

Relatório Trabalho II.

ALEST II

Rodrigo Pacheco*, Guilherme Langaro[†]
Faculdade de Informática — PUCRS

19 de junho de 2025

Resumo

Este artigo descreve alternativas de solução para o problema do segundo trabalho proposto na disciplina de Algoritmos e Estrutura de Dados II no semestre 1/2025, que trata da resolução do problema "Os fenícios estão chegando".

Serão apresentadas duas possibilidades de solução para o problema.

Introdução

O problema a ser resolvido é, com base na quantidade de linhas e colunas de um mapa, e o próprio, montar um algoritmo para calcular o menor caminho, do Porto 1 até o Porto 9, passando, se possível, pelos portos 2 - 8, e evitando os nodos '*', que representam trechos não-navegáveis.

O enunciado também informa que só é possível se mover nas direções Norte, Sul, Leste e Oeste e explica como os mapas são descritos:

Os numeros iniciais nos fornecem o tamanho do mapa, com o primeiro sendo o numero de linhas e o segundo sendo o numero de colunas, assim criando uma matriz de pontos, asteriscos e números

Modelagem do problema

De início retiramos do arquivo ".map" os números que nos dariam o tamanho esperado do mapa, em seguida, utilizando estes valores, criamos uma matriz (dinamicamente alocada) para salvar o conteúdo do mapa, a fim de processar melhor os dados, atribuir tipos a cada nodo e fazer a ligação entre os nodos, utilizando como critério, que um nodo 'c' (caminho) só pode se ligar a um 'c' ou 'p' (porto) e vice-versa.

Primeira solução

Depois de considerar o problema, tivemos a ideia de utilizar o algoritmo de Dijkstra, que vai de vertice em vertice utilizando um algoritmo de BFS, que visita todos os vizinhos ligados a um vertice e salva o valor do caminho que ele já fez até aquele ponto, mas que também sobrescreve o valor caso aquele vertice já tenha sido visitado e que o caminho feito pelo vertice atual seja menor do que

*rodrigo.pacheco004@edu.pucrs.br

[†]guilherme.langaro@edu.pucrs.br

o salvo no vetor.

Mas quando estavamos implementando o algoritmo percebemos que no problema atual, utilizar Dijkstra ou BFS nos devolveria o mesmo valor, e como o problema não envolve a determinação de um caminho mais curto e só pede a distancia menor entre os portos, decidimos simplesmente utilizar uma BFS para determinar a distancia entre os portos, e consequentemente a quantidade de combustível necessaria.

Segunda solução

Decidimos no final utilizar um algoritmo de BFS para resolver nosso problema. O algoritmo BFS(breadth-first search) utiliza uma fila para visitar cada nodo, retirando da fila o nodo inicial e a partir dele adicionando todos os nodos vizinhos que ainda não tenham sido visitados, ajustando um vetor que salva distancia entre o vetor atual e o inicial.

No nosso caso, cada vizinho é um nodo em uma direção cardinal, norte, sul, leste, oeste, que não seja "terra", um nodo do tipo "i".

Este algoritmo, em pseudo-codigo, funciona desta forma:

