# MAC0338 - ANÁLISE DE ALGORITMOS

## LISTA 8

FIZ: 9/10

**CHECKLIST**

entregue ☐ fiz ☐ incompleto ☐ não entendi ☐ CRLS ☐

- algoritmo de Dijkstra

1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐

8 ☐ 9 ☐ 10 ☐

1. O diâmetro de um grafo é o máximo das distâncias entre dois vértices. Escreva código que usa o algoritmo de Dijkstra para calcular o diâmetro de um grafo.

O algoritmo de Dijkstra será modificado de modo que a maior estimativa de caminho mais longo seja encontrada, contrariando a proposta original, que é encontrar a menor estimativa.

```
Diametro (G, c, s)
para v ∈ V(G) faça d[v] ← -1        define todas as distâncias como a menor possível
d[s] ← ∞    π[s] ← nil
Q ← V(G)
enquanto Q ≠ ∅ faça
    u ← Extract-Max(Q)        o primeiro é d[s] ← ∞
    para cada v ∈ adj(u) faça
        se v ∈ Q e d[u] + c(uv) > d[v]        maior estimativa
            então π[v] ← u   d[v] ← d[u] + c(uv)
devolva (π, d)
```

2. Considere um digrafo (grafo dirigido) com custos positivos associados aos vértices. O custo de um caminho num tal digrafo é a soma dos custos dos vértices do caminho. Queremos encontrar um caminho de custo mínimo dentre os que começam num vértice $s$ e terminam num vértice $t$. Adapte o algoritmo de Dijkstra para resolver esse problema.

```
Dijkstra (G, s, t)
    para todo v ∈ V(G) faça d[v] ← ∞
    d[s] ← 0     π[s] ← 0
    Q ← V(G)
    caminho_minimo ← ∞
    enquanto Q ≠ ∅ faça
        u ← extract-min (Q)
        se u == t devolva π[t]
        para cada v ∈ adj(u) faça
            se v ∈ Q e d[u] + c(uv) < d[v]
                então π[v] ← u e d[v] ← d[u] + c(uv)
    devolva ∞ (não achou caminho)
```

**ALGORITMO PARA IMPRIMIR O CAMINHO**

```
caminho (π, s, t)
    se s == t então
        imprima "s"
    senão
        caminho (π, s, π[t])
        imprima "t"
```

3. Seja $s$ um vértice de um digrafo $G$ com custos positivos nos arcos. Para cada vértice $v$ de $G$, seja $x[v]$ o custo de *algum* caminho de $s$ a $v$ em $G$. Escreva um algoritmo eficiente que verifique se $x[v]$, para todo $v$, é a distância de $s$ a $v$ em $G$. Explique porque seu algoritmo está correto.

podemos adaptar Dijkstra

```
caminho (G, s, x)
para todo v ∈ V(G) faça
      d[v] ← ∞  e  y[v] ← false      vetor booleano para verificar o vetor x
d[s] ← 0    π[s] ← nil
Q ← V(G)
enquanto Q ≠ ∅ faça
      u ← extract-min (Q)
      para todo v ∈ adj(u) faça
            se v ∈ Q e d[u] + c(uv) < d[v]
                  então d[v] ← d[u] + c(uv)
            se x[v] == d[v]
                  então y[v] ← true
devolva y
```

O algoritmo devolve um vetor y. Se y[v] == true, então x[v] é o custo do caminho de s a t. Se y[v] == false, então x[v] não é o custo do caminho de s a t.

4. Mostre que o algoritmo de Dijkstra pode produzir resultados errados se o digrafo tiver arcos de custo estritamente negativo.

O algoritmo de Dijkstra calcula as estimativas de caminhos mínimos. Ele considera que um vértice está fechado quando a estimativa de caminho mínimo já está obtida, não mudando essa estimativa nas próximas chamadas do algoritmo. Quando há arcos de custo estritamente negativos, o algoritmo produz resultados errados, pois caso um arco negativo seja descoberto e um vértice anterior adjacente já tenha sido fechado, a estimativa de menor caminho fica errada.

Sabrina Araújo da Silva nUSP 12566182

# MAC0338 - ANÁLISE DE ALGORITMOS

## LISTA 8

exercícios 5 e 8

5. Escreva um algoritmo que recebe conjuntos $S$ e $T$ de vértices de um grafo e calcula a distância de $S$ a $T$, ou seja, o custo de um caminho de custo mínimo que começa em algum vértice em $S$ e termina em algum vértice em $T$. O algoritmo deve consumir o mesmo tempo de execução que o algoritmo de Dijkstra. Justifique que seu algoritmo está correto. *Dica:* Basta introduzir uma pequena modificação no algoritmo de Dijkstra.

```
Dijkstra (G, c, S, T)
    para v ∈ V(G) faça d[v] ← ∞
    source ← 0
    para s ∈ S faça
        adiciona arco (source − s) com custo c(source, s) igual a zero
    custo_minimo ← ∞
    Q ← V(G)
    enquanto Q ≠ ∅ faça
        u ← ExtractMin (Q)
        para cada v ∈ adj(u) faça
            se v ∈ Q e d[u] + c(uv) < d[v]
                então d[v] ← d[u] + c(uv)
            se v ∈ T e d[v] < custo_minimo
                então custo_minimo ← d[v]
    devolva custo_minimo
```

O algoritmo insere novos arcos de um novo vértice 0 para todos os vértices de S, esses arcos têm custo 0. A partir desse vértice chamado source, o algoritmo de Dijkstra consegue encontrar qual dos vértices de S forma um caminho mínimo com algum vértice de T. Para cada vértice pertencente a T encontrado, o algoritmo analisa e armazena se o caminho de source para esse vértice possui o menor custo.

6. Escreva um algoritmo que encontre um arco cuja remoção causa o maior aumento na distância de um vértice $s$ a um vértice $t$.

A ideia do algoritmo é inicialmente chamar o algoritmo de Dijkstra com início no vértice s para encontrar a distância até t. Após isso, o algoritmo de Dijkstra é chamado mais E (número de arestas) vezes para cada versão do grafo sem uma determinada aresta. Caso essas E execuções resultem em alguma distância maior que a original, a maior encontrada é retornada.

```
Dijkstra (G, s)
para v ∈ V(G) faça d[v] ← ∞
d[s] ← 0    π[s] ← nil
Q ← V(G)
enquanto Q ≠ ∅ faça
    u ← extract_min (Q)
    para v ∈ adj(u) faça
        se v ∈ Q e d[u] + c(uv) < d[v]
            então d[v] ← d[u] + c(uv)
devolva (d)
```

```
REMOCAO (G, s, t)
d ← Dijkstra (G, s)
original ← d[t]
aumento ← 0    arco_aumento ← nil
para e ∈ E(G) faça
    G' ← G
    G' ← G' sem a aresta e
    d' ← Dijkstra (G', s)
    se d'[t] > aumento então aumento ← d'[t] e arco_aumento ← e
se aumento > original então devolva arco_aumento
senão devolva nil
```

7. Suponha que trocamos a linha 4 do algoritmo do Dijkstra como segue

$$4. \quad \text{while } |Q| > 1$$

Isso faz com que a execução do laço execute $|V|-1$ vezes no lugar de $|V|$ vezes. Será que o algoritmo continua correto?

O algoritmo continua correto. Quando o algoritmo chega no último vértice, sabemos que seu custo atual é pelo menos tão grande quanto o maior dos outros vértices. Como nenhum dos pesos das arestas é negativo, seu valor mais o custo de qualquer aresta que saia dele será pelo menos tão grande quanto os valores d de todos os outros vértices. Isso significa que os relaxamentos que ocorrem não alterarão nenhum dos valores d de nenhum vértice e, portanto, não alterarão seus valores de $\pi$.

8. Dado um digrafo $G = (V, E)$ em que cada aresta $(u, v) \in E$ tem associado um valor $r(u, v)$, que é um número real no intervalo $[0, 1]$ que representa a confiança de um canal de comunicação do vértice $u$ até o vértice $v$. Interpretamos $r(u, v)$ como a probabilidade de que o canal de $u$ a $v$ não falhe, e supomos que tais probabilidades são independentes. Dê um algoritmo eficiente (mesmo tempo de execução que o de Dijkstra) que acha um caminho mais confiável entre dois vértices dados.

```
Dijkstra (G, c, u, v)
para v ∈ V(G) faça d[v] ← ∞
d[u] ← 0        π[u] ← nil
Q ← V(G)
s ← u
enquanto Q ≠ ∅ faça
    s ← ExtractMin (Q)
    para cada t ∈ adj(u) faça
        se t ∈ Q e d[s] + r(st) < d[t]
            então π[t] ← s    d[t] ← d[s] + r(st)
caminho ← caminho ∪ u
devolva caminho_confiavel (π, u, v, caminho)
```

```
caminho_confiavel (π, u, v, caminho)
v = π[v]
se u == v então retorne
caminho-confiavel (π, u, v, caminho)
caminho ← caminho ∪ v
```

O algoritmo inicia Dijkstra com o vértice $u$ e, assim, devolve o vetor $\pi$ com os predecessores que formam um caminho de custo mínimo com os vértices de G. A função caminho confiável devolve uma fila de vértices que formam um caminho de custo mínimo de $u$ até $v$.

9. Seja $G = (V, E)$ um digrafo com pesos $w : E \to \{0, 1, \ldots, W\}$ para algum $W$. Modifique o algoritmo de Dijkstra para que compute os caminhos mínimos a partir de um vértice $s$ em tempo $O(W|V| + |E|)$.

```
Dijkstra (G, s)
para v ∈ V(G) faça d[v] ← ∞
d[s] ← 0        π[s] ← nil
Q ← V(G)
enquanto Q ≠ ∅ faça
    u ← extract_min(Q)
    para v ∈ adj(u) faça
        se v ∈ Q e d[u] + c(u,v) < d[v]
            então d[v] ← d[u] + c(u,v)
devolva (d, π)
```

- o consumo de tempo de Dijkstra depende da implementação da fila de prioridade
- primeiro processamos os vértices mais próximos de s.
- o maior valor possível de um caminho é $(V-1)W$
- assuma que d[ ] representa um bucket de 0 até W, e um bucket no final para valores infinitos.
- percorremos todos os buckets e o primeiro que não estiver vazio tem o seu primeiro item removido, percorremos até esvaziar a fila de V vértices. – $O(WV)$
  - percorremos os vértices adjacentes desse item e relaxamos, percorremos o total de E arestas $O(E)$
- no total temos: $O(WV + E)$

10. Seja $G = (V, E)$ um digrafo com pesos $w : E \to \{0, 1, \ldots, W\}$ para algum $W$. Modifique o algoritmo de Dijkstra para que compute os caminhos mínimos a partir de um vértice $s$ em tempo $O((|V| + |E|) \lg W)$. (*Dica:* Quantas estimativas distintas de caminhos mínimos podem existir em $V - S$ em cada iteração do algoritmo?)

https://stackoverflow.com/questions/44498598/dijkstra-s-algorithm-to-compute-the-shortest-paths-from-a-given-source-vertex-s