MC558 - Projeto e análise de algoritmos II - Unicamp

Laboratório 4 - Corrida Maluca

Erik Perillo, RA135582

7 de novembro de 2016

1 Abordagem

O problema da Corrida Maluca pode ser visto como o de encontrar um caminho de s a t em um grafo orientado que tenha o peso não maior que w_{limit} e com uma aresta de maior peso possível. A ideia é checar todas as arestas do grafo procurando pela maior delas que satisfaça a restrição de peso de um caminho de s a t. O menor caminho possível incluindo uma aresta (u, v) é o menor caminho de s a u e o menor caminho de v a t ligados pela aresta (u, v). Se algum destes caminhos não existir, definimos que o seu peso é ∞. O pseudo algoritmo abaixo explica de forma abstraida a ideia.

Algorithm 1

```
1: procedure CRAZYRACE(G, s, t, w_limit)
2:     w_max ← −1
3:     G^T ← transpose(G)
4:     d_s ← dijkstra(G, s)
5:     d_t ← dijkstra(G^T, t)
6:     for (u, v) ∈ G do
7:         w ← (u, v).weight
8:         if w > w_max and d_s[u] + w + d_t[v] ≤ w_limit then
9:             w_max ← w
       return w_max
```

2 Complexidade

O grafo foi implementado como uma lista de adjacências. Seja V o número de vértices do grafo e E o número de arestas do mesmo.

A linha 3 do algoritmo consome tempo $O(V + E)$ pois basta cópia dos vértices de G com uma passagem pelas arestas. O algoritmo de *Dijkstra* [1] nas linhas 4 e 5 consome tempo $O((V + E)lg(V))$ pois o mesmo foi implementado com uma *min-heap*. O *loop* da linha 6 acontece em $O(V + E)$. Assim, temos como tempo total $O(V + E + (V + E)lg(V)) = O((V + E)lg(V))$.

Referências

[1] Cormen et al. *Introduction to Algorithms, 3th ed.* The MIT Press, 2009, pp. 658–662.