Henrique Soares Assumpção e Silva
Matrícula: 2020006620
henriquesoares@dcc.ufmg.br

Trabalho Prático III
Algoritmos I
Universidade Federal de Minas Gerais

2022-02-02

1 Introdução

Um grande produtor de Santa Catarina deseja implementar uma nova estratégia para otimizar o trabalho de colheitaderas em uma plantação de macieiras. O produtar deseja encontrar um caminho que maximize o volume final colhetado, dado certas restrições com respeito à movimentação da colheitadeira.
Formalmente, existem $1 \leq F \leq 10^6$ fileiras com a mesma quantidade $1 \leq W \leq 50$ de árvores, sendo que cada árvore produz uma quantidade $0 \leq Q \leq 10^9$ diferente de fruta. A colheitadera deverá passar por todas as fileiras, coletando frutos de apenas uma árvore por fileira, sendo que a colheitadeira pode se movimentar de acordo com três ações possíveis: (i) mover-se para a macieira com o mesmo alinhamento da atual, (ii) mover-se para a macieira que se encontra na diagonal esquerda e (iii) mover-se para a macieira que se encontra na diagonal direita. A partir dessas regras e parâmetros, deseja-se construir um algoritmo eficiente que encontre o caminho ótimo que deve ser realizado pela colheitadeira.

2 Solução

Para resolver o problema em questão, optou-se por utilizar uma abordagem dinâmica. Primeiramente será fornecida a intuição em alto nível da solução proposta, e então uma prova formal de sua corretude.
A intuição por trás da solução fornecida consiste em quebrar o problema principal em problemas menores de modo que a solução de um dado problema pode ser utilizada para construir a solução do problema subsequente. Primeiramente, iremos construir uma matriz dos chamados *Scores* para cada macieira, que nada mais são do que a soma da quantidade de fruta disponível para a macieira atual com o maior score da fileira anterior, ou seja, o score de uma dada macieira j na fileira i pode ser calculado a partir do maior score das macieiras em $i-1$ que conseguem chegar a i de forma válida. Para uma macieira m qualquer, o seu Score representa **o maior volume de fruta que pode ser coletado dado que a colheitadera termine seu caminho em m**, isto é, o score armazena a informação do melhor caminho que termina nessa dada macieira. Após computarmos os scores para todas as macieiras, podemos fazer o caminho contrário para obter o caminho ótimo: começamos da última fileira e escolhemos a macieira com o maior score, e então andamos para a fileira anterior escolhendo a macieira com o maior score até chegarmos na fileira inicial, e esse caminho percorrido será justamente um dos caminhos ótimos possíveis. Esse fato é intuitivo dado a construção feita, sendo que estamos atribuindo a cada macieira um valor que codifica o volume total de frutas que podem ser coletadas a partir de um dos melhores caminhos possíveis que passam por esta, ou seja, as macieiras com maior score da última fileira permitem a construção de um caminho que por definição deve possuir o maior volume de frutas coletadas possível. O algoritmo proposto, portanto, possui duas etapas distintas: (i) obter os scores para todas as macieiras e (ii) reconstruir o caminho ótimo a partir deste conjunto de scores.
Agora, formalizaremos a solução proposta e provaremos sua otimalidade. Seja A a matriz de tamanho $F \times W$, onde cada elemento da matriz representa um valor de Q para uma dada macieira. Denotaremos por a_{ij} a j-ésima macieira da fileira i, e portanto $q(a_{ij})$ representa a quantidade de frutas que podem ser coletadas nessa macieira. Denotaremos por $s(a_{ij})$ o valor do score para a macieira a_{ij}. A partir das instruções do enunciado, podemos os seguintes casos para o cálculo do score para uma dada macieira:

1. $s(a_{ij}) = q(a_{ij})$ se a macieira se encontra na primeira fileira, i.e., se $i = 0$;
2. $s(a_{ij}) = q(a_{ij}) + max(s(a_{(i-1)(j-1)}), s(a_{(i-1)j}), s(a_{(i-1)(j+1)}))$ se a macieira se encontra em alguma fileira que não a primeira e em uma posição que não é uma extremidade, i.e., se $i > 0$ e $0 < j < W - 1$;
3. $s(a_{ij}) = q(a_{ij}) + max(s(a_{(i-1)(j-1)}), s(a_{(i-1)j}))$ se a macieira se encontra em alguma fileira que não a primeira e na extremidade esquerda, i.e., se $i > 0$ e $j = 0$;
4. $s(a_{ij}) = q(a_{ij}) + max(s(a_{(i-1)(j)}), s(a_{(i-1)(j+1)}))$ se a macieira se encontra em alguma fileira que não a primeira e na extremidade direita, i.e., se $i > 0$ e $j = W - 1$;

Agora, considere o seguinte algoritmo em pseudocódigo para obter o caminho ótimo dado a matriz A em questão:

Algorithm 1 Solução
Entrada: $A \in \mathbb{N}^{F \times W}$,$F$,$W$
Saída: Caminho de macieiras que maximiza o volume coletado

Obtenha $s(a_{ij})$,$\forall a_{ij} \in A$
Seja C o vetor que conterá os índices das macieiras que compoe o caminho ótimo
for $i \in F$ **do**
  $C[i] \leftarrow 0$
end for
$C[F-1] \leftarrow j_{max}$, onde j_{max} é o índice da macieira da última fileira com maior score
for $i \in \{F-2, F-3, ..., 0\}$ **do**
  if $0 < C[i+1] < W-1$ **then**
    Dentre os índices $\{C[i+1]-1, C[i+1], C[i+1]+1\}$, obtenha j^* cujo score $s(a_{ij^*})$ é máximo
    $C[i] \leftarrow j^*$
  else
    if C[i+1] = 0 **then**
      Dentre os índices $\{C[i+1], C[i+1]+1\}$, obtenha j^* cujo score $s(a_{ij^*})$ é máximo
      $C[i] \leftarrow j^*$
    else
      Dentre os índices $\{C[i+1]-1, C[i+1]\}$, obtenha j^* cujo score $s(a_{ij^*})$ é máximo
      $C[i] \leftarrow j^*$
    end if
  end if
end for
  return C

O algoritmo realiza o caminho inverso do cálculo do score, primeiro encontrando o maior score da última fileira e então reconstruíndo um caminho possível para obter tal score. O caminho final obtido, por definição, é um sequência de macieiras que fornece um volume final de colheita máximo, pois a construção de tais scores é tal que um score qualquer armazena informação sobre diversos caminhos possíveis passando por fileiras anteriores.
Agora provaremos formalmente que o algoritmo fornece uma solução ótima. Primeiramente, seja $S = \{a_1, a_2, ..., a_F\}$ a sequência de macieiras qualquer, onde cada $a_i \in S_a$ representa uma macieira da i-ésima fileira da matriz de entrada. Para uma dada sequência S de macieiras, denotamos por $V(S) = \sum_{m_i \in S} q(m_i)$ como sendo o volume total que pode ser coletado se seguirmos essa dada sequência de macieiras, isto é, $V(S)$ é o somatório da quantidade de fruta disponível para cada macieira m_i da sequência S. Queremos portanto mostrar que o caminho encontrado pelo algoritmo possui $V(S)$ máximo.

Proposição 1. *O algoritmo proposto encontra uma sequência de macieiras tal que o volume total de fruta que pode ser coletado ao se percorrer essa sequência é máximo, i.e., dado uma sequência S_a encontrada pelo algoritmo, $V(S_a)$ é máxima, e portanto não existe nenhuma sequência S' tal que $V(S') > V(S_a)$*

