UNIVERSIDADE FEDERAL DO MARANHÃO
DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS II (DEIN0083) 2024.2 P1

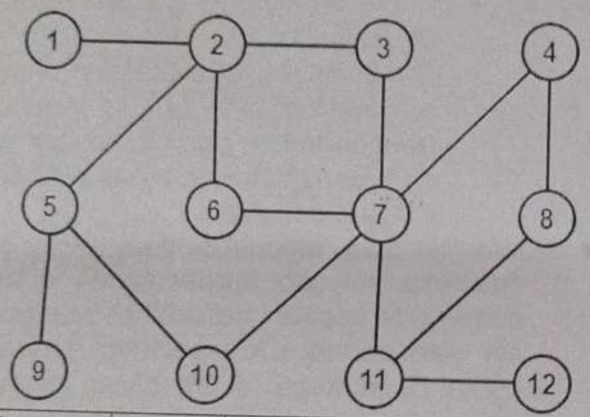
Professor(a): João Dallyson Sousa de Almeida

Data: 26/02/2025

Matrícula: 2023041536 **Aluno:** [illegible]

3ª Avaliação (90%)

1) Execute o algoritmo BFS no grafo ilustrado abaixo, com o vértice $s = 10$ como origem. Apresente uma tabela com a representação da lista de adjacência, as cores, as distâncias da origem e os pais na árvore BFS. Liste os vértices na ordem em que eles entram na fila. Desenhe a árvore BFS resultante.



Vértice	Adjac.	Cor	Distân.	Pai (Pi)
. . ..				

2) (20%) Escreva (em java ou pseudocódigo) uma função dfs(G,u,v) para fazer uma busca em profundidade a partir do nó u até v no grafo G. A função deve retornar o número de nós que foram visitados durante a busca.

3) (20%) A prefeitura de São Luís planeja a construção de uma rede de transporte público, por linhas de metrô, para conectar diferentes bairros da cidade. O objetivo é minimizar o custo total de construção das linhas, garantindo que todos os bairros estejam conectados diretamente ou indiretamente. Assim, cada bairro terá uma estação que precisará ser conectada. A tabela abaixo representa os bairros (estações) e as possíveis rotas com seus respectivos custos de construção (em milhões de reais):

Conexão	Custo R$ milhões
A-B	5
A-C	3
A-D	7
A-F	6
B-C	2
B-D	4
B-F	7
C-D	6
C-E	9
C-F	4
D-E	2

Dado o problema, apresente o algoritmo de grafo adequado ao problema. Descreva a solução passo a passo, apresente as linhas de metrô que devem ser construídas e o custo total da construção da rede de metrô.

BFS é adequada para problemas em que a menor distância é requerida, como as [illegible]
de amizade em redes sociais ou sistemas.

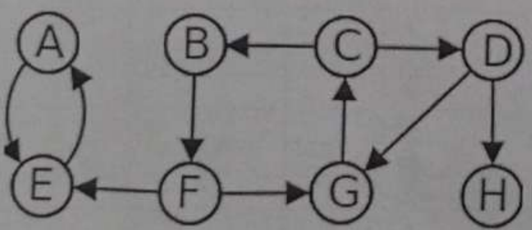
melhor para

[illegible] completo,

Guilherme [illegible]

UNIVERSIDADE FEDERAL DO MARANHÃO
DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS II (DEIN0083) 2024.2 P1

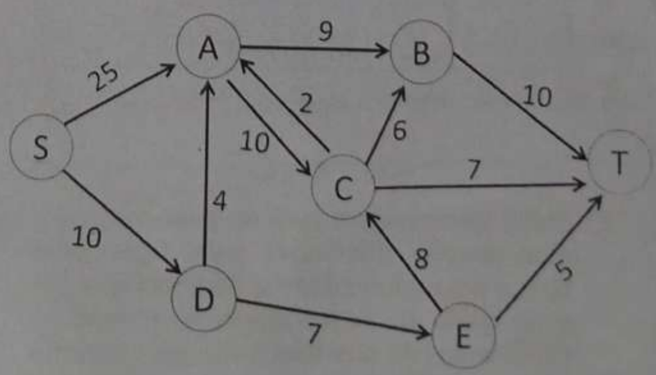
4) (20%) Considere uma rede de tráfego aéreo, na qual os aeroportos são conectados por rotas de voo. Essas rotas podem ser representadas como um grafo dirigido, onde: vértices são os aeroportos e arestas são as rotas de voos entre os aeroportos. Um componente fortemente conectado (CFC) representa um grupo de aeroportos onde é possível viajar de qualquer aeroporto no grupo para qualquer outro aeroporto no mesmo grupo, direta ou indiretamente. Esses grupos podem representar hubs regionais ou redes de aeroportos fortemente interconectados. Sua tarefa é detectar os CFCs, identificando os hubs de transporte. O resultado é importante para otimizar rotas e planejar conexões eficientes. Assim, demonstre a execução do algoritmo de detecção de componentes fortemente conectados no grafo abaixo. Apresente o número de componentes fortemente conectados e os nós presentes em cada componente.