Algoritmo:

```
1      Alocamos um vetor de inteiros "distTo" do tamanho lin * col , inicializado com zeros
2
3      Inicializamos a variável "totalCombustivel" em 0
4      Inicializamos a variável "portoAtual" com o número do porto inicial
5
6      Enquanto verdadeiro faça
7
8          Para cada posição i de 0 até lin * col:
9              javisitado[i] = falso
10             distTo[i] = 0
11
12         Esvaziamos a fila
13         Inserimos o nodo "inicio" na fila
14         javisitado[inicio → num] = verdadeiro
15
16         Defnimos uma variável booleana "achouProximoPorto" = falso
17         Calculamos "proximoPorto" como (se portoAtual == 9 então 1 senão portoAtual + 1)
18
19         Enquanto a fila não estiver vazia:
20
21             Removemos o nodo "atual" da frente da fila
22
23             Se atual → porto == proximoPorto então
24                 totalCombustivel += distTo[atual → num]
25                 Imprimimos qual porto foi encontrado e o combustível gasto até agora
26
27                 inicio = atual
28                 portoAtual = proximoPorto
29                 achouProximoPorto = verdadeiro
```

```

30      Saímos do laço interno
31
32      Para cada direção (norte , sul , leste , oeste):
33          Se existe vizinho nessa direção E ele não foi visitado E tipo diferente de 'i'
34              Marcamos vizinho como visitado
35              Inserimos vizinho na fila
36               $distTo[vizinho \rightarrow num] = distTo[actual \rightarrow num] + 1$ 
37
38      Se achouProximoPorto == falso então
39          Imprimimos que o porto portoAtual+1 não é acessível
40          portoAtual += 1
41          (O início permanece igual)
42
43      Se portoAtual == 1 E achouProximoPorto == verdadeiro então
44          Encerramos o laço principal
45
46      Imprimimos o total de combustível necessário
47
48      Liberamos a memória do vetor distTo

```

Explicação

Ao chamar a função BFS, o algoritmo tenta encontrar o caminho mínimo entre portos consecutivos (de 1 até 9) em um mapa representado por nodos interligados. Para isso, ele utiliza a estrutura de busca em largura (BFS), reinicializando a cada iteração o vetor de visitados e as distâncias a partir do nodo atual.

Em cada busca, a fila é esvaziada e o nodo de início é inserido, marcando-o como visitado. O algoritmo então percorre a fila, expandindo para os vizinhos (norte, sul, leste e oeste) que ainda não foram visitados e não são inválidos. Se durante a busca o próximo porto for encontrado, a distância até ele é somada ao total de combustível gasto, e a busca é reiniciada a partir desse novo porto.

Caso um porto seguinte não seja acessível, o algoritmo apenas avança para o próximo número de porto, sem atualizar o ponto de partida. Ao final, é exibida a quantidade total de combustível necessária para visitar todos os portos possíveis de 1 até 9 e devolta para 1 no final. O vetor de distâncias é liberado da memória ao término do processo.

Resultados

Depois de implementar o algoritmo acima em linguagem C e executá-lo, obtivemos os seguintes resultados:

Caso	Resultado	Tempo de execução(s)
0	248	0,003s
1	598	0,010s
2	1112	0,032s
3	2210	0,217s
4	3519	0,900
5	11518	3,510s

Outputs do terminal:

