Geometria Computacional

Departamento de Ciência da Computação – IME/USP

Segundo Semestre de 2007

Lista 2

(1) Ajuste o algoritmo dado nas transparências da aula 2 para que funcione mesmo que a coleção de pontos dada inclua pontos com coordenadas x ou y coincidentes.

(2) Refine o algoritmo e o argumento que garante que ele funciona para que, na rotina Combina, cada ponto da faixa central seja comparado com menos que sete pontos.

[Note que para a ordem de complexidade do algoritmo tanto faz se para cada $m \in M$ calculamos 7, 6 ou 100000 distâncias—o algoritmo continua tendo complexidade de tempo $O(n \log n)$.]

(3) [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, na sua fase Combinar, somente a distância entre cada ponto $m \in M$ e os 5 pontos que estão a seguir de m em M. A idéia é sempre colocar os pontos da reta l no conjunto E da esquerda. Então, não poderá haver um par de pontos coincidentes sobre a reta l com um ponto em E e outro ponto em D. Portanto, no máximo 6 pontos podem estar na retâgulo de dimensões $\delta \times 2\delta$. Onde está a bobagem no esquema proposto pelo professor Sperto?

(4) [CLRS 33.4-2] Sem aumentar a complexidade de tempo assintótica 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 M. Por que não é suficiente verificar somente 5 pontos? Ou é suficiente?

(5) Considere a fase Combinar do algoritmo de divisão-e-conquista para o Problema do Par Mais-Próximo, visto em sala de aula. Modifique a rotina Combina para que seja calculada a distância entre cada ponto e apenas pontos do outro lado da partição feita em Divide. Mostre que, nesta fase, para cada ponto na esquerda, é **suficiente** o algoritmo calcular a distância entre dele com no máximo 6 pontos na direira.

(6) Para a modificação proposta no exercício anterior, você consegue mostrar um exemplo onde é realmente **necessário** calcularmos a distância entre cada ponto e 6 pontos $d_1, \ldots, d_6$ do outro lado da partição? (Ou seja, o seu exemplo deve mostrar um ponto e da esquerda, digamos, e pontos $d_1, \ldots, d_6$ 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 $d_1, \ldots, d_5$ então o algoritmo não devolve o par mais-próximo (que por azar é o par $\{e, d_6\}$.) 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 6 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 a L_1-distância (também conhecida como *Manhattan distance*).

(8) [CLRS 33.4-4] Dados dois pontos p e q no plano, a L_∞-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_∞-distância.

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