Ata da resolução do problema 24.3-6

Washington Luís Pereira Barbosa
RA: 971766
16 de junho de 2009

Enunciado do problema

Seja $G = (V, E)$ um grafo orientado ponderado com função peso $w: E \rightarrow \{0, 1, ..., W\}$ para algum inteiro não negativo W. Modifique o algoritmo de Dijkstra para calcular os caminhos mais curtos a partir de um vértice de origem s dado no tempo $O(WV + E)$.

Algoritmo modificado

```
W é uma constante global

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)
```

Notas sobre as modificações realizadas no algoritmo

Sempre andamos em uma direção no vetor *Q*, mesmo após um relaxamento. É porque neste caso um valor de estimativa poderia diminuir de valor, mas quando ele diminui, ele sempre será diminuído pelo valor do vértice atual *u* que acabei de remover do vetor *Q* mais o peso dele até o vértice sendo relaxado (*v*, na lista de adjacências), logo, na pior das hipóteses, o que vai ocorrer é o vértice ser relaxado de $d[v] <= d[u] + w(u, v)$ e, portanto, estará armazenado no vetor em uma posição posterior a posição atual que estou visitando.

INSERT-LIST(Q, v): Insere um vértice v no topo da lista duplamente ligada *Q[d[v]]*, e ajusta o ponteiro do vértice para a posição na fila, tempo $\theta(1)$. Ajusta para *NIL* quando não há um valor de estimativa na posição.

REMOVE-LIST(Q, v): remove o vértice *v* da lista *Q*. Vale lembrar que para cada vértice representado na lista de adjacência possui um ponteiro para a estrutura que o representa no vetor *Q*, como discutido em aula e ajustado no *INSERT-LIST*. A busca pelo nó leva tempo $\theta(1)$, remoção leva tempo $\theta(1)$ também (lista duplamente ligada).

Resolução apresentada por Jônatas Costa