Universidade de São Paulo
IME-USP

Relatório Final
MAC0331

Gabriel Fernandes de Oliveira - 9345370
Luis Gustavo Bitencourt Almeida - 9298207

1 Projeto

O projeto implementado foi o do Par de pontos mais próximos. Programamos, como orientado, as quatro soluções: Força bruta, Aleatorizado como descrito no Tardos [1], o algoritmo de Divisão e Conquista visto em sala e um algoritmo de Line Sweep [2].

2 Teoria

Não serão tratados aqui os algoritmos Força Bruta e Divisão e Conquista, que foram abordados em classe.

2.1 Line Sweep

A ideia do algoritmo é analisar os pontos da entrada em ordem crescente de coordenada x, mantendo o par de pontos mais próximo dentre os pontos analisados até o momento e uma ABBB armazenando os pontos que podem participar do par de pontos mais próximos.

Seja $minDist_i$ a menor distância entre um par de pontos de índice até $i-1$, manteremos esse valor atualizado durante nosso algoritmo.

A análise de cada ponto, p, se dá em três etapas:

1. Limpa-se a ABBB, retirando da mesma todos os pontos com coordenada x menor que $p.x - minDist_i$, pois é impossível que esses pontos atualizem a distância mínima da próxima iteração, $minDist_{i+1}$

2. Verifica-se se o ponto p forma, com os pontos armazenados na ABBB, um par de pontos mais próximos que o par mais próximo mantido até o momento

3. Atualiza-se a ABBB inserindo na mesma o ponto p

O passo 1. da análise descrita é feito em complexidade amortizada $\mathcal{O}(n \lg n)$, ou $\mathcal{O}(\lg n)$ por análise de um novo ponto, sendo n o número de pontos da entrada. Isso porque cada ponto da entrada deverá entrar e sair da ABBB no máximo uma vez, por isso, ao final da análise dos n pontos, o algoritmo realizaria, no pior dos casos, $2n$ inserções ou remoções da ABBB, cada uma delas sendo realizada em tempo $\mathcal{O}(\lg n)$, por isso, a complezidade final fica da ordem $\mathcal{O}(n \lg n)$ e a complexidade amortizada $\mathcal{O}(\lg n)$.

O passo 2. se realiza a partir de buscas na ABBB, devem se buscar os pontos que pertencem à ABBB e possuem a coordenada y com valor entre $y - minDist_i$ e $y + minDist_i$, a área cinza escura representada na figura 1. Esse passo é realizado em complexidade $\mathcal{O}(i \lg n)$, sendo i o número de pontos no dado intervalo. Com uma prova totalmente análoga ao usado na análise do algoritmo de Divisão e Conquista, podemos provar que i é no máximo 8, deste modo, o passo 2 é resolvido também em tempo $\mathcal{O}(\lg n)$, assim como o passo 1.

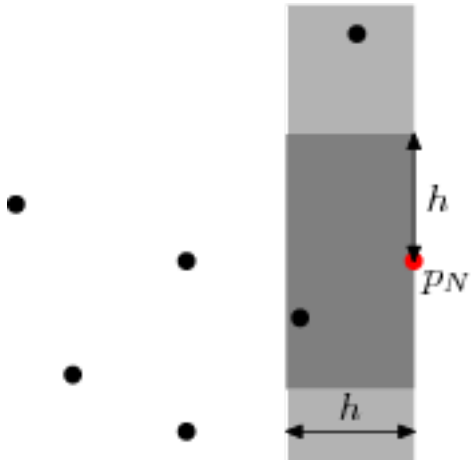

Figure 1: Exemplo do passo 2 de análise do ponto pn

O passo 3, que consiste de uma simples inserção na ABBB também ocorre em tempo $\mathcal{O}(\lg n)$

Portanto, como cada uma das três etapas da análise do line sweep é feita em complexidade $\mathcal{O}(\lg n)$, e como essa análise será feita para cada um dos pontos da entrada, concluímos que o algoritmo final roda em tempo $\mathcal{O}(n \lg n)$.

2.2 Randomizado

A implementação e análise desse algoritmo foram totalmente baseadas no livro de Tardos [1], no capítulo 13, subseção 7.

A ideia do algoritmo é

References

[1] Jon Kleinberg e Éva Tardos, *Algorithm Design*. Addison-Wesley Professional; 1ª Edição

[2] Post de bmerry sobre algoritmos de Line Sweep
`https://www.topcoder.com/community/competitive-programming/tutorials/line-sweep-algorithms/`