra** (=DijkstraSP) no digrafo a seguir. O digrafo tem comprimentos nos arcos e 0 é o vértice origem/inicial usado pelo algoritmo.



As tabelas a seguir mostram os valores dos vetores `distTo[]` e `edgeTo[]` imediatamente após o vértice 4 ter sido removido da fila priorizada (=IndexMinPQ) e **examinado**.

v	distTo[]	edgeTo[]
0	0.0	null
1	2.0	0→1
2	13.0	5→2
3	23.0	0→3
4	11.0	5→4
5	7.0	1→5
6	36.0	5→6
7	19.0	4→7

(a) Dê a ordem em que os 4 primeiros vértices são removidos da fila priorizada.

--	--	--	--

(b) Quais os possíveis valores para o comprimento **X**?

(c) Quais os possíveis valores para o comprimento **Y**?

(d) Qual é o próximo vértice a ser removido da fila priorizada e examinado?

(e) Na tabela a seguir, preencha **apenas as entradas** de `distTo[]` e `edgeTo[]` que são diferentes as da página anterior depois do próximo vértice do item (d) ser examinado.

v	distTo[]	edgeTo[]
0		
1		
2		
3		
4		
5		
6		
7		

4. Propriedades de caminhos mínimos (1 ponto)

Seja G um **digrafo acíclico** (=DAG) com comprimentos positivos em seus arcos. Suponha que todo vértice é alcançável a partir de um vértice origem s . Para cada afirmação à esquerda, identifique se ela é uma propriedade do **algoritmo de Dijkstra** (DijkstraSP) ou do algoritmo de caminhos mínimos para DAGs (=AcyclicSP) escrevendo no quadrado a letra da **melhor correspondência** à direita.

☐ Se G possui um arco $v \rightarrow w$, então o vértice v é examinado antes do vértice w . A. DijkstraSP

☐ Cada vértice é examinado no máximo uma vez. B. AcyclicSP

☐ Se o comprimento de um caminho mínimo de s a v é menor que o comprimento de um caminho mínimo de s a w , então o vértice v é examinado antes do vértice w . C. Ambas A e B

☐ Imediatamente após examinar um arco $v \rightarrow w$, $\text{distTo}[w]$ é o menor comprimento caminho de s a w . D. Nem A nem B

☐ Após examinar um arco qualquer, para cada vértice v , o valor de $\text{distTo}[v]$ ou não é alterado ou diminui.

5. Ordenação de strings (1 ponto)

A coluna 0, mais à esquerda, contém strings a serem ordenadas. A coluna 8, mais à direita, contém as strings já ordenadas. As colunas de 1 a 7 contêm as strings em um passo intermediário de algum dos algoritmos listados a seguir. Para cada um dos algoritmos listados a seguir **indique a coluna** entre 1 e 7 que corresponde ao estado das strings em um de seus passos intermediários.

algoritmo	coluna(s)
LSD radix sort	
MSD radix sort	
3-way radix quicksort	

KISS	ABBA	ENYA	ABBA	ENYA	ACDC	SOAD	SADE	ABBA
ENYA	ACDC	INXS	ACDC	ABBA	ABBA	WHAM	CAKE	ACDC
INXS	AQUA	DIDO	AQUA	AQUA	AQUA	ABBA	CARS	AQUA
STYX	BECK	CARS	BECK	ACDC	BUSH	MOBY	JAYZ	BECK
SOAD	BLUR	ACDC	BLUR	SOAD	BLUR	BECK	ABBA	BLUR
ACDC	BUSH	FUEL	BUSH	CAKE	BECK	ACDC	ACDC	BUSH
KORN	CAKE	BUSH	CAKE	MUSE	CAKE	SADE	BECK	CAKE
FUEL	CARS	ABBA	CARS	HOLE	CARS	DIDO	WHAM	CARS
BUSH	DIDO	AQUA	DIDO	SADE	DIDO	FUEL	DIDO	DIDO
ABBA	ENYA	CAKE	ENYA	BUSH	ENYA	CAKE	KISS	ENYA
WHAM	FUEL	BLUR	FUEL	RUSH	FUEL	HOLE	BLUR	FUEL
CAKE	HOLE	JAYZ	HOLE	BECK	HOLE	TSOL	INXS	HOLE
BLUR	INXS	BECK	INXS	FUEL	INXS	KORN	ENYA	INXS
MUSE	JAYZ	HOLE	JAYZ	TSOL	JAYZ	CARS	SOAD	JAYZ
BECK	KISS	KORN	KISS	WHAM	KISS	MUSE	MOBY	KISS
MOBY	KORN	KISS	KORN	KORN	KORN	BUSH	HOLE	KORN
HOLE	MUSE	TSOL	TSOL	DIDO	MUSE	RUSH	KORN	MOBY
TSOL	MOBY	MOBY	MOBY	BLUR	MOBY	KISS	AQUA	MUSE
JAYZ	RUSH	MUSE	MUSE	KISS	RUSH	AQUA	TSOL	RUSH
AQUA	STYX	SADE	SADE	INXS	STYX	BLUR	STYX	SADE
SADE	SOAD	WHAM	WHAM	CARS	SOAD	INXS	FUEL	SOAD
CARS	SADE	SOAD	SOAD	STYX	SADE	ENYA	MUSE	STYX
DIDO	TSOL	RUSH	RUSH	MOBY	TSOL	STYX	BUSH	TSOL
RUSH	WHAM	STYX	STYX	JAYZ	WHAM	JAYZ	RUSH	WHAM
0	1	2	3	4	5	6	7	8

