Siga o Dinheiro

Gabriel D. Rebello*
Faculdade Politécnica — PUCRS

16 de abril de 2024

Resumo

Este artigo mostra o passo a passo para a resolução do primeiro problema proposto na disciplina de Algoritmos e Estruturas de Dados II no terceiro semestre do curso de Engenharia de Computação, que trata de percorrer um mapa advindo de um arquivo de texto, contabilizando a quantidade de dinheiro que ficou para trás em uma fuga do banco.

Para isso, foi desenvolvida uma proposta de resolução trabalhando com matrizes, bem como apresentados seus resultados obtidos a partir de diversos casos de teste previamente fornecidos, assim como a análise de sua eficiência.

Introdução

Dentro do escopo da disciplina de Algoritmos e Estruturas de Dados II, o problema a ser solucionado tem a seguinte descrição: ao realizar com sucesso um assalto a banco, os bandidos saíram pela cidade deixando uma trilha de notas para trás e, enquanto a polícia os seguia para efetuar a captura, aproveitaram para recolher o dinheiro que caiu, desenhando um mapa detalhado com a rota seguida e os locais que certas quantias foram recuperadas, que será usado para uma análise, com base nas seguintes informações:

1. Em cada mapa existe a trilha deixada pelos bandidos;
2. A primeira informação do mapa é o tamanho dele, em linhas e colunas;
3. A trilha inicia em algum ponto do lado esquerdo, com o carro dos bandidos andando para a direita;
4. O carro sempre anda em linha reta a não ser que encontre os símbolos / ou \, que são os sinais para fazer uma curva;
5. Os bandidos foram capturados no local marcado com um #;
6. O dinheiro deverá ser somado à quantia final na ordem em que foi encontrado.

Por exemplo, seguindo um caminho simples iniciado na esquerda apresentado a seguir

---15--2--3--#

O resultado da soma dos valores obtidos seria 20, porém, caso o caminho começasse pela direita e continuasse até a esquerda, o resultado final seria 56.

* g.rebello@edu.pucrs.br, turma 11

Para resolver o problema proposto, analisaremos uma possível alternativa de solução, bem como suas particularidades, demonstrando sua eficiência e abordagem central, com uma metodologia relativamente simples e intuitiva. Juntamente a isso, serão apresentados os resultados finais obtidos com base nos casos de teste previamente fornecidos, bem como o tempo de execução em cada um dos casos e a complexidade do algoritmo.

Solução

Depois de considerar o problema, foi elaborado um código capaz de ler um arquivo de texto recebido que conterá o mapa, bem como transportar este mapa para uma matriz de caracteres, que será lida dentro do programa e interpretada, para seguir corretamente o caminho feito pelos bandidos e coletar corretamente o dinheiro. Para tal abordagem, o código foi feito de maneira modular, usando uma função capaz de ler o arquivo e transportá-lo para uma matriz, bem como quatro funções sendo usadas para percorrer a matriz, duas para casos horizontais (esquerda e direita), bem como duas outras para casos verticais (cima e baixo), lidando corretamente com as variáveis que manipulam o caminho a ser seguido, e coletando o dinheiro deixado no chão. Sendo assim, o código será demonstrado em partes, mostrando primeiramente suas funções e, após isso, a parte principal. Começaremos com a função de ler o arquivo:

```
funcao LerArquivo():
   arquivo = recebe(“arquivo.txt”)
   se arquivo.abriu() é FALSE então
       exibe “Erro ao ler arquivo”
       sai do programa
   fim
   int linhas, colunas
   linhas = recebe()
   colunas = recebe()
   vetor[string] mapa
   string linha
   enquanto arquivo tem linha
       mapa.recebe(linha)
   arquivo.fechou()
   retorna mapa
fim
```

A função “LerArquivo” é bem simples, criando uma variável que receberá o endereço do arquivo e colocando os dados que esse arquivo contém em um mapa, fazendo verificações básicas para garantir que o programa terá feito a abertura e o fechamento do arquivo de maneira correta, a fim de não comprometer o andamento do programa. A seguir temos a função que percorre o mapa:

```
funcao segue(mapa, altura, direcao, i, j, valor, valorFinal):
   enquanto mapa[atual] não é ‘#’ nem ‘/’ nem ‘\’
       se mapa[atual] é número então
           valor = valor + mapa[atual][atual]
       se não então
```

```
            valorFinal = valorFinal + valor
            anula valor
        fim
    se não está na borda nem na dobra nem no fim então
        valorFinal = valorFinal + valor
        anula valor
    fim
    se mapa[atual][atual] é ‘/’ então
        se direcao é direita então
            direcao é cima
        se não se direcao é esquerda então
            direcao é baixo
        se não se direcao é cima então
            direcao é direita
        se não
            direcao é esquerda
    fim
    se não se mapa[atual][atual] é ‘\’ então
        se direcao é direita então
            direcao é baixo
        se não se direcao é esquerda então
            direcao é cima
        se não se direcao é cima então
            direcao é esquerda
        se não
            direcao é direita
    fim
fim
```

