MC558 – 2020s1

Teste 3

Tiago de Paula Alves

187679

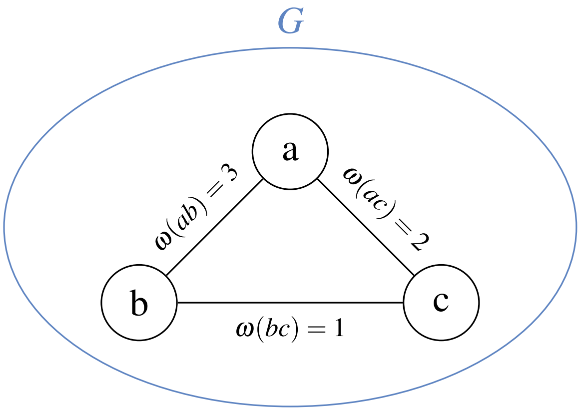
---

1. Prove que o algoritmo TALVEZ-AGM funciona ou mostre um contra-exemplo. Justifique.

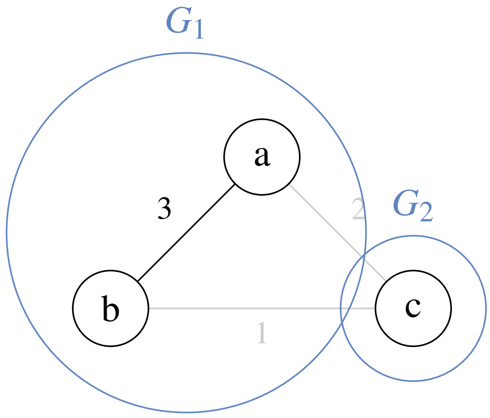
TALVEZ-AGM(G, ω)

se $|V[G]| \leq 2$ **então devolva** G
seja V_1, V_2 qualquer partição balanceada de $V[G]$
seja G_i o subgrafo completo de G contendo apenas os vértices de V_i e
todas as arestas de G ligando vértices de V_i para $i = 1, 2$
seja ω_i a restrição de ω a G_i para $i = 1, 2$
$T_1 \leftarrow$ TALVEZ-AGM(G_1, ω_1)
$T_2 \leftarrow$ TALVEZ-AGM(G_2, ω_2)
seja e uma aresta de peso mínimo no corte (V_1, V_2) em G
devolva $T_1 \cup T_2 \cup \{e\}$

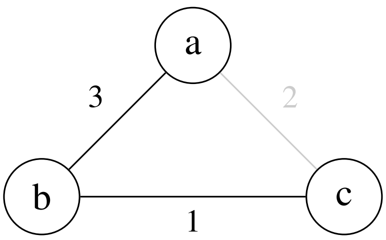
O algoritmo nem sempre é capaz de encontrar uma AG **mínima**. Como os subgrafos G_1 e G_2 são arbitrários, a condição de serem balanceados não é o bastante para que a subestrutura ótima apareça. É possível que G_1 contenha, em um caso extremo, todas as arestas de maior peso do grafo G. Nessa situação, a AGM de G_1 não seria parte da AGM de G, sendo todas suas arestas trocadas (já que G é completo) por arestas do corte (V_1, V_2), em vez de apenas uma aresta de peso mínimo.



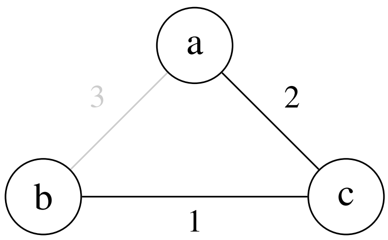
(a) Entrada G com $n = 3$.



(b) Possível partição V_1, V_2 balanceada.



(c) Resultado (não-mínimo) de TALVEZ-AGM, usando a partição proposta.



(d) Árvore Geradora Mínima para G.

Figura 1: Contra-exemplo para o algoritmo TALVEZ-AGM.

Podemos ver na figura 1 como esse caso pode acontecer em um grafo G completo de $n = 3$ vértices (figura 1a). Se na etapa de Divisão (linha 2) a partição for escolhida como $V_1 = \{a, b\}$ e $V_2 = \{c\}$, que é a balanceada, teremos G_1 e G_2 como na figura 1b. No entanto, a AGM $T_1 = G_1$ de G_1 não faz parte da AGM de G, diferentemente do que foi assumido na etapa de Combinação (linha 8). Em uma situação como essa, o resultado de TALVEZ-AGM seria a árvore da figura 1c, que tem pesos maiores que a da figura 1d.

2. Dada um grafo (G,ω), $s,t \in V[G]$ e um valor $k > 0$, mostre como usar Dijkstra para determinar o maior preço entre todos os caminhos de s a t de peso total no máximo k. Explique sua ideia sucintamente e escreva um pseudo-código para seu algoritmo. Você não precisa provar que o algoritmo está correto, mas sua explicação deve ser clara o suficiente para eu me convencer disto. A complexidade do seu algoritmo deve ser $O(V + E + f(V+E))$. Note que você não precisa devolver o caminho, apenas o preço dele.

(a) Nesta questão, todos os grafos são e devem ser representados por listas de adjacências. Além, não têm arestas de peso negativo.

(b) Você tem à disposição um algoritmo chamado Dijkstra que dado um grafo orientado ponderado (G,ω) e um vértice $s \in V[G]$, devolve um vetor $d[\,]$ indexado por $V[G]$ tal que $d[v] = \text{dist}(s,v)$ para todo $v \in V[G]$. Este algoritmo é dado como uma caixa-preta, i.e., você não sabe como ele é implementado internamente.

(c) Suponha que Dijkstra tem complexidade de tempo $O(f(V+E))$ (uma função de V e E). A complexidade exata não é importante, mas deve ser respeitada na solução (veja abaixo).

O algoritmo é baseado em três informações principais:

1. Como o preço de um caminho só depende da aresta de maior peso, não é preciso calcular o preço de todos os caminhos. No caso, a solução será apenas o maior peso $\omega(u,v)$ dentre todas arestas uv que aparecem em um caminho válido (caminhos de s a t com peso total $\leq k$). Isto é:

$$\max_{C \text{ é válido}} \{\text{preço}(C)\} = \max_{C \text{ é válido}} \left\{ \max_{uv \in C} \{\omega(u,v)\} \right\} = \max_{uv \text{ é parte de um cam. válido}} \{\omega(u,v)\}$$

2. Para decidir se uma aresta uv é parte de um caminho válido, basta checar se $\text{dist}(s,u) + \omega(u,v) + \text{dist}(v,t) \leq k$. Note que $\text{dist}(s,u) + \omega(u,v) + \text{dist}(v,t)$ é o peso total do menor caminho de s a t que passa por uv. Se esse caminho não for válido, então nenhum outro que passa por uv será e, portanto, uv não deve ser considerada. Caso contrário, temos um caminho válido com uv.

3. $\text{dist}(s,u)$ é dado diretamente com o resultado de Dijkstra. Para $\text{dist}(v,t)$, no entanto, podemos aplicar o algoritmo novamente, mas no grafo transposto $G^\intercal$, de modo que o resultado é $\text{dist}_{G^\intercal}(t,v) = \text{dist}_G(v,t)$ para todo $v \in V[G] = V[G^\intercal]$.

   Para chegar nesse resultado, precisamos ainda de uma função peso ω^* de $G^\intercal$ que associa para cada aresta transposta $vu \in E[G^\intercal]$ o mesmo peso ω da aresta equivalente no grafo original $uv \in E[G]$.

MAIOR-PREÇO(G,ω,s,t,k)

$d_s[\,] \leftarrow$ DIJKSTRA(G,ω,s)
seja ω^* definida por $\omega^*(u,v) = \omega(v,u)$
$d_t[\,] \leftarrow$ DIJKSTRA$(G^\intercal,\omega^*,t)$

preco $\leftarrow -\infty$
para cada vértice $u \in V[G]$ **faça**
    para cada vértice adjacente $v \in \mathrm{Adj}[u]$ **faça**
        se $d_s[u] + \omega(u,v) + d_t[v] \leq k$
            então *preco* $\leftarrow \max\{preco, \omega(u,v)\}$
devolva *preco*

O pseudo-código assume que k é finito. Se for possível que $k = \infty$, seria necessário checar se $d_s[u] < \infty$ e $d_t[v] < \infty$ para garantir que o caminho $(s,\ldots,u,v,\ldots,t)$ existe.