6. **KMP** e **DFA** (1 ponto)

(a) Desenhe a representação gráfica do **autômato finito determinístico** (=DFA) do algoritmo de Knuth-Morris-Pratt (=KMP) correspondente a string "CCACCACB" sobre o alfabeto {A, B, C}.

(b) Preencha a matriz `dfa[][]` que representa o autômato.

	0	1	2	3	4	5	6	7
	C	C	A	C	C	A	C	B
A								
B								
C								

Considere a execução do algoritmo de Boyer-Moore baseada apenas na heurística do “mau-caractere” (*“bad-character heuristic”*) para encontrar o padrão (**pat**)

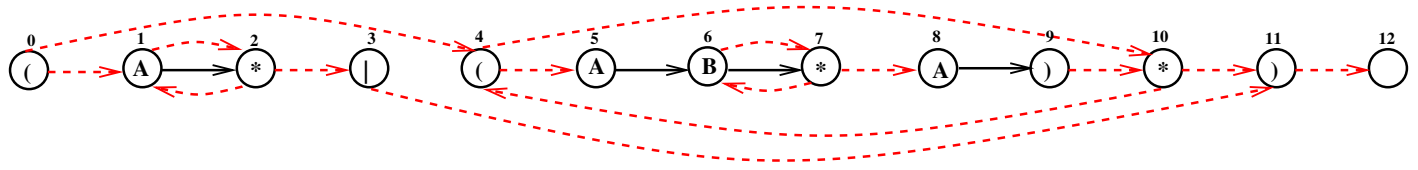
no texto (**txt**)

Mostre a simulação do algoritmo na tabela a seguir. **Alinhe o padrão** com as posições do texto que poderão ser comparadas. **Circule cada caractere** do padrão que é comparado com o correspondente caractere do texto.

[illegible]

8. Expressões regulares e NFA (1 ponto)

O digrafo a seguir representa um **autômato finito não-determinístico** (=NFA) construída pelo algoritmo que transforma expressões regulares em NFAs. No digrafo, as ϵ -transições estão representadas com arcos tracejados.



(a) Qual é a expressão regular representada pelo digrafo?

(b) Suponha que soletramos os caracteres da string "AAAAAAA" andando no NFA acima. Ao final, em quais estados podemos estar? Circule os possíveis estados.

0 1 2 3 4 5 6 7 8 9 10 11 12

(c) Suponha que desejamos construir um NFA para representar a expressão regular

$(A * \mid (A B * A) +)$

onde o operador '+' significa **uma ou mais cópias**. Qual é a menor modificação que devemos fazer no NFA acima para que reconheça a linguagem determinada por essa expressão regular?

9. Burrows-Wheeler (1 ponto)

(a) Qual é o resultado da transformada de Burrows-Wheeler da string "BDABACAC".

Sinta-se à vontade para usar a tabela abaixo como rascunho.

[illegible]

(b) Qual é a inversa da transformada de Burrows-Wheeler de "4 DADCCCB".

--	--	--	--	--	--	--	--

Sinta-se à vontade para usar a tabela abaixo como rascunho.

[illegible]

10. LZW e compressão (1 ponto)

Considere o algoritmo LZW para codificar strings sobre o alfabeto ASCII de 7 bits que possui $R = 128$ símbolos. Suponha que cada código LZW é de $W = 8$ bits. Utilizando esse algoritmo LZW para codificar uma string obtivemos seguinte codificação em hexadecimal

43 41 42 42 82 43 81 41 87 82 80 .

O código 80 em hexadecimal (128 na base decimal) é reservado para significar *fim de arquivo* (=EOF). Obtenha a string original e o correspondente dicionário de códigos.

codificação	43	41	42	42	82	43	81	41	87	82	80
string	C										EOF

dicionário	
string	código
...	...
A	41
B	42
C	43
D	44
E	45
...	...
EOF	80 (= R)