2011/9/14 A.A #2

Alg 5 (Another linear time algorithm) $O(n)$

MaxEndAt$_1$ = max$(a_1, 0)$

for $m \leftarrow 2, 3, \cdots, n$ do

MaxEndAt$_m \leftarrow \max\{$ MaxEndAt$_{m-1} + a_m$, $0\}$

sol $\leftarrow 0$

for $m \leftarrow 1, 2, \cdots n$ do

if MaxEndAt$_m >$ sol then

sol $\leftarrow$ MaxEndAt$_m$

return sol

Thm) There exists no deterministic algorithm for problem 1 that runs in sublinear time.

pf) [Sketch] Suppose that $\exists$ such algorithm.

then, for some sufficiently large n_0, $\exists$ an input that consists of $[a_1, \cdots, a_{n_0}]$,

for which the algorithm does not access the value of (a_k) for some $1 \leq k \leq n_0$

If every max-sum subsequence contains (a_k)

then the algorithm would produce an incorrect output when $a_k = -\infty$

O/W, it would produce an incorrect output when $a_k = \infty$ □

⊛

$[a_1, a_2, \cdots a_{k-1}, -\infty, a_{k+1}, \cdots, a_{n_0}]$

$[a_1, a_2, \cdots a_{k-1}, \infty, a_{k+1}, \cdots, a_{n_0}]$

Def) Efficient Algorithm.

We say an algorithm is efficient (if) