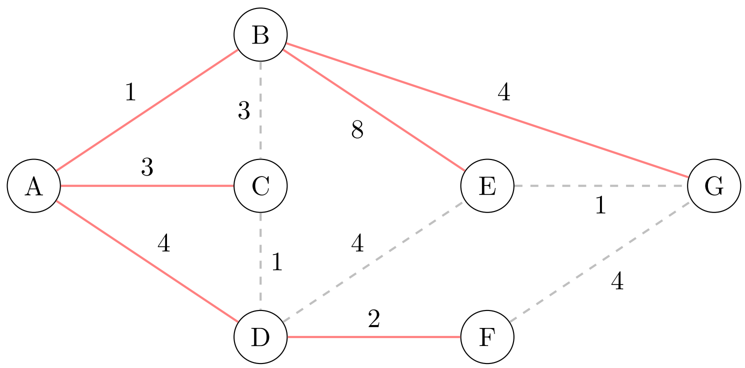
CPS740 - Lista 3

Pedro Maciel Xavier
116023847

1 de setembro de 2020

Questão 1.:

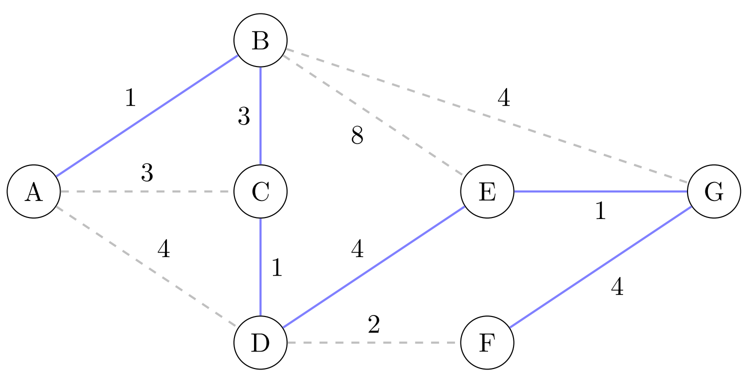
a) Inicialmente, na busca em largura, adicionamos a raiz da árvore geradora à uma fila. Como passo geral do algoritmo, removemos o primeiro elemento da fila e visitamos o vértice correspondente. Ao visitar um vértice marcamos-no e, em seguida, adicionamos à fila todos os seus vizinhos ainda não marcados. Este processo tem fim quando a fila se encontrar vazia.



F	G	E	D	C	B	A

Figura 1: Árvore geradora de uma BFS e a fila utilizada.

b) O procedimento usual para a busca em profundidade é o mesmo, a menos da estrutura de dados utilizada, que neste caso será uma pilha no lugar da fila.



F
G
E
D
C
B
A

Figura 2: Árvore geradora de uma DFS e a pilha utilizada.

Questão 2.:

As especificidades do grafo $G(V, E)$ nos revelam que este se trato, de fato, de uma árvore, uma vez que $|E| = |V| - 1$ e o grafo não possui ciclos.

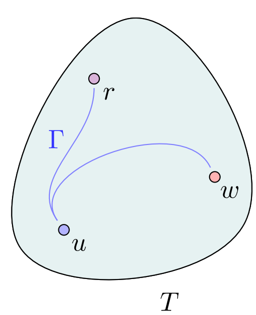
a) Para preencher o vetor de retorno, vamos alterar o algoritmo da BFS,

Algoritmo 1.: BFS dos pais

```
def retorno(G(V, E)):
    // Vetor de Retorno
    seja R[n]

    // vértice qualquer em G
    seja r em G

    // Fila contendo a raiz da árvore geradora
    seja S ← fila({r})

    para cada w em v(G):
        w.cor ← nulo

    enquanto |S| > 0:
        seja w ← S.remover()

        para cada u em viz(G, w):
            se u.cor == nulo:
                continua
            senão:
                u.cor ← 1
                S.inserir(u)
                R[u] ← w

    retorna R
```

b) Calcular o diâmetro de uma árvore T é o mesmo que computar o tamanho do seu maior caminho.

