Documentação Projeto e Análise de Algorítimos
Transporte Eficiente

Ronald Davi Rodrigues Pereira
Matrícula: 2019721249
ronald.pereira@dcc.ufmg.br
Universidade Federal de Minas Gerais

1 Modelagem

O problema foi modelado como uma busca por caminho mínimo em um grafo utilizando-se do algoritmo de Dijkstra. Para a execução de tal algoritmo nesse problema, foi necessário modelá-lo como um problema de grafo. Para tal, foi utilizada uma representação em forma de matriz (vector<vector<int») de cada estado do container sendo um vértice do grafo e cada aresta saindo desse vértice como o custo de se realizar a troca entre duas caixas adjacentes, ligando-o a um vértice que representa o próximo estado do container com essa mudança aplicada. Desse modo, foi possível a modelagem de um problema de busca de caminho mínimo em grafos, pois temos as configurações inicial (vértice s) e final (vértice t) dos containers dado como entrada de um caso de teste. No entanto, montar toda a estrutura do grafo logo na inicialização da execução se mostrou demasiadamente ineficiente, pois muitos dos vértices nem mesmo eram visitados. Portanto, por motivos de eficiência, o grafo é montado dinamicamente enquanto o algoritmo de caminho mínimo caminha no grafo. Desse modo, ao se encontrar um caminho mínimo do vértice s para t, temos o custo mínimo global do caminho que liga a configuração inicial à final, resolvendo o problema com sucesso.

2 Análise de Complexidade

O programa, em sua maioria, é constituído de funções de inicialização e pré-processamento dos dados de entrada. Para tais, serão ignoradas suas análises de complexidade mais aprofundadas, pois somente são executados para os vértices inicial e final do caso de teste, obtendo uma complexidade constante ($O(1)$). No entanto, as demais funções desempenham funções cruciais do algoritmo e serão analisadas abaixo:

- FlatConfig - Para um determinado container, recebe-se a configuração dele em forma matricial e o transforma para uma forma vetorizada, servindo como um identificador único daquela configuração. Para cada vértice, ele roda em $O(H * W)$, sendo H a altura do container e W a largura. No entanto, esses valores são constantes e, de acordo com a especificação do problema, limitados por $H * W \leq 9$. Portanto, a complexidade final dessa função é $O(V)$.
- GenerateNextMoves - Para um determinado vértice contendo uma configuração, são realizadas a geração somente das alterações de ordem 1 (somente 1 troca entre duas caixas adjacentes), por meio da função GenerateConfigs, que move uma caixa para a esquerda ou para cima, caso seja possível. Para cada vértice, essa função executa em $O(deg(V))$, que se traduz nas ligações de uma configuração à outras configurações possíveis de serem alcançadas com uma troca. Portanto, a complexidade final dessa função é $O(V * deg(V))$, que no pior caso possível (grafo totalmente conectado) se traduz em $O(V * (V - 1)) = O(V^2)$.
- Dijkstra - Para um determinado vértice inicial s, por meio de um min-heap (heap binário aonde sempre se extrai o menor elemento em $O(1)$ e se insere novos elementos em $O(\log n)$), busca o caminho mínimo até um vértice final t. Complexidade: $O(E \log V)$.