MAC 5747 GEOMETRIA COMPUTACIONAL

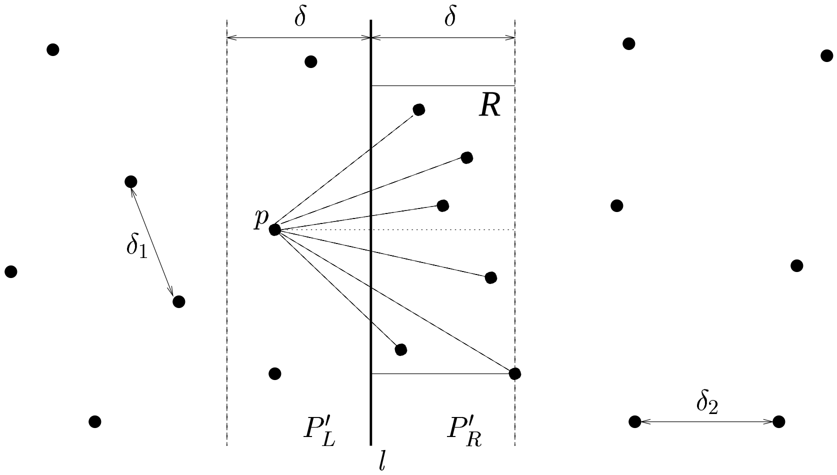
LISTA DE EXERCÍCIOS 1

1. Instruções

Entregue os exercícios 1, 3 e 6 na aula do dia 28 de março, quarta-feira.

2. Exercícios

1. (Suficiente) Considere a fase Combinar do algoritmo de divisão-e-conquista para o Problema do Par Mais-Próximo, visto em sala de aula. Mostre que, nesta fase, para cada ponto $p \in P'_L$ é **suficiente** o algoritmo calcular a distância entre p e no máximo 6 pontos em P'_R.
2. (Necessário) Considere a fase Combinar do algoritmo de divisão-e-conquista para o Problema do Par Mais-Próximo, visto em sala de aula. Nesta fase, para cada ponto $p \in P'_L$ o algoritmo calcula a distância entre p e no máximo 6 pontos em P'_R. Você consegue mostrar um exemplo onde é realmente **necessário** calcularmos a distância entre p e 6 pontos $q_1, \ldots, q_6$ de P'_R? (Ou seja, o seu exemplo deve mostrar um ponto p em P_L e pontos $q_1, \ldots, q_6$ contidos no retângulo R_p de tal forma que se o algoritmo calcular a distância entre p e cada um dos pontos (digamos) $q_1, \ldots, q_5$ então o algoritmo não devolverá como resposta o par mais-próximo (que por azar é o par $\{p, q_6\}$.) Se você não conseguir encontrar um tal exemplo, então tente mostrar que é **suficiente** o algoritmo calcular a distância entre cada ponto de P'_L e menos do que 6 pontos de P'_R.

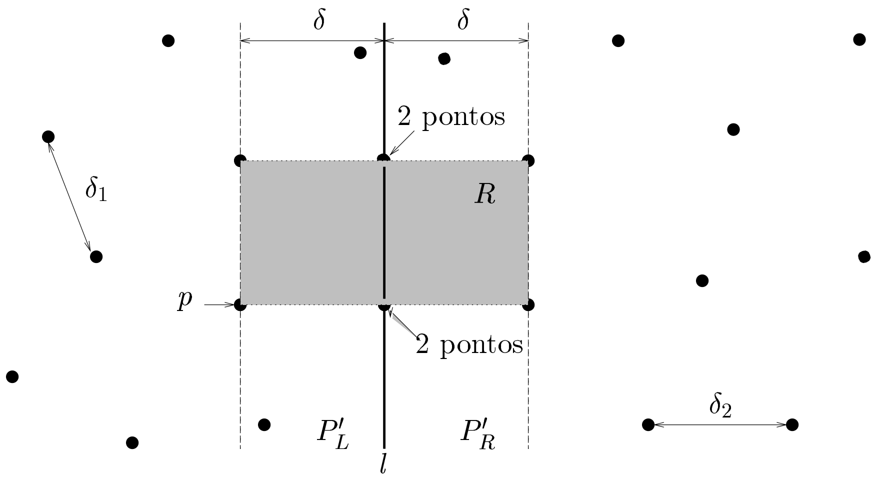


3. Considere agora a seguinte pequena variação da fase Combinar do algoritmo. Seja P^* a lista dos pontos em $P'_L \cup P'_R$ ordenados em relação as suas y-coordenadas (abcissas). Percorra a lista P^* em ordem e para cada

ponto $p \in P^*$ calcule a distância entre p e os 7 pontos que seguem p na lista P^*. (Note que algumas vezes poderemos estar calculando a distância entre pontos de P'_L ou entre pontos de P'_R (i=1,2).) Mostre que com esta variação da fase COMBINAR o algoritmo está correto (i.e. encontra o par mais-próximo).

[Note que com esta variação não precisamos mais ficar testando as y-coordenadas dos pontos que terão as suas distâncias calculadas.

Sugestão: Veja a figura abaixo ou de uma olhada em [1] página 910.]



4. Considere ainda a variação da fase COMBINAR proposta no exercício anterior. Você mostrou que é **suficiente** calcularmos as distâncias entre cada ponto $p \in P^*$ e os 7 pontos que seguem p na lista P^* para que o algoritmo esteja correto. Será que é **necessário** calcularmos essas 7 distâncias? Ou seja, será que não poderiamos calcular a distância entre cada ponto $p \in P^*$ e os (digamos) 6 pontos que seguem p na lista P^* e ainda assim o algoritmo estaria correto? Apresente um exemplo mostrando que é necessário calcularmos essas 7 distâncias ou mostre que é possível encontrarmos o par mais próximo calculando menos do que 7 distâncias para cada $p \in P^*$.

   [Note que para a ordem de complexidade do algoritmo tanto faz se para cada $p \in P^*$ calculamos 7, 6 ou 100000 distâncias—o algoritmo continua tendo complexidade de tempo $O(n \log n)$.]
5. (Exercício 33.4-1 de [1]) Considere ainda a variação da fase COMBINAR proposta no Exercício 3. O professor 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 $p \in P^*$ e os 5 pontos que estão a seguir de p em P^*. A idéia é sempre colocar os pontos da reta l no conjunto P_L. Então, não poderá haver um par de pontos coincidentes sobre a reta l com um ponto em P_L e outro ponto em P_R. 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.

6. (Exercício 33.4-2 de [1]) 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 P^*. É suficiente verificar somente 5 pontos?
7. (Exercício 33.4-3 de [1]) A distância entre dois pontos pode ser definida de diversas maneiras além da Euclidiana. No plano, a L_m-distância entre dois pontos $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$ é dada por $(|x_1 - x_2|^m + |y_1 - y_2|^m)^{1/m}$. 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 próximos em relação a L_1-distância (também conhecida como *Manhattan distance*).
8. (Exercício 33.4-4 de [1]) Dados dois pontos p_1 e p_2 no plano, a L_∞-distância entre eles é dada por $\max\{|x_1 - x_2|, |y_1 - y_2|\}$. Modifique o algoritmo para o par mais próximo para que ele encontre um par mais-próximo de acordo com a L_∞-distância.

Referências

[1] T.H. Cormen, C.E. Leiserson e R.L. Rivest, *Introduction to algorithms*, The MIT Electrical Engineering and Computer Scienece Series, The MIT Press, MacGraw-Hill Book Company, 1990, QA758 C811i.