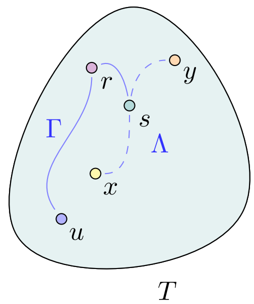
Suposição. *Em uma árvore T, seja u o vértice mais distante da raiz r e w o vértice mais distante de u, temos que o caminho entre u e w é o caminho máximo em T.*



Demonstração. Uma vez que se tem certeza de que u é uma das extremidades do caminho máximo, segue que w se encontra na outra ponta, já que é o vértice mais distante de u, isto é, não existe v tal que $d(u,v) > d(u,w)$.

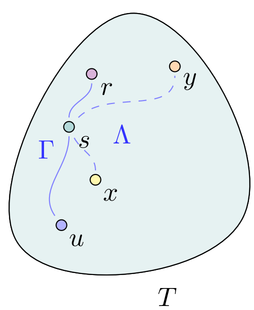
Para demonstrar que u é, de fato, uma das extremidades do caminho de maior comprimento, vamos supor que x e y são os limites do caminho máximo de T. Seja Γ o caminho entre r e u e Λ o caminho entre x e y. Seja s o primeiro vértice em Λ a ser descoberto pela busca iniciada em r.

I.: Se Γ e Λ não possuírem vértices em comum, então certamente r estará no caminho entre s e u.



Assim, temos que $d(s,u) \geq d(r,u)$. Pela busca realizada a partir de r, sabemos que $d(r,u) \geq d(r,x) \geq d(s,x)$. Portanto, $d(s,u) \geq d(s,x)$. Logo, $d(s,u) + d(s,y) \geq d(s,x) + d(s,y) \implies d(y,u) \geq d(y,x)$. Se Λ é o maior caminho e $d(x,y) \geq d(u,y)$, então $d(x,y) = d(u,y)$. Com isto se confirma o fato de que u está em uma das extremidades do caminho máximo para esta configuração.

II.: Se Γ e Λ possuem algum vértice em comum, então s está em Γ.



Argumentando mais uma vez que u é o resultado da busca, reconhecemos que $d(s,u) \geq d(s,x) \implies d(y,s) + d(s,u) \geq d(y,s) + d(s,x)$. Isso nos diz que $d(y,u) \geq d(y,x)$ Contudo, como Λ é o maior caminho da árvore, $d(x,y) \geq d(u,y)$. Logo $d(y,u) = d(y,x)$. De toda forma, u está em uma das pontas do caminho máximo.

Recapitulando o raciocínio apresentado no início da demonstração concluímos que w também será uma extremidade do caminho máximo. ■

Seguindo a construção, basta realizar uma busca em largura (BFS) para encontrar u e uma outra para descobrir w. Uma busca desta natureza visita cada vértice uma única vez, de onde contabilizamos custo $O(2n) = O(n)$ para encontrar os dois extremos.

Por fim, considerando que preenchemos um vetores de retorno ao realizar as buscas e arcando com um custo extra de tempo linear $O(n)$, contamos o número de passos necessários para retornar ao vértice u a partir de w. Este será o diâmetro da árvore e o algoritmo é de tempo linear.

Questão 3.:

Suposição. *Existe um contra-exemplo para a afirmação.*

Demonstração. Para obter um contra-exemplo à afirmação, basta construir uma rede em camadas $D(V, E)$, de origem s e destino t, com n_1 vértices no primeiro nível e n_2 no segundo. Conectamos todos os vértices de níveis adjacentes da seguinte forma: s está ligado aos n_1 vértices do nível 1 por arestas de capacidade a. Cada um destes possui n_2 ligações de enorme capacidade, que denotaremos por ∞, com os vértices do nível 2. Estes, por sua vez, tem cada um uma ligação de capacidade b com o destino t. Por fim, precisamos de $n_1, n_2, a, b \in \mathbb{N}$ que satisfaçam

$$n_1 \cdot a < n_2 \cdot b \ \wedge \ n_1 \cdot (a + 1) > n_2 \cdot (b + 1) \tag{1}$$

A primeira desigualdade advém da rede antes do acréscimo, enquanto a segunda já conta com o incremento.

