Trabalho Prático 2
Triângulos

Davi Santos Rodrigues - 2022043752

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

davisrodrigues@ufmg.br

1. Introdução

Este trabalho abordou a resolução de dois problemas pedagógicos. O primeiro: usar um algoritmo guloso para achar o maior triângulo isósceles possível em uma parede de tijolos; o segundo: achar o menor triângulo possível em um conjunto de árvores.

O trabalho visa testar e aprofundar os aprendizados sobre os paradigmas apresentados em sala e sobre a linguagem proposta para desenvolver o código. O código deve ser eficiente para casos muito grandes.

2. Modelagem

Modelei os problemas utilizando vector, da stl, como estrutura principal. Para o primeiro problema, cada altura de cada pilha era o valor armazenado naquela posição do vector. Para o segundo, criei uma classe ponto, para representar cada uma das árvores e uma struct Triângulo para receber o menor perímetro dos retornos recursivos. Como base para o meu algoritmo que resolve o segundo problema, utilizei o algoritmo dado em sala “Closest Pair”, mas adaptei-o para a situação, utilizando de princípios matemáticos, principalmente.

3. Solução

Parte 1:

Desenvolvi um algoritmo com base no fundamento dos algoritmos gulosos: tomar a melhor decisão a cada iteração. Se nessa iteração, a pilha tiver potencial para ser o maior triângulo, o algoritmo testa se é válido e, se for, atualiza. O que é interessante mais interessante no algoritmo é: ele não verifica ambos os lados por pilha. Na realidade, ele já constroi a maior rampa da esquerda conforme percorre o vetor, de maneira bem elegante: não importa a altura da pilha, ela só pode contribuir com +1 em altura em relação a altura anterior, quando a pilha diminui, ele reconstroi a escada degrau por degrau.

Parte 2:

Para resolver o problema dois adaptei o algoritmo dado em sala na aula de paradigmas, “Closest Pair”. O algoritmo funciona quando os pontos estão ordenados por coordenada x, permitindo dividir o conjunto ao meio eficientemente.

A estratégia segue a metodologia de divisão e conquista. A cada passo, o conjunto de pontos é dividido ao meio pelo índice, gerando dois subproblemas: um com os pontos à esquerda da linha vertical e outro com os pontos à direita. O algoritmo retorna, recursivamente, o menor triângulo encontrado em cada lado. Além disso, também se verifica a possibilidade de o menor triângulo envolver pontos de ambos os lados da divisão, caso em que dois dos vértices pertencem a um lado e o terceiro ao

outro. O caso base ocorre quando o subproblema possui 10 ou menos pontos. Nesse caso, resolve-se diretamente por força bruta, testando todas as combinações possíveis de triângulos.

Para avaliar se o triângulo se encontra através dos lados, ele delimita uma faixa em que se os pontos estão além do limite é impossível eles participarem do menor triângulo. Matematicamente, para avaliar a faixa consideremos o menor dos perímetros da esquerda e da direita como δ. Se o triângulo contém um dos lados $l >= \delta/2$ é impossível que ele possua perímetro menor que δ e seja um triângulo válido. Postulado isso, a faixa adaptada é δ/2 e o algoritmo filtra os pontos além dessa distância.

Em conseguinte, ele orienta os pontos verticalmente, e para cada ponto na faixa, começando do mais baixo, checa os possíveis triângulos formáveis, se achar um menor que δ, atualiza δ com o menor valor e os pontos que o compõem. O argumento matemático para que, para cada ponto, haja um número constante de operações e que o problema não é n^2 é que, é impossível que para qualquer ponto à mais de δ/2 do ponto Si da iteração atual, fazer parte de um triângulo com ele que possua um perímetro menor que δ, então não os incluímos.

Dessa forma, traçando um quadrado δ/2 x δ/2 em ambos os lados da faixa, com a aresta mais baixa cruzando Si, quase temos a prova. Mas, e se todos os pontos da faixa estiverem inscritos no quadrado? O resto da prova mostra que é impossível que haja mais do que um número constante de pontos neste quadrado.

Suponha que dividimos o quadrado em quadrados menores δ/4xδ/4, dessa forma é impossível que existam 3 pontos inscritos neste mesmo quadrado, pois se houvesse, o perímetro obrigatoriamente seria menor que δ, e seria uma contradição, pois este seria o triângulo que deveria ser retornado recursivamente. Dessa forma, podem existir no máximo 2 pontos por quadrado δ/4xδ/4 e extrapolando para o quadrado δ/2xδ/2, no máximo 32 pontos. Uma quantidade constante, que pode ser melhorada filtrando os pontos que estão a mais de δ/2 do Si, para no máximo 25 pontos além de Si, que, testando os triângulos possíveis, dão aproximadamente 300 operações por Si.

4. Análise de Complexidade

Parte 1:

Espaço: O algoritmo resolve o problema in-place em um array, onde cada elemento é uma pilha. Para n pilhas, é $O(n)$.

Tempo: O algoritmo percorre o vetor uma vez e para cada elemento, se ele for candidato a maior pirâmide, analisa se realmente é. Para uma parede com muitos buracos e flutuações, a performance do algoritmo melhora bastante e aproxima de $O(n)$, mas para o pior caso, a parede totalmente crescente, é $O(n^2)$, pois para cada elemento ele verificará se é uma pirâmide válida e passará várias vezes nos mesmos elementos.

Parte 2:

Espaço: O vetor de pontos ocupa um espaço $O(n)$. Mas, para cada iteração o algoritmo cria um vetor auxilar $O(s)$, s sendo o tamanho do subproblema, e adiciona mais chamadas na pilha de recursão, que possui profundidade $O(\log n)$.

Tempo: A complexidade em tempo segue a complexidade do algoritmo Closest Pair que é $O(n\log^2 n)$, pela ordenação em cada recursão.

5. Considerações Finais

Realizar esse trabalho prático foi um satisfatório desafio na área de paradigmas e resolução de problemas. Foi desafiador desenvolver o pensamento matemático para provar que o algoritmo era menos que quadrado na parte 2 e encontrar uma lógica gulosa para a parte 1 não era muito óbvio. Os resultados foram satisfatórios e produziram dois algoritmos com aspectos elegantes nas suas estruturas em diferentes pontos. Produzir os algoritmos foi uma boa oportunidade para aprofundar os princípios matemático-computacionais do curso.

6. Referências

MARQUES, Jussara. Divisão e Conquista: Closest Pair. Material de aula (slides). Algoritmos I. Disponível no moodle da disciplina. Acesso em: 11/2025.