Ciência da Computação

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI

AEDS III Trabalho Prático

Alunos:
Vinicius de Almeida Lima (212050070)
Matheus Duraes da Cunha Pereira (212050094)

Professor: Leonardo Rocha

Introdução

No trabalho prático de AEDS 3 nos foi proposto que criássemos um programa para calcular o número máximo de triângulos que podem ser formados com duas âncoras sem que os triângulos colidam entre si.

Para solucionarmos o problema utilizamos de um algoritmo que encontra a maior sequência, utilizando um procedimento não recursivo, sua complexidade é de $O(n^2)$, porem isso não será um problema visto que foi informado que o número de entradas não será maior que 100.

Implementação

1.0 Estruturas de Dados;

Para a criação do programa foram criadas 3 estruturas de dados sendo elas

- 1.0.1 Ponto
  Formado por dois int x, y;

- 1.0.2 Âncora
  Formado por dois Pontos, referente às duas Âncoras.

- 1.0.3 Triângulo
  Formado por três Pontos, referente a cada ponto de um triangulo.

1.1 Função inputFile*:*

função que lê dados de um arquivo e os armazena na memória. Leva três parâmetros: uma string fileName que especifica o nome do arquivo a ser aberto, um ponteiro para um ponteiro P do tipo Ponto e um ponteiro ancora do tipo Ancora. Ele tenta abrir o arquivo especificado por fileName no modo somente leitura usando a função fopen(). Se o arquivo não puder ser aberto, ele imprime uma mensagem de erro e retorna. Ele lê a primeira linha do arquivo, que deve conter três inteiros separados por espaços. O primeiro inteiro especifica o número de pontos de dados a serem lidos, enquanto o segundo e o terceiro inteiros especificam as coordenadas x de dois pontos âncora. Ele inicializa as coordenadas dos dois pontos de ancoragem em 0.

Função outputFile:

a função outputFile() é usada para escrever um valor inteiro em um arquivo especificado por fileName. Se o arquivo não puder ser aberto, a função imprime uma mensagem de erro.

Função ioFlags:

No geral, a função ioFlags() lê as opções de linha de comando fornecidas ao programa e executa as ações apropriadas com base nessas opções. Ele pode abrir um arquivo de

entrada e armazenar seus dados na memória, manipular um arquivo de saída ou lidar com outros tipos de entrada/saída dependendo das opções especificadas.

Função TrianguloEstaDentroDoTriangulo:

A função trianguloEstaDentroDoTriangulo() em C recebe dois parâmetros de entrada, t e p, que são do tipo Triângulo. Ele retorna um valor inteiro indicando se p está dentro de t. A função primeiro verifica se t e p são iguais. Se estiverem, retorna 0, indicando que p não está dentro de t.
A função calcula as áreas com sinal de três triângulos: (t.a, t.b, p.c), (t.b, t.c, p.c) e (t.c, t.a, p.c). Para calcular a área com sinal de um triângulo, ele usa o produto vetorial de dois vetores: (p1.x - p0.x, p1.y - p0.y) e (p2.x - p0.x, p2.y - p0 .y). O sinal do produto vetorial determina se a área é positiva ou negativa. A função verifica se alguma das áreas com sinal é negativa. Se for, ele define um sinalizador negativo para 1. A função verifica se alguma das áreas com sinal é positiva. Se não, ele define um sinalizador positivo para 1. Se negativo e positivo forem 1, significa que p está fora de t e a função retorna 0. Caso contrário, significa que p está dentro de t e a função retorna 1.

Função criarTriangulos:

Esta função cria triângulos usando os pontos de entrada e os pontos de ancoragem. Leva um ponteiro para um ponteiro de pontos P, um ponteiro para as âncoras e um ponteiro para um ponteiro de triângulos T como parâmetros de entrada. A função aloca memória dinamicamente para os triângulos e atribui os vértices do primeiro triângulo aos pontos de ancoragem e ao primeiro ponto na matriz de pontos de entrada. Em seguida, itera pelos pontos restantes na matriz de pontos de entrada e cria um novo triângulo para cada ponto usando os pontos âncora e o ponto atual como vértices. Finalmente, a função atualiza o campo de comprimento do ponteiro *T para o número de triângulos criados.

