TP3 - Documentação

Nome: Rodrigo Ferreira Araújo — Matrícula: 2020006990

Algoritmos 1 - Fevereiro 2022

1 Modelagem Computacional

Para a implementação desse trabalho, criamos um tipo de dado "Matriz ", com os atributos "val", "acc", e "idx". Este tipo Matriz tem como objetivo representar uma macieira no grid da colheita descrito, de modo que "val" armazena a disponibilidade (Qi) de frutas da macieira em questão. Além disso, os atributos "acc" e "idx" corroboram a abordagem usada do problema de encontrar o caminho que maximize a colheita de frutas de cima para baixo (respeitando a movimentação da colheitadeira). No programa principal, criamos uma matriz usando esse tipo com F linhas e W colunas, com F e W lidos da entrada padrão.

A abordagem em questão consiste em uma adaptação de Programação Dinâmica Bottom-up, de modo que, de baixo para cima na matriz de colheita, calculamos, para cada célula, o maior acúmulo de frutas, que é obtido considerando a disponibilidade Qi da macieira somado ao acúmulo máximo dentre as células acessíveis pela colheitadeira na linha de baixo. Este acúmulo é armazenado no parâmetro "acc" de cada célula.

Nesse sentido, o acúmulo de frutas de uma célula é:
M[i][j].acc = M[i][j].val + max(M[i+1][j-1].acc, M[i+1][j].acc, M[i+1][j+1].acc).
Consideramos o máximo dos acumulados dentre as células acessíveis pela colheitadeira na linha de baixo: "para a macieira com mesmo alinhamento da atual, a macieira diagonalmente a direita da atual ou a macieira que se encontra diagonalmente a esquerda da atual. A colheitadeira nunca se movimenta para trás ou para os lados, ou seja, nunca para uma macieira da fileira anterior ou para uma macieira na mesma fila da atual.". Note que, caso j = 0

ou j = W - 1, consideraremos o máximo de acumulados dentre apenas duas células da linha de baixo para não extrapolar os limites da matriz.

O campo "idx" de cada célula armazena o índice da célula na linha de baixo que proporcionou o maior acúmulo de frutas, de modo que, para uma célula M[i][j], M[i][j] -> M[i+1][M[i][j].idx] forma um passo de um possível caminho de máxima colheita. Após a leitura dos limites F e W e criação da matriz no programa principal, preenchemos ela para prepará-la de acordo com a abordagem descrita: lemos os valores de disponibilidade para cada célula (val), e os campos "acc" e "idx" são inicializados com 0. No entanto, os acúmulos (acc) e os índices (idx) das células da linha mais abaixo da matriz são, respectivamente, o próprio valor de disponibilidade (val) e o índice j da célula na matriz.

2 Algoritmos e Estruturas de Dados

2.1 Estruturas de Dados

Tipo Matriz: Armazena informações imprescindíveis à uma célula do grid da colheita.
Inteiros: val, acc, idx.

vector<vector<Matriz>> Colheita(F,vector<Matriz>(W)): Matriz usando Vector (STL) principal com todas as células com as macieiras.

2.2 Funções/Algoritmos

2.2.1 max

Função que retorna o máximo entre dois valores de disponiblidade de macieiras (val). Complexidade de tempo: O(1).

2.2.2 max_3

Função que retorna o máximo entre três valores de disponiblidade de macieiras (val). Complexidade de tempo: O(1).

2.2.3 index_of_max

Função semelhante à **max**. Acha o máximo dentre dois valores de disponibilidade de macieiras (val) e retorna o índice j correspondente à esse máximo na matriz Colheita. Complexidade de tempo: O(1).

2.2.4 index_of_max_3

Função semelhante à **max_3**. Acha o máximo dentre três valores de disponibilidade de macieiras (val) e retorna o índice j correspondente à esse máximo na matriz Colheita. Complexidade de tempo: O(1).

2.3 Programa Principal (main)

Com a matriz Colheita pronta, calculamos o valor do acumulado de frutas (acc) e o índice da linha de baixo que proporcionou o maior acumulado (idx) para cada célula da matriz a partir da penúltima linha até a primeira (Bottom-up). A cada passada de linha, salvamos o valor máximo de acumulado nessa linha e o seu respectivo índice j. Desse modo, ao terminar a passada na primeira linha da matriz, teremos o valor e o índice da célula com o maior acumulado de frutas ao longo da Colheita.

Com todos os campos "idx" e "acc" registrados adequadamente, podemos, então, realizar um backtracking de cima para baixo para recuperar e imprimir os índices do caminho que maximiza a colheita. Desse modo, começando da célula do maior acúmulo na primeira linha, realizamos, para cada linha da Colheita, o seguinte: imprima o campo "idx" da célula em questão e vá para a célula na linha seguinte indexada por "idx" da célula em questão. Assim, serão impressos os índices de um possível caminho que maximiza a colheita, uma vez que o campo "idx" armazena o índice da célula na próxima linha que acumula o maior número de frutas, dada a movimentação da colheitadeira.

Análise de Complexidade de Tempo Assintótica: A complexidade de tempo assintótica do programa como um todo é a complexidade do programa principal. Assim, temos a complexidade do preenchimento da matriz Colheita através do input: $O(F \cdot W)$. Ademais, temos a complexidade do cálculo dos campos "acc" e "idx" de cada célula da Colheita, que também é $O(F \cdot W)$. Por fim, temos a complexidade do backtracking descrito, que acessa e imprime

um índice por linha da matriz, produzindo complexidade O(F). Somando as complexidades, totalizamos **O(F · W)**.

2.4 Pseudocódigo

caminho_máximo (Matriz Colheita[F][W])
for (penúltima linha de Colheita até a primeira)
| **for** (primeiro item da linha até o último)
——| calcule os campos "acc" e "idx" para esta célula
——| **if** (o acumulado obtido é o maior acumulado até então)
————| atualize o maior acumulado e o seu índice com o
————| campo "acc" recém calculado e o índice da célula em questão

imprima o maior acumulado (soma das frutas da maior colheita possível)
imprima o índice desse maior acumulado na primeira linha da matriz

for(primeira linha da matriz até a última)
| imprima o "idx" da célula indexada pelo índice do maior acumulado
| atualize o índice do maior acumulado com o valor do campo
| "idx" da célula em questão

3 Análise da Equação de Bellman-Ford

A partir do problema descrito e solucionado, podemos inferir uma equação similar à análise de Bellman-Ford com a seguinte modelagem:
OPT(i,j): Conjunto de **i** fileiras de comprimento **j** de uma plantação de macieiras em que podemos encontrar um caminho de colheita máxima.
Objetivo: **OPT(F,W)**.

Seja **M**[**F**][**W**] nossa memória, **val**[**i,j**] o valor de disponibilidade da macieira na célula [i,j] e **acc**[**i,j**] o acumulado de colheita como descrito antes para uma célula de **M**.
Começando o preenchimento de acumulados a partir da penúltima fileira de **M**, como descrito a partir da matriz colheita, podemos inferir a seguinte análise: (próx página)

OPT(i,j) =
| 0, se i = 0 e/ou j = 0,
| **val[i,j]**, se i = 1
| **val[i,j]** + **max**(**OPT(i+1,j)**, **OPT(i+1,j+1)**), se j = 1
| **val[i,j]** + **max**(**OPT(i+1,j-1)**, **OPT(i+1,j)**), se j = W
| **val[i,j]** + **max**(**OPT(i+1,j-1)**, **OPT(i+1,j)**, **OPT(i+1,j+1)**), caso contrário.