Seja $(S, \bar{S})$ um corte mínimo de $D(V, E)$ onde $s \in S$ e $v \in \bar{S}$. As arestas entre os níveis 1 e 2 jamais pertencerão a $(S, \bar{S})$, pois suas capacidades são muitíssimo elevadas. Assim, um corte mínimo não pode conter nenhuma destas ligações. Sabemos portanto, que $n_1 \cdot a$ será o corte mínimo pois todas as arestas de peso a estarão inclusas no corte, já que $s \in S$. Assim, $S = a$ e $\bar{S} = V - S$, uma vez que $n_1 \cdot a < n_2 \cdot b$. Temos argumento semelhante para obter o corte mínimo após aumentar as capacidades em uma unidade.

De (1) temos que

$$\begin{aligned} & n_1 \cdot a + n_2 \cdot (b + 1) < n_2 \cdot b + n_1 \cdot (a + 1) \\ & \therefore n_2 < n_1 \end{aligned} \tag{2}$$

além de que

$$\frac{a}{b} < \frac{n_2}{n_1} < \frac{a + 1}{b + 1} \tag{3}$$

de onde concluímos que $a < b$. Fixados $a, b \in \mathbb{N}$, temos de (3) que

$$\frac{a}{b} < \frac{a + 1}{b + 1} \in \mathbb{Q} \tag{4}$$

Os racionais ($\mathbb{Q}$) são um conjunto denso. Isto é, para todo $p \in \mathbb{Q}$ e todo $\epsilon > 0$ existe $q \in \mathbb{Q}$ tal que $|p - q| < \epsilon$. Isto equivale a dizer que para todo par $p, q \in \mathbb{Q}$ com $p < q$, existe $r \in \mathbb{Q}$ tal que $p < r < q$. Basta que seja r a média aritmética entre p e q. Subdividindo sucessivamente o aberto (p, q) com médias, concluímos que existem infinitos números racionais entre p e q. Logo, existe uma infinidade de pares $n_1, n_2 \in \mathbb{N}$ que satisfazem (3). ■

Questão 4.:

a) Somando o fluxo de cada uma das arestas que deixa a origem s temos $3+8+4=15$. Este valor é o mesmo que chega ao destino t, onde $3+11+1=15$.

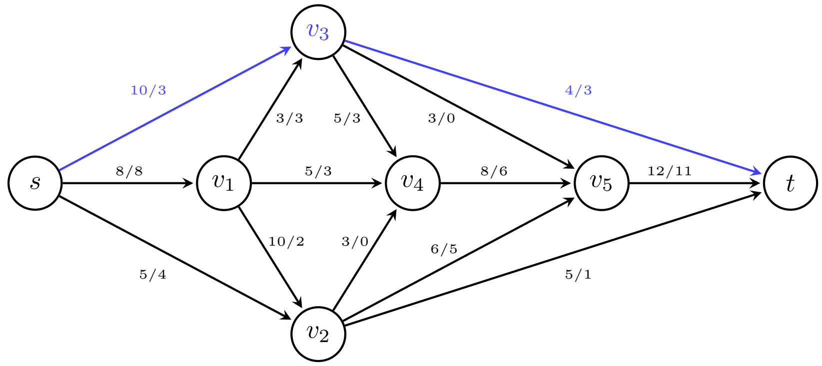


Figura 3: Uma rede de fluxos

O fluxo, no entanto, não é máximo. Basta perceber que temos um caminho aumentante (s, v_3, t) de gargalo 1 que permite aumentar o fluxo para 16.

b) O corte mínimo $(S, \bar{S})$ na rede é dado por $S = \{s, v_1, v_2, v_3, v_4, v_5\}$ e $\bar{S} = \{t\}$. A soma das capacidades de suas arestas e portanto, o fluxo máximo, é $4+12+5=21$.

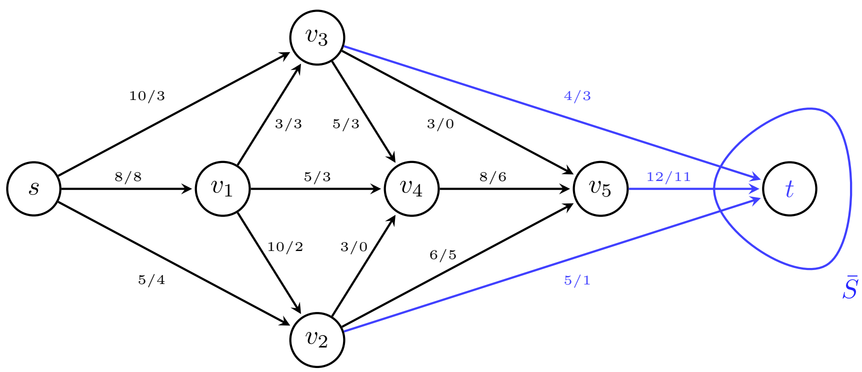


Figura 4: Uma rede de fluxos

Questão 5.:

Para organizar a viagem das F famílias em V veículos podemos modelar o problema como uma rede de fluxos em camadas. Imaginemos a origem s como o ponto de encontro dessas famílias, onde veículos virão buscá-las. Em determinado momento, os guias solicitam que as p_i pessoas da i-ésima família se reúnam em Q_i, o i-ésimo quiosque do local.

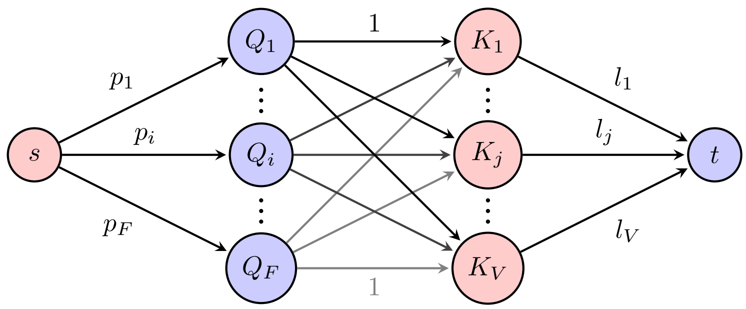


Figura 5: Capacidades de uma rede.

Ao chegarem os V veículos, os turistas foram informados que K_j, a j-ésima Kombi, comporta apenas uma pessoa de cada família, a fim de evitar desentendimentos durante o trajeto destas pessoas que já não aguentam mais ficar juntas após os 14 meses que passaram dentro de casa. Assim, entre cada Q_i e cada K_j existe uma aresta de peso 1.

Após uma viagem tranquila os passageiros chegam à areia de Saquarema, destino conhecido como t. De cada Kombi K_j podem descer, no máximo, l_j passageiros, pois existe uma guarita da Polícia Rodoviária nas redondezas.

Analisando o processo de distribuição das pessoas nos veículos, conforme indica a figura, podemos representar a organização através de uma rede. Com isso, aplicamos algum algoritmo para o fluxo máximo (Ford-Fulkerson, Dinitz, etc.), a fim de obter o valor f_{max}. Só conseguiríamos organizar uma viagem sem problemas se $f_{max} \geq \sum_{i=1}^{F} p_i$.

Referências

[1] Jayme Luiz Szwarcfiter, **Teoria Computacional de Grafos**, 1ª edição, Rio de Janeiro, 2018.

[2] Erik Demaine, Charles E. Leiserson &, Lee Wee Sun, **Introduction to Algorithms**, MIT, 2001.