5) (20%) A Lanchonete do Cabañas (CCET) planeja inovar no atendimento e entrega dos lanches. Para isso, pretende utilizar drones para entregar pacotes de lanches saindo da cozinha central para um ponto de entrega no CCET. Cada drone terá uma capacidade máxima de carga e um limite de distância que pode percorrer. O objetivo é maximizar o número de pacotes de lanches entregues por dia, respeitando as capacidades dos drones e as restrições de distância. Dado o grafo abaixo que representa:

- Fonte (origem): Cozinha central (S).
- Sumidouro (destino): Pontos de entrega de lanche (T).
- Arestas: Rotas de voo dos drones.
- Ponto de recarga e troca de drone (A, B, C, D e E)
- Capacidades das arestas: Número máximo de pacotes que podem ser transportados por drone em cada rota, considerando a capacidade de carga e a distância.

Informe qual algoritmo será utilizado e descreva solução apresentando o maior número de pacotes de lanches que podem ser transportados da cozinha central ao ponto de entrega. Além disso, mostre o fluxo possível de pacotes em cada rota. Descreva a sua solução passo a passo.



Guilherme Bonia Nascimento 6,1 26 / 02 / 25

2) DFS(G, u, v) // considera a lista de adjacências em ordem crescente.

0,5 não usou pilha

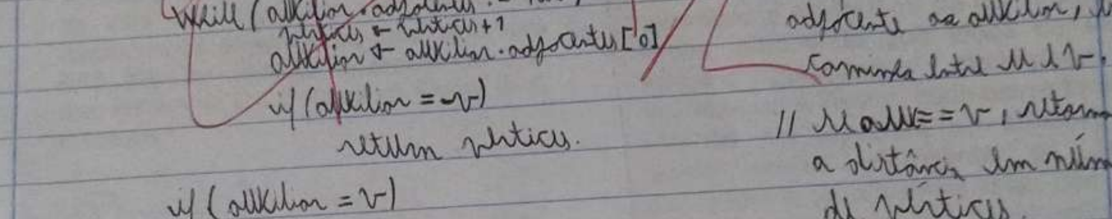
```
vertices ← 0

while (u.adjacentes != nil)                  // enquanto houver vértice adjacente a u,
    vertices ← vertices+1
    auxiliar ← u.adjacentes[0]
    while (auxiliar.adjacentes != nil)       // enquanto houver vértice adjacente ao auxiliar, busca caminho entre u e v.
        vertices ← vertices+1
        auxiliar ← auxiliar.adjacentes[0]
        if (auxiliar = v)                    // se aux == v, retorna a distância em número de vértices.
            return vertices.
    if (auxiliar = v)
        return vertices
    else
        return 0                             // se o vértice u = v ou v não foi achado, retorna a distância 0.

if (u = v)
    return 0.
```

0,5

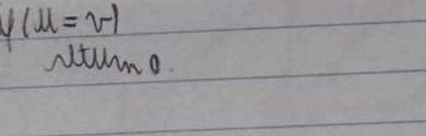
4) Usando DFS para achar componentes conectados, começando em A e ~~assumindo ordem alfabética~~ na lista de adjacências. • = vértice visitado (cinza)



- árvore: (A,E), (B,F), (F,G), (G,C), (C,D), (D,H), - retorno: (E,A), (C,B), (D,G), - cruzamento: (F,E)

~~Existem três componentes fortemente conectados no grafo, sendo eles: A ⇄ E, B → F → G → C → B, e G → C → D.~~

São apenas 2



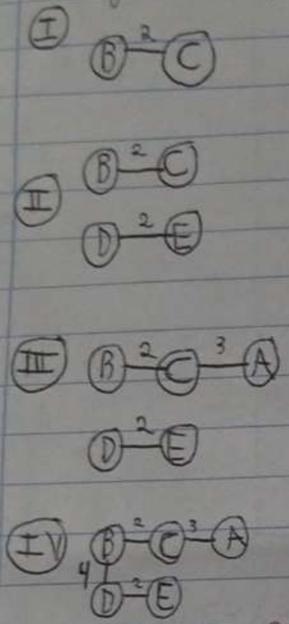
Deveria executar DFS novamente no G^T

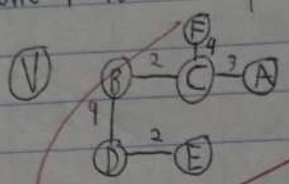
de amizade em redes sociais, ou sistemas de navegação, enquanto DFS é melhor para problemas de maior profundidade ou de teste de todas possibilidades numa completa, como jogos de tabuleiro ou labirintos...