Proof. Podemos realizar essa prova com uma indução em F. A intuição da prova é que, para um tamanho de F qualquer, escolher o caminho obtido por meio da reconstrução do algoritmo para a macieira de maior score na F-ésima fileira da matriz (última fileira) é sempre a melhor escolha.
Passo base: Seja $F = 1$. Nesse caso, para todas as macieiras $a_{0j} \in \{a_{00}, a_{01}, ..., a_{0(W-1)}\}$, vale que $s(a_{0j}) = q_{0j}$, isto é, o score de cada macieira é simplesmente o valor de fruta disponível. O algoritmo nesse caso escolhe a macieira de maior score, i.e., a solução do algoritmo necessariamente fornece um volume total coletado máximo. Qualquer outra solução para ser considerada ótima deverá escolher essa mesma macieira ou outra que possua o mesmo valor de fruta disponível, e portanto vale que o algoritmo fornece um caminho ótimo para $F = 1$. Além disso, também vale que para qualquer a_{0j}, $s(a_{0j})$ fornece o caminho com o maior volume total coletado que termine em a_{0j}. Esse resultado é trivial no caso de $F = 1$, pois de fato todo caminho nessa situação começa e termina em uma mesma macieira, e portanto o score de qualquer macieira também representa o maior volume que pode ser coletado por um caminho que termine nesta macieira.
Hipótese Indutiva: Para qualquer $F \leq n, n \in \mathbb{N}^+$, vale que o caminho obtido pelo algoritmo, i.e., escolher a macieira de maior score da última fileira e então escolher as macieiras anteriores de maior score, fornece uma solução ótima para o problema. Além disso, para qualquer a_{nj}, vale que $s(a_{nj})$ fornece o maior volume total que pode ser coletado por um caminho que termine em a_{nj}. Note que esse volume não é necessariamente a solução do problema, e sim a melhor solução que pode ser obtida se a última macieira do caminho for a_{nj}.
Passo Indutivo: Considere $F = n + 1$. Sabemos que existem $n + 1$ fileiras com W macieiras cada. Sabe-se que, para maximizar o volume total coletado, **não** basta escolhermos o melhor caminho que pode ser feito até a n-ésima fileira e então adicionarmos a macieira da $(n + 1)$-ésima fileira com maior quantidade de fruto

disponível, pois pode existir alguma macieira nessa última fileira que é inancalcável a partir do melhor caminho prévio e que possui uma quantidade de fruto disponível que faça com que outro caminho que até então não era ótimo se torne ótimo. Portanto, para obter a melhor solução para o caso de $F = n + 1$, devemos considerar os melhores caminhos que terminam em cada uma das macieiras da n-ésima fileira – ou seja, para cada macieira dessa fileira considerar, dentre todos os caminhos possíveis que terminam nela, aquele que possui maior volume total – e então, para cada um desses caminhos, escolher a macieira da $(n + 1)$-ésima fileira que possua maior quantidade de fruta disponível, e a partir disso devemos escolher o caminho que agora possui maior volume total.
Sabemos que o algoritmo nesse caso iria computar os scores para todas as macieiras da $(n + 1)$-ésima fileira e escolher a macieira dessa fileira com maior score para reconstruir o caminho ótimo. Lembre que, para uma macieira $m_{(n+1)j}$ qualquer na ùltima fileira, o seu score pode ser obtido por meio da soma de seu valor de fruta disponível ($q_{(n+1)j}$) com o maior score dentre as macieiras da fileira anterior que conseguem alcançar $m_{(n+1)j}$ de forma válida, i.e., estão imediatamente atrás ou em alguma das diagonais. No entanto, por hipótese sabemos que os scores dessas macieiras da fileira anterior também representam os maiores volumes totais que podem ser coletados em qualquer caminho que termine em cada uma dessa macieiras, ou seja, a macieira da $(n + 1)$-ésima fileira com maior score agora representa o caminho ótimo. Portanto, percebe-se que para o caso de $F = n + 1$, o algoritmo está realizando um processo análogo ao procedimento descrito no último parágrafo: como por hipótese os scores das macieiras da n-ésima fileira já representam o maior volume total que pode ser coletado por um caminho que termine em cada uma destas macieiras, o algoritmo está justamente considerando todos esses caminhos e escolhendo aquele que, após se considerar a adição da macieira da $(n + 1)$-ésima fileira que é alcançavel a partir desse caminho e que possui maior quantidade de fruta disponível, possui maior volume total de fruta a ser coletado.
Portanto, a indução em F realizada é suficiente para demonstrar que o algoritmo calcula de fato um caminho ótimo que é solução do problema.

