MO417 – Ata Referente ao Problema 24.3-7

Fonte: Livro "Algoritmos, Teoria e Prática - Tradução da 2ª edição americana".

Enunciado:
Modifique seu algoritmo do Exercício 24.3-6 para ser executado no tempo O((V+E) lg W). (Sugestão: Quantos valores de caminhos mais curtos distintos podem existir em V-S em qualquer instante?)

Solução:
A solução deste exercício depende do algoritmo descrito no exercício 24.3-6, cuja explicação é encontrada em sua respectiva ata, dado abaixo:

```
DIJKSTRA (G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 BUILD-VECTOR(Q, G)
4 while !IS-EMPTY(Q)
5       u ← EXTRACT-MINIMAL(Q)
6       S ← S ∪ {u}
7       for cada vértice v ∈ Adj[u] do
8             RELAX(u, v, w, Q)

INICIALIZE-SINGLE-SOURCE (G, s)
1 for cada vértice v ∈ V[G] do
2       d[v] ← ∞
3       π[v] ← NIL
4       d[s] ← 0

BUILD-VECTOR (Q, G)
1 global_position ← 0
2 length[Q] ← |V| * W
3 for cada vertice v ∈ V[G] do
4       if d[v] = ∞ then
5             d[v] ← length[Q]
6       INSERT-LIST(Q, v)

IS-EMPTY (Q)
1 while global_position <= length[Q] and Q[global_position] = NIL
2       global_position ← global_position + 1
3 return (global_position > length[Q])

EXTRACT-MINIMAL (Q)
1 while global_position <= length[Q] and Q[global_position] = NIL
2       global_position ← global_position + 1
3 if global_position <= length[Q] then
4       u ← primeiro vertice de Q[global_position]
5       REMOVE-LIST(Q, u)
6       return u
7 return NIL

RELAX (u, v, w, Q)
1 if d[v] > d[u] + w(u, v) then
2       DECREASE-KEY(d[u] + w(u, v), Q, v)
```

```
3       π[v] ← u

DECREASE-KEY (key, Q, v)
1 REMOVE-LIST(Q, v)
2 d[v] ← key
3 INSERT-LIST(Q, v)
```

No algoritmo original, o procedimento DIJKSTRA tem complexidade assintótica O(WV + E), pois:

- Na linha 1, INITIALIZE_SINGLE_SOURCE trata da inicialização de cada vértice, assim como seu predecessor. Desta forma, este procedimento tem complexidade O(V). A linha 2 inicializa o conjunto S como vazio. Esta operação é feita em tempo de execução constante, O(1).

- Na linha 3, BUIILD_VECTOR tem complexidade O(V), pois insere cada vértice no topo de uma lista duplamente encadeada.

- Nas linhas 4 e 5, IS_EMPTY e EXTRACT_MINIMAL tem complexidade O(VW), já que os procedimentos fazem uma busca linear no vetor, cujo tamanho é VW. Linha 6 é executada em tempo constante.

- Linha 7 é executada uma vez para cada aresta na lista de adjacências de um vértice. O procedimento RELAX é executado em tempo constante, O(1), na linha 8. Assim, a linha 7 tem complexidade O(V + E).

Concluindo, a complexidade total de DIJKSTRA neste caso é O(VW + E). Dado que o maior custo desta complexidade é atribuída à busca linear feita ao vetor Q de tamanho VW, que representa a estimativa da distância de cada vértice, *d[v]*, temos que implementar uma estrutura de dados na qual a busca por uma *d[v]* seja feita em lg W, no máximo.

Uma das formas de alcançarmos tal resultado é trocar o vetor Q por um *heap* binário mínimo, pois cada operação de busca custa O(lg W). Como temos a garantia de que a distância estimada é inteira e no máximo W, teremos no pior caso W elementos no *heap* em qualquer instante.

Cada nó do *heap* binário armazena um valor de distância estimada, que é a chave daquele nó. Dados satélites, como os vértices que possuem uma determinada *d[v],* são administrados por uma lista duplamente encadeada, como aquela apresentada no algoritmo do exercício 24.3-6, ligada ao nó do *heap*. Isto significa que todos os vértices que possuam a mesma distância estimada estão armazenados em uma lista duplamente encadeada ligada ao nó do *heap* que corresponde ao valor *d[v]* desses vértices.

Claramente, os procedimentos que fazem referência ao vetor Q no algoritmo original precisarão de alterações para manipular o *heap* binário que o substituirá. Por exemplo, o procedimento EXTRACT_MINIMAL precisará ser alterado para que o elemento do *heap* possa ser removido apenas se não há mais qualquer vértice na lista duplamente encadeada com a distância estimada correspondente a aquele primeiro.

Dado que a busca em um *heap* é O(W) e operações de inserção, remoção e busca na lista duplamente encadeada custam, respectivamente O(1), O(1) e O(V+E), a complexidade total deste novo algoritmo é O((V+E) lg W).

Redatora: Gabriela Batista Leão.