Guilherme Borsa Nascimento 26/02/25

2,0

3) O algoritmo de Kruskal pode ser usado para resolver o problema, ele gera a árvore ~~com~~ menor custo que conecta todos vértices (MST)



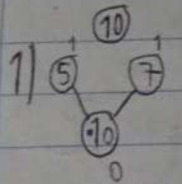


Custo total = 2+2+3+4+4 = 15 milhões de reais.

2,0

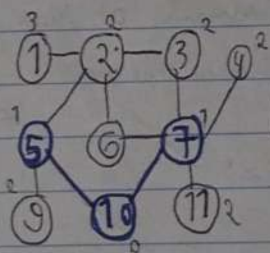
• = visitado.

1)

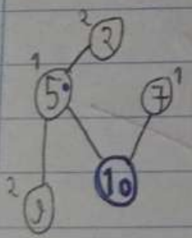


[10]

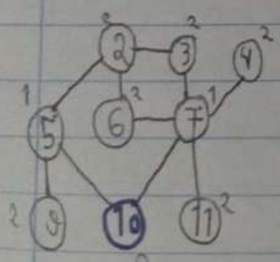
[5,7]



[9,3,4,6,11,1]

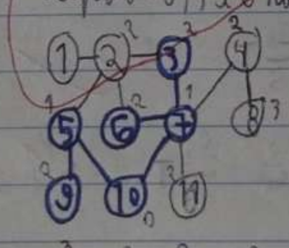


[7,2,9]

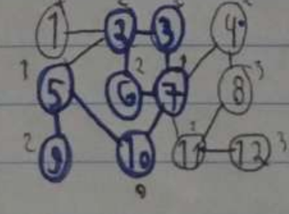


[2,9,3,4,6,11]

Explorar 9, 3 e 6 não descobre vértice.



[11,1,8]



[1,8,12]

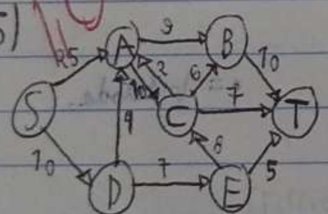
de amizade em ... problemas de maior profundidade ou de teste de todos ... como jogos de tabuleiro ou implementações e arranjos.

Guilherme Borba Nascimento

26/02/25

vértice	adjacência	cor	distância	pai
10	5, 7	preta	0	nil
5	2, 9, 10	preta	1	10
7	3, 4, 6, 10, 11	preta	1	10
2	1, 3, 5, 6	preta	2	5
9	5	preta	2	5
3	2, 7	preta	2	7
4	7, 8	preta	2	7
6	2, 7	preta	2	7
11	7, 8, 12	preta	2	7
1	2	preta	3	2
8	4, 11	preta	3	4
12	11	preta	3	11

5) 10



Usamos o algoritmo de Ford-Fulkerson, usando as rotas possíveis e otimizando seu fluxo máximo.

$F_{máx} = 5 + 5 + 10 + 9 = 29$

I $S \underset{9}{\overset{16}{\rightleftarrows}} A \underset{9}{\overset{0}{\rightleftarrows}} B \underset{9}{\overset{1}{\rightleftarrows}} T \quad F = 9$

II $S \underset{5}{\overset{5}{\rightleftarrows}} D \underset{5}{\overset{2}{\rightleftarrows}} E \underset{5}{\overset{0}{\rightleftarrows}} T \quad F = 5$

S – D suportou no máximo 10

III $S \underset{15}{\overset{0}{\rightleftarrows}} D \underset{15}{\overset{0}{\rightleftarrows}} E \underset{15}{\overset{0}{\rightleftarrows}} C \underset{5}{\overset{3}{\rightleftarrows}} T \quad F = 5$

IV $S \underset{25}{\overset{0}{\rightleftarrows}} D \underset{25}{\overset{0}{\rightleftarrows}} E \underset{20}{\overset{0}{\rightleftarrows}} C \underset{10}{\overset{0}{\rightleftarrows}} B \underset{10}{\overset{0}{\rightleftarrows}} T \quad F = 10$