* CAMINHOS MÍNIMOS A PARTIR DE UM ÚNICO VÉRTICE * (Cormen, cap. 24)

1. DEFINIÇÃO DO PROBLEMA: dados um grafo direcionado G = (V,E) ponderado por uma função w : E -> R_+ e um vértice de origem o ∈ V, desejamos obter caminhos (de pesos) mínimos de "o" a todos os demais vértices do grafo.

2. Uma implementação do algoritmo de Dijkstra por meio de vetores:

```
============================================================
Algoritmo: dijkstra_vet
Entrada: "G = (V,E)", "w" e "o", como descrito acima.
Saída: vetores "d" e "pai" indexados pelos vértices.
------------------------------------------------------------
01. PARA cada v ∈ V
02. | d[v] := INFINITO , F[v] := falso , pai[v] := nulo
03. d[o] := 0 , F[o] := verd
04. PARA cada aresta (o,v) ∈ E
05. | d[v] := w(o,v) , pai[v] := o
06. REPITA SEMPRE
07. | u := nulo , du := INFINITO
08. | PARA cada v ∈ V
09. | | SE F[v] = falso
10. | | | SE d[v] < du
11. | | | | u := v , du := d[v]
12. | SE u = nulo
13. | | RETORNE (d,pai).
14. | F[u] := verd
15. | PARA cada aresta (u,v) ∈ E
16. | | SE F[v] = falso
17. | | | SE d[u] + w(u,v) < d[v]
18. | | | | d[v] := d[u] + w(u,v) , pai[v] := u
============================================================
```

3. OBSERVAÇÕES SOBRE O ALGORITMO ACIMA:

Observe que o algoritmo acima executa em tempo $O(n^2)$, onde $n = |V|$, pois cada iteração do laço da linha 6 executa em tempo O(n), e no máximo "n" iterações desse laço são executadas -- a saber, para cada vértice u != o, uma iteração para incluir "u" na fronteira "F", e mais uma última iteração para verificar que não há mais vértices a serem incluídos na fronteira.

Observe também que, dependendo do grafo de entrada, nem todos os vértices acabam sendo incluídos na fronteira. Mais precisamente, um vértice "u" é incluído na fronteira se e somente se existe em G um caminho de "o" a "u".

Com relação à correção do algoritmo acima, um dos fatos cruciais é o de que, quando um vértice é incluído na fronteira, a distância até ele já está corretamente calculada. Esse fato é demonstrado no lema a seguir.

4. LEMA: se

   a) "G = (V,E)", "w" e "o" são uma entrada para o problema (definição 1);

   b) o ∈ F ⊆ V;

   c) Para cada v ∈ V \ F, seja P(v) o conjunto dos caminhos de "o" a "v" que, exceto por "v", possuem apenas vértices de "F". Nesse caso,

      d[v] = | mín { w(p) : p ∈ P(v) }, se P(v) não for vazio;
             | INFINITO,   se P(v) for vazio;

   d) u ∈ V \ F e d[u] = mín { d[v] : v ∈ V \ F } < INFINITO,

então d[u] é igual à distância de "o" a "u" em G.

==========
PROVA:

Suponha, por absurdo, que d[u] não é igual à distância de "o" a "u" em G. Nesse caso, ou d[u] é maior que essa distância ou então é menor que ela. Agora, como d[u] < INFINITO, então existe um caminho de "o" a "u" de peso d[u], e portanto a distância de "o" a "u" é menor ou igual a d[u]. Logo, pela suposição inicial, a distância de "o" a "u" em G é menor que d[u]. Nesse caso, existe em G um caminho "p" de "o" a "u" tal que w(p) < d[u]. Seja "v", então, o primeiro vértice de "p" que não pertence a F (tal vértice existe, pois u ∉ F). Além disso, seja "q" o trecho inicial de "p" que vai de "o" a "v". Observe então que, como "q" possui apenas uma parcela das arestas de "p", e como toda aresta de G tem peso não-negativo, então w(q) <= w(p) < d[u]. Além disso, pela forma como "v" foi escolhido, temos que q ∈ P(v). Agora, como, pelo item "c" do enunciado, temos d[v] <= w(q), então d[v] < d[u], o que contradiz a hipótese "d" do enunciado.

Logo, d[u] é igual à distância de "o" a "u" em G, CQD.
==========

5. EXERCÍCIOS:

a) O algoritmo "dijkstra_vet" continua correto se eliminarmos o teste da linha 16? Por quê? É possível eliminar o vetor "F" completamente do algoritmo?

b) O algoritmo "dijkstra_vet" continua correto se eliminarmos o laço da linha 4 e a atribuição "F[o] := verd" da linha 3? Por quê?

c) Escreva a variação do algoritmo de Dijkstra discutida em sala, que utiliza um monte ("heap") binário para descobrir de forma mais rápida o próximo vértice a ser inserido na fronteira. Atente para o fato de que, na linha 18 do algoritmo "dijkstra_vet", o campo d[v] é atualizado; isso significa que, na versão com um monte, as chaves dos elementos do monte podem ser diminuídas (ou, em outras palavras, as prioridades podem ser aumentadas).

   A sua versão modificada deve executar em tempo O(n*lg n + m*lg n), onde n = |V| e m = |E|. O termo "n*lg n" corresponde às no máximo "n" remoções feitas no monte, e o termo "m*lg n" às no máximo "m" atualizações de prioridade (realizadas quando a vizinhança de "u" é percorrida, para todo vértice "u" incluído na fronteira).

d) Você consegue pensar num algoritmo O(n + m) para o caso particular do problema em que se sabe que G não possui circuitos?

e) Mostre que o algoritmo de Dijkstra nem sempre funciona se o grafo possui arestas de peso negativo, mesmo que o grafo não possua circuitos de peso negativo.

   (Um algoritmo O(n*m) para essa versão mais geral do problema é apresentado na seção 24.1 do nosso livro-texto.)