

# Geometria Computacional

Departamento de Ciência da Computação – IME/USP  
Segundo Semestre de 2009

## Lista 1

1. Considere o retângulo  $d \times 2d$  cuja base está sobre um ponto  $p$  da faixa usada no algoritmo COMBINE. Mostre uma coleção de pontos em que este retângulo contenha o maior número possível de pontos. Considere duas possibilidades: o caso em que a coleção pode ter pontos repetidos, e o caso em que ela não tem pontos repetidos.
2. [CLRS 33.4-1] O Prof. Maqui Sperto teve uma idéia genial e veio com um novo esquema para que o algoritmo encontre o par mais próximo verificando, no COMBINE, somente a distância entre cada ponto  $p$  em  $F$  e os 5 pontos que estão a seguir de  $p$  em  $F$ . A idéia é sempre colocar os pontos da reta separadora no conjunto  $E$  da esquerda. Assim não haverá um par de pontos coincidentes sobre a reta com um ponto em  $E$  e outro ponto em  $D$ . Portanto, no máximo 6 pontos podem estar no retângulo  $d \times 2d$ . Onde está a bobagem do esquema proposto pelo professor Sperto?
3. [CLRS 33.4-2] Sem aumentar o consumo assintótico de tempo do algoritmo, mostre como garantir que o conjunto de pontos passados para a primeira chamada recursiva do algoritmo não contenha pontos coincidentes. Prove que então é suficiente que o algoritmo verifique os 6 (e não 7) pontos que seguem cada ponto em  $F$ . Por que não é suficiente verificar somente 5 pontos? Ou é suficiente?
4. Modifique o algoritmo visto em aula para que, no COMBINE, seja calculada a distância entre cada ponto da faixa e apenas pontos do outro lado da partição feita em DIVIDA. Faça a modificação de modo a manter o consumo de tempo do algoritmo em  $O(n \lg n)$ .
5. Mostre que, com a modificação do exercício anterior, no COMBINE, para cada ponto na esquerda, é **suficiente** calcular a distância entre ele e no máximo 4 pontos na direita.
6. Ainda sobre a modificação do exercício 4, você consegue mostrar um exemplo onde é realmente **necessário** calcularmos a distância entre cada ponto e 4 pontos  $d_1, \dots, d_4$  do outro lado da partição? (Ou seja, o seu exemplo deve mostrar um ponto  $e$  da esquerda e pontos  $d_1, \dots, d_4$  da direita na proximidade do ponto  $e$  de tal forma que, se o algoritmo calcula a distância entre  $e$  e apenas cada um dos pontos  $d_1, \dots, d_3$ , então ele não devolve o par mais próximo (que por azar é o par  $\{e, d_4\}$ .) Se você não conseguir encontrar um tal exemplo, então tente mostrar que é **suficiente** o algoritmo calcular a distância entre cada ponto de um lado e menos do que 4 pontos do outro lado.
7. [CLRS 33.4-3] A distância entre dois pontos pode ser definida de diversas maneiras além da Euclidiana. No plano, a  $L_m$ -distância entre dois pontos  $p = (p_x, p_y)$  e  $q = (q_x, q_y)$  é dada por  $(|p_x - q_x|^m + |p_y - q_y|^m)^{1/m}$ . Portanto, a distância Euclidiana é a  $L_2$ -distância. Modifique o algoritmo de divisão-e-conquista para o problema do par mais próximo de tal forma que ele devolva o par de pontos mais próximo em relação à  $L_1$ -distância (também conhecida como *Manhattan distance*).
8. [CLRS 33.4-4] Dados dois pontos  $p$  e  $q$  no plano, a  $L_\infty$ -distância entre eles é dada por  $\max\{|p_x - q_x|, |p_y - q_y|\}$ . Modifique o algoritmo para o par mais próximo para que ele encontre um par mais próximo de acordo com a  $L_\infty$ -distância.

**Observação:** Sempre que descrever uma modificação de um algoritmo dado, exiba o resultado da modificação em pseudo-código.