linhas = 27 colunas = 50 Porto 2 encontrado com combustivel 27 Total de combustivel ate agora: 27 Porto 3 encontrado com combustivel 39 Total de combustivel ate agora: 66 Porto 4 encontrado com combustivel 12 Total de combustivel ate agora: 78 Porto 5 encontrado com combustivel 11 Total de combustivel ate agora: 89 Porto 6 encontrado com combustivel 42 Total de combustivel ate agora: 131 Porto 7 nao e acessivel Porto 8 encontrado com combustivel 48 Total de combustivel ate agora: 179 Porto 9 encontrado com combustivel 42 Total de combustivel ate agora: 221 Porto 1 encontrado com combustivel 27 Total de combustivel ate agora: 248 Total de combustivel necessario: 248 Tempo de execucao: 0.003 segundos	linhas = 55 colunas = 100 Porto 2 encontrado com combustivel 61 Total de combustivel ate agora: 61 Porto 3 encontrado com combustivel 97 Total de combustivel ate agora: 158 Porto 4 encontrado com combustivel 84 Total de combustivel ate agora: 242 Porto 5 encontrado com combustivel 72 Total de combustivel ate agora: 314 Porto 6 nao e acessivel Porto 7 encontrado com combustivel 51 Total de combustivel ate agora: 365 Porto 8 encontrado com combustivel 33 Total de combustivel ate agora: 398 Porto 9 encontrado com combustivel 63 Total de combustivel ate agora: 461 Porto 1 encontrado com combustivel 137 Total de combustivel ate agora: 598 Total de combustivel necessario: 598 Tempo de execucao: 0.010 segundos	linhas = 111 colunas = 200 Porto 2 encontrado com combustivel 102 Total de combustivel ate agora: 102 Porto 3 encontrado com combustivel 96 Total de combustivel ate agora: 198 Porto 4 encontrado com combustivel 163 Total de combustivel ate agora: 361 Porto 5 encontrado com combustivel 113 Total de combustivel ate agora: 474 Porto 6 encontrado com combustivel 135 Total de combustivel ate agora: 609 Porto 7 encontrado com combustivel 120 Total de combustivel ate agora: 729 Porto 8 encontrado com combustivel 114 Total de combustivel ate agora: 843 Porto 9 encontrado com combustivel 62 Total de combustivel ate agora: 905 Porto 1 encontrado com combustivel 207 Total de combustivel ate agora: 1112 Total de combustivel necessario: 1112 Tempo de execucao: 0.032 segundos
linhas = 277 colunas = 500 Porto 2 encontrado com combustivel 92 Total de combustivel ate agora: 92 Porto 3 encontrado com combustivel 402 Total de combustivel ate agora: 494 Porto 4 encontrado com combustivel 79 Total de combustivel ate agora: 573 Porto 5 encontrado com combustivel 335 Total de combustivel ate agora: 908 Porto 6 nao e acessivel Porto 7 encontrado com combustivel 283 Total de combustivel ate agora: 1191 Porto 8 encontrado com combustivel 185 Total de combustivel ate agora: 1376 Porto 9 encontrado com combustivel 400 Total de combustivel ate agora: 1776 Porto 1 encontrado com combustivel 434 Total de combustivel ate agora: 2210 Total de combustivel necessario: 2210 Tempo de execucao: 0.217 segundos	linhas = 555 colunas = 1000 Porto 2 encontrado com combustivel 649 Total de combustivel ate agora: 649 Porto 3 encontrado com combustivel 574 Total de combustivel ate agora: 1223 Porto 4 nao e acessivel Porto 5 nao e acessivel Porto 6 encontrado com combustivel 222 Total de combustivel ate agora: 1445 Porto 7 encontrado com combustivel 340 Total de combustivel ate agora: 1785 Porto 8 encontrado com combustivel 714 Total de combustivel ate agora: 2499 Porto 9 nao e acessivel Porto 1 encontrado com combustivel 1011 Total de combustivel ate agora: 3510 Total de combustivel necessario: 3510 Tempo de execucao: 0.900 segundos	linhas = 1111 colunas = 2000 Porto 2 encontrado com combustivel 1252 Total de combustivel ate agora: 1252 Porto 3 encontrado com combustivel 2355 Total de combustivel ate agora: 3607 Porto 4 encontrado com combustivel 488 Total de combustivel ate agora: 4095 Porto 5 encontrado com combustivel 1425 Total de combustivel ate agora: 5520 Porto 6 encontrado com combustivel 1535 Total de combustivel ate agora: 7055 Porto 7 encontrado com combustivel 949 Total de combustivel ate agora: 8004 Porto 8 encontrado com combustivel 918 Total de combustivel ate agora: 8922 Porto 9 encontrado com combustivel 1154 Total de combustivel ate agora: 10076 Porto 1 encontrado com combustivel 1442 Total de combustivel ate agora: 11518 Total de combustivel necessario: 11518 Tempo de execucao: 3.519 segundos

Figura 1: Casos 0 - 5

Complexidade do código apresentado: $O(k + n)$

(n) = número de linhas do mapa

(k) = número de colunas do mapa

Conclusões

O problema foi resolvido de forma satisfatória ao utilizar um algoritmo de BFS para determinar a quantidade de combustivel necessario para chegar de porto em porto e calcular a quantidade minima de combustivel para passar por todos os portos e voltar para o inicio.

Salvamos o nosso progresso durante este trabalho em um repositório do github:

<https://github.com/TheThird4993/ALEST-II-Trabalho-2>