□

3 Modelagem

O programa foi desenvolvido na linguagem C++, compilada pelo compilador G++ da GNU Compiler Collection, no sistema operacional Windows 10. O programa foi inicialmente implementado e compilado em uma máquina com 16GB de RAM em um processador Intel Core I5-9400F.

3.1 Funções

O problema em questão foi solucionado em termos de implementação por meio de três funções que serão especificadas a seguir:

- `void calculate_scores(long int scores, int num_rows, int num_trees)`:
  Função que calcula os scores para cada macieira disponível e os armazena em uma matriz, de acordo com a definição de Score fornecida na seção 2. Itera sobre as fileiras e macieiras e armazena o score calculado no respectivo elemento da matriz de scores;

- `build_opt_path(int* &path)`
  Função que constrói o caminho ótimo a partir dos scores calculados. Seus argumentos de entrada podem ser encontrados no código fornecido e foram omitidos devido à sua extensão. Armazena os índices da macieiras de cada fileira que formam o caminho ótimo em um vetor de inteiros.

Uma função `read_data` também foi implementada para auxiliar na leitura dos dados de entrada, mas como sua assinatura é extensa e seu comportamento é trivial, optou-se por omitir sua descrição detalhada nesta documentação.

3.2 Programa principal

Esta seção ia descrever o funcionamento da função principal `main` do programa desenvolvido. Os passos do programa principal são descritos a seguir:

1. Primeiramente os dados de entrada são lidos por meio da função `read_data`, e os valores de Q são armazenados em uma matriz chamada *grid*;

2. Após isso, uma cópia dessa matriz é criada e então os valores dos scores são armazenados nessa, a partir do uso da função `void calculate_scores`;

3. Os scores então são passados para a função `build_opt_path`, que então obtém o caminho que maximiza o volume final coletado, que então é impresso na saída padrão. Os procedimentos de limpeza de memória são realizados e o programa encerra.

4 Análise de Complexidade

Nesta seção, será realizada a análise de complexidade de tempo e espaço do algoritmo proposto. As complexidades descritas serão expressas em termos do número de fileiras F e do número de árvores por fileira W.

4.1 Análise de espaço

Optou-se por utilizar matrizes para representar os dados em questão, e portanto é necessário um espaço inicial de ordem quadrática em qualquer situação do programa. Uma matriz auxiliar é criada para armazenar os scores, e as funções implementadas realizam apenas operações de ordem constante com respeito ao espaço, ou seja, podemos afirmar que a ordem de complexidade de espaço do programa em qualquer caso é $\Theta(F \cdot W)$.

4.2 Análise de tempo

O algoritmo que apresenta a solução para o problema tem duas partes: (i) cálculo dos scores para cada árvore, e (ii) cálculo do caminho ótimo dado os scores. O passo (i) requer iterar sobre todos os elementos da matriz inicial, e portanto realiza operações da ordem de $\Theta(F \cdot W)$ independente da entrada fornecida. O passo (ii) primeiro encontra o maior elemento da última fileira da matriz, que é da ordem de $\mathcal{O}(W)$, e então itera sobre as demais fileiras realizando sempre um número constante de operações, ou seja, o segundo passo do algoritmo realiza operações da ordem de $\mathcal{O}(W + F)$, e poranto a complexidade de tempo final do programa em qualquer caso é de $\Theta(F \cdot W)$.

References

[1] Almeida, J. (2021). Slides virtuais da disciplina de Algoritmos 1. Disponibilizado via moodle. Departamento de Ciencia da Computação. Universidade Federal de Minas Gerais. Belo Horizonte.

[2] Cormen, Thomas H. Algoritmos: Teoria e Prática, Editora Campus, v. 2 p. 296, 2002