O princípio da função é simples, enquanto o valor atual do mapa percorrido não for #, indicando o final, nem \ ou /, indicando que deve mudar de direção, o programa seguirá indo para a direção atual e somando ao valor final o dinheiro contabilizado. No programa implementado em linguagem C++, foi feito o uso de quatro funções desse tipo, uma para cada direção, porém, pensando em simplicidade e buscando deixar o texto menos redundante, foi apresentado uma função geral que compila a ideia central por trás das funções que percorrem o código, se atentando à estrutura da mesma sem estar apegado aos detalhes de implementação.

Ao final dessa função, existe uma área responsável por mudar a direção que a matriz está sendo percorrida, e para tal, foram usados duas flags de direção, sendo números inteiros de 1 a -1, onde a flag horizontal indica 1 para direita, -1, para esquerda e 0 para sem ir para nenhum dos lados, bem como a flag vertical que indica 1 para baixo, -1 para cima e 0 para quando não sobe nem desce, assim podendo fornecer uma organização simples para as direções tomadas no programa.

```
funcao main():
    int valorFinal = 0, inicio = 0, altura = 0, direcao = 1
    enquanto mapa[inicio][0] não for ‘-‘
```

```
        inicio = inicio + 1
    int i = inicio, j = 0
    enquanto mapa[i][j] não for ‘#’
        se direita então
            segueDireita(mapa, altura, direcao, i, j, valor,
            valorFinal)
        se não se esquerda então
            segueEsquerda(mapa, altura, direcao, i, j, valor,
            valorFinal)
        se não se baixo então
              segueBaixo(mapa, altura, direcao, i, j, valor, valorFinal)
        se não
              segueCima(mapa, altura, direcao, i, j, valor, valorFinal)
        fim
     fim
fim
```

A função main apenas instancia os primeiros valores necessários para serem usados no programa, descobre onde deverá começar a percorrer a matriz e então entra em um laço que chamará as funções responsáveis por percorrer a matriz, a fim de contabilizar os valores recebidos.

Resultados

Depois de implementar o algoritmo acima em linguagem C++ e executar o mesmo nos casos de teste fornecidos, executando-os dez vezes e fazendo uma média de tempo de execução, chegamos aos seguintes resultados:

Caso de teste	Valor final (R$)	Tempo de execução (ms)
casoG50	14111	0.191
casoG100	25411	0.736
casoG200	73424	2.838
casoG500	871897	20.739
casoG750	20481572	50.448
casoG1000	13680144	85.798
casoG1500	18727169	206.633
casoG2000	20748732	361.896

Figura 1 - Casos de teste

Após a obtenção destes valores, foi montado um gráfico com a relação dos casos de teste e o tempo de execução dos mesmos, para conseguirmos fazer uma melhor análise da relação do tamanho da matriz com o tempo de execução dos testes:

Figura 2 - Complexidade do algoritmo

Visualizando o gráfico em que o eixo y é composto pelo tempo de execução e o eixo x composto pelo valor das linhas e colunas da matriz, fica nítido perceber que o algoritmo implementado é de complexidade $O(n^2)$, sustentado pelo fato de o programa percorrer uma matriz usando dois laços aninhados.

Conclusões

A abordagem escolhida para resolver o problema da leitura de um mapa foi o uso da matriz, o que facilitou na estruturação do código, por se tratar de uma estrutura simples de compreender e atuar, resultando em um algoritmo fácil de ser entendido e ainda assim eficiente para o caso proposto, visto que o problema possivelmente não possa ser resolvido em uma complexidade abaixo de $O(n^2)$, portanto acreditamos que a solução pensada resolve satisfatoriamente o problema, chegando nos resultados corretos e tomando cuidado com todas as possibilidades de erro, como encerrar o programa sem ter armazenado todos os valores no valor final, ou perder valores na troca de direção.

Sendo assim, foi modelada apenas uma solução para o problema, que foi capaz de chegar em resultados satisfatórios com uma complexidade justa para o que o mesmo se propõe a fazer.