ISOSCELES TRIANGLES

Autores

Bruno Villas Bôas da Costa

Guilherme Augusto Monteiro dos Santos

Henriete Hossi Iamarino

Wagner Takeshi Obara

Triângulo Isosceles

- Um triângulo dado pode ser classificado de acordo com os limites das proporções relativas de seus lados.
- Quando um triângulo possui apenas dois lados congruentes, é chamado de isósceles.

O Objetivo

- Dada uma seqüência de coordenadas no plano $\mathbb{R}^2$ (sem possuir três pontos na mesma linha), calcular quantas são as possíveis escolhas de vértices que podem formar triângulos isósceles.

O Algoritmo

O algoritmo lê um arquivo de texto:

- A primeira linha contém um inteiro N informando a quantidade de pontos da seqüência.
- As próximas N linhas descrevem coordenadas formadas de dois inteiros X e Y, pertencentes ao plano $\mathbb{R}^2$, separados por um espaço simples.
- Deve calcular a quantidade de triângulos isósceles que são possíveis formar com as N coordenadas dadas.
- Ao encontrar o inteiro $N = 0$ ele finaliza a execução.
- Após ler cada conjunto, calcula a quantidade de isósceles presentes e salva esses valores em um arquivo de saída.

Classes

- Implementado em C++.
- Classes:
  - Ponto;
  - Programa.

Solução Proposta

- Lê o inteiro N e, cria um vetor de pontos (classe Ponto) e o ‘popula’ com as próximas N linhas do arquivo.
- Calcula a quantidade de isósceles desse conjunto de pontos (‘quantidadeIsosceles(Ponto *p, int tam)’) e armazena no arquivo de saída.

‘quantidadeIsosceles(Ponto *p,int tam)’

Caso Base

- tam = 3.
  - Retorna 1 se formarem isóscele;
  - Caso contrário retorna 0.

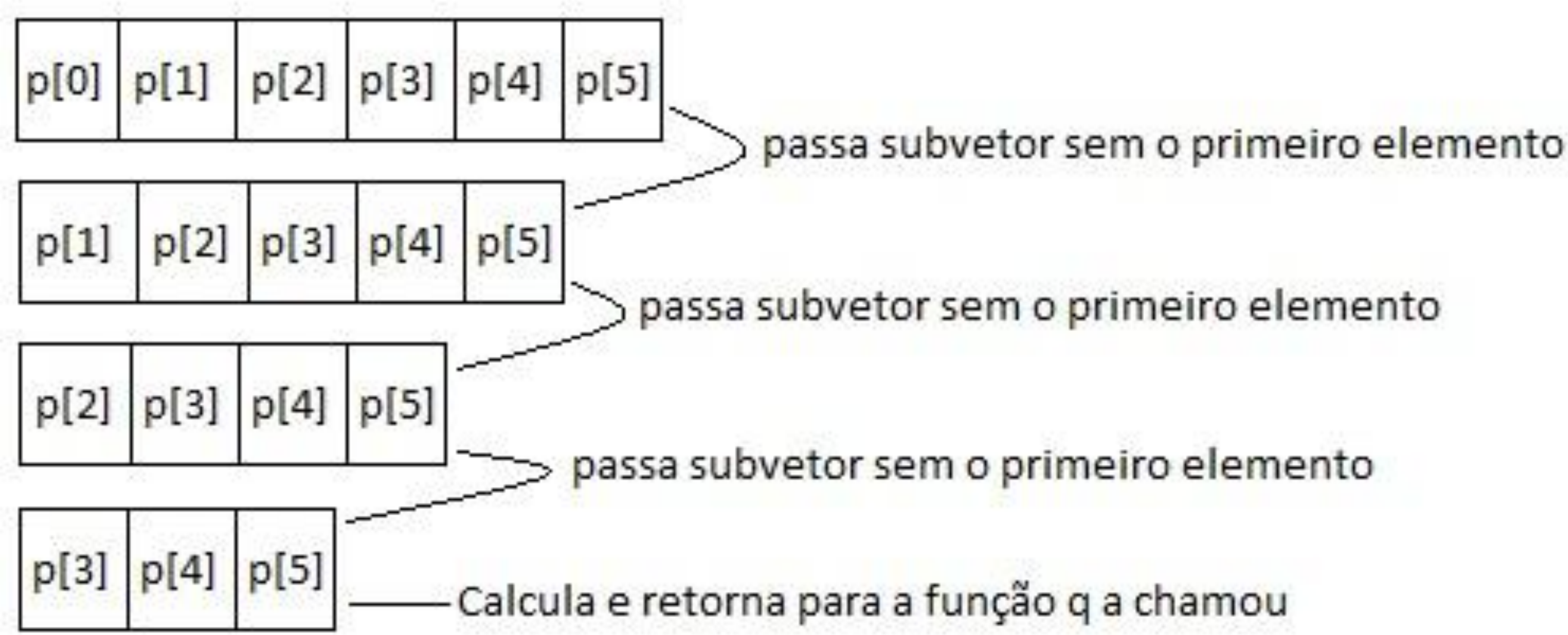
- Caso não se enquadrar no caso base:
  - Cria-se inteiro ‘qtdIsosceles’(0).

  - Laço de repetição:

```
for (int i = 1; i < tam - 1; i++) {
        for (int j = 1; j < vezes; j++) {
                qtdIsosceles += calculaIsosceles(p[0], p[i], p[i + j]);
        }
        vezes--;
}
```

  - Cria *p2;
  - Chama a função por recursão com parâmetros p2 e tam-1;

Exemplificando...



Análise da complexidade do algoritmo

$$T(n) = \begin{cases} 0, & n = 3 \\ T(n-1) + n, & n > 3 \end{cases}$$

- Quando o número de pontos dados é igual a '3', seu tempo é constante, portanto T(n) = 0 neste caso.
- Então a complexidade do algoritmo pertence à classe $\Theta(n^2)$.

Executando...

Conclusão

- Conseguimos entender melhor como planejar algoritmos que utilizem menos memória e que sejam assintoticamente mais rápidos.
- Como devemos realmente passar por todos os pontos e fazer todas as verificações, torna-se inviável aumentar a velocidade deste algoritmo.

Bibliografia

- Slides vistos em aula.

Obrigado!