Algoritmos I

Trabalho Pratico 3

Hideki Yoshizane Olivera

10 de Fevereiro de 2022

1 Enunciado do Problema

O cenário apresentado a ser tratada é otimizar o caminho de colheitadeiras, de modo que seja possível colher o maior número de maçãs com um único caminho.

Para tratar desse problema, foi utilizado programação dinâmica em uma matriz, atualizando os valores da mesma até a última linha, onde será possível encontrar o total de maçãs colhidas pelo caminho.

2 Desenvolvimento

2.1 Solução Apresentada

Para armazenar os valores foi utilizado *vector*, um container padrão presente no C++ para armazenar os dados.

Um *vector de vector* de *unsigned long long* foi implementado para criar uma matriz onde cada célula representa uma macieira, o valor das células representando a quantidade de maçãs na macieira.

Uma segunda matriz zero, de nome *path*, com o mesmo tamanho é utilizada para armazenar o caminho.

Para uma célula [i][j] da matriz principal, é utilizada a função $max()$ da biblioteca *algorithm* padrão do C++. Os parâmetros utilizados na função são os valores das células da linha acima, sendo ela na mesma coluna ou as colunas diagonais. Tal resultado é armazenado em uma variável temporária *temp*. (Exemplo na figura 1).

A variável *temp* é utilizado na função $FindIndex()$, junto com a linha atual do looping. Essa função retorna o indexador da coluna que contém

o valor presente na variável. Após utilizado na função, o valor de *temp* é somado à [i][j].

o Indexador retornado é adicionado a célula [i][j] da matriz secundária *path*.

Ao realizar essa operação em todas as células da matriz, é utilizado novamente a função $max()$ na última linha da matriz, encontrando o resultado final de maior valor. Utilizando novamente $FindIndex()$ para encontrar o indexador da coluna desse resultado final. Esse indexador será utilizado na próxima etapa.

Primeiro, o indexador é adicionado ao vetor Caminho, sendo esse a última coluna do caminho. Indo na matriz *path*, na célula [linhaFinal][indexador], encontramos um outro indexador, este sendo de onde ele veio. Armazenamos-o no vetor Caminho e acessamos a célula com esse indexador. Realizando essa operação repetidamente até a linha 1, encontramos o caminho de valor máximo invertido.(Exemplo na figura 2).

Utilizando a função $reverse()$, para encontrarmos o caminho correto, do começo ao final.

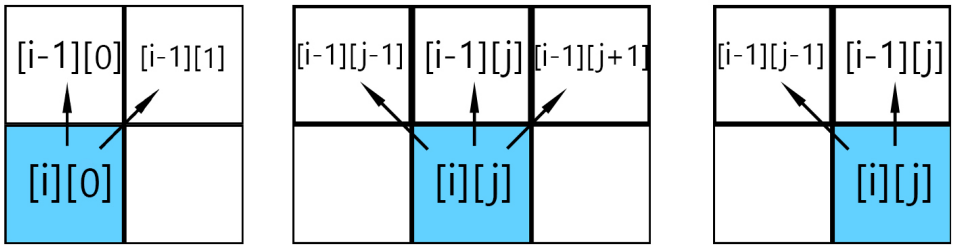

Figura 1: Exemplo visual da utilização do max

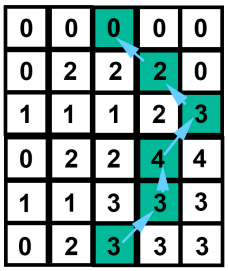

Figura 2: Exemplo visual do path invertido

2.2 Implementação

O programa em si, pode ser organizado em 3 módulos e uma função principal.

Módulo de Estruturação – Esse módulo consiste em receber os valores do arquivo e preencher a matriz.

- *StartVariable*(int linhas, int colunas, vector<vector< unsigned long long» matriz) – Recebe as variáveis vazias e em seguida insere os seus valores de acordo com o arquivo de entrada. Preenche a matriz princial.

Módulo de Programação Dinâmica – Esse módulo é responsável pela função de encontrar o indexador e a função responsável por achar o caminho de soma máxima.

- *FindIndex*(vector<vector< unsigned long long» matrix, int linha, int valor) - Método que retorna o indexador coluna do int valor.

- *MaxPath*(vector<vector< unsigned long long» matrix, vector<int> caminho, linhas,colunas) – Realiza a operação para encontrar o total de frutas coletadas, preenche a matriz temporária que será usada para achar o caminho e encontra o caminho de soma máxima invertida.

Módulo de impressão – Esse módulo contém a função que imprime o caminho máximo.

- *PrintPath*(vector<int> caminho) - Imprime o vetor caminho.

Modulo Main (Principal) – Fica a cargo apenas de receber os parâmetros, atribuir as variáveis de manipulação e chamar as funções e procedimentos mais relevantes do programa.

3 Análise de Complexidade

- *main()* – A função principal, realiza a chamada das funções e procedimentos abaixo, e portanto possui uma das chamadas mais caras.

  Complexidade Temporal: $O(N * M)$

- *StartVariable(int linhas, int colunas,vector<vector<unsigned long long» matrix)* - Insere os valores do arquivo nas respectivas variáveis.

  Complexidade Temporal: $O(N * M)$

- *FindIndex(vector<vector< unsigned long long» matrix, int linha, int valor)* - Percorre a linha da matriz recebida em busca do mesmo valor passado por parâmetro, ao encontrar, retorna o indexador da coluna.

  Complexidade Temporal: $O(N)$

- *MaxPath(vector<vector< unsigned long long» matrix, vector<int> caminho, int linhas, int colunas)* - Percorre a matriz célula a célula, atualizando seus valores para ao final da matriz ter o total de frutas colhidas. Preenche a matriz path que armazena o caminho realizado, percorrendo a mesma matriz de baixo para cima encontrando o caminho de maior soma e inserindo o caminho no vetor caminho, este invertido.

  Complexidade Temporal: $O(N * M)$

- *PrintPath(vector<int> caminho)* - Inverte o vetor caminho para encontrar a ordem correta do caminho, em seguida imprimindo-o.

  Complexidade Temporal: $O(N)$

4 Testes Experimentais

Infraestrutura Utilizada

Os testes do programa foram realizados em um computador com processador Intel Core i5-9300H, com 16GB de memória RAM. O Ambiente utilizado, foi o Windows 11.

Execução

O programa consiste de 1 modulo, um header e uma função principal, que são:

functions.cpp functions.h main.cpp

OBS: Foi criado também, um arquivo Makefile para a compilação. O programa gerado chama-se *tp*03, conforme solicitado em instruções do Trabalho Prático.