Função compararTriangulos():

Esta função é uma função de comparação usada pela função qsort() para classificar uma matriz de triângulos. A função qsort() usa um ponteiro para uma matriz, o número de elementos na matriz, o tamanho de cada elemento na matriz e um ponteiro de função de comparação como parâmetros de entrada. Ele classifica a matriz em ordem crescente de acordo com a função de comparação. A função compararTriangulos() recebe dois ponteiros void a e b, que são convertidos em ponteiros Triangulo* dentro da função. Em seguida, compara a coordenada y do terceiro vértice dos dois triângulos (t1 e t2). Se a coordenada y de t1 for maior que a de t2, a função retornará -1, indicando que t1 deve vir antes de t2 na matriz classificada. Se a coordenada y de t1 for menor que a de t2, a função retornará 1, indicando que t2 deve vir antes de t1 na matriz classificada. Se a coordenada y de t1 for igual à de t2, a função retorna 0, indicando que a ordem de t1 e t2 não importa.

Função maiorSequencia():

A função maiorSequencia recebe um array de objetos Triangulo (T) e retorna um inteiro representando o comprimento da maior subsequência de objetos Triangulo que estão dentro um do outro. Ele primeiro inicializa duas variáveis: max com valor 0 (para

armazenar o comprimento máximo da sequência) e count com valor 1 (para contar o comprimento da sequência atual). Em seguida, ele percorre cada elemento na matriz T, definindo aux para o objeto Triangulo atual e redefinindo a contagem para 1. Dentro do loop, ele percorre os elementos restantes do array T (começando no próximo índice após i), verificando se cada objeto Triangulo subseqüente está dentro do objeto aux Triangulo usando a função trianguloEstaDentroDoTriangulo. Se for, ele define aux para o novo objeto Triângulo e incrementa a contagem. Após a conclusão do loop interno, se count for maior que max, ele atualizará max para ser igual a count. Por fim, a função retorna o valor de max, que representa o comprimento da maior subsequência de objetos Triangulo que estão dentro um do outro.

4.0 Resultados obtidos:

Número de entradas	Tempo de CPU em Milissegundos
2	2.138
4	1.574
14	2.138
20	2.650
30	2.519
40	3.653
50	3.223
75	2.205
100	2.065

Foi observado que o tempo de execução varia pouco, isso se deve ao fato de que a ordem de complexidade do algoritmo para encontrar a maior sequência ser $O(n^2)$ e o tempo de execução de um algoritmo $O(n^2)$ não varia para diferentes entradas porque o número de operações executadas pelo algoritmo é diretamente proporcional ao quadrado do tamanho da entrada. Isso significa que, independentemente dos valores específicos da entrada, o algoritmo $O(n^2)$ sempre terá o mesmo comportamento de tempo de execução em termos de complexidade.

5.0 conclusão:

Em geral concluímos que a experiência da realização do trabalho foi proveitosa e envolveu de forma coerente os conteúdos trabalhados na matéria de AEDS 3 . Sendo portanto uma maneira efetiva e desafiadora de aprendizado e prática.

A principal dificuldade encontrada foi descobrir uma forma de contabilizar a maior sequência, estratégia utilizada é uma solução de força bruta, onde para cada triângulo no vetor, é verificado se ele está contido em algum dos triângulos restantes. Embora a solução de força bruta apresentada seja simples e direta, ela pode ser ineficiente para casos com muitas entradas.

Referências

Para realizar esse trabalho usamos os slides dispostos no Campus Virtual da Universidade Federal de São João Del-Rei (UFSJ), na disciplina de Algoritmos e Estrutura de Dados III: https://campusvirtual.ufsj.edu.br/portal/2023_1n/course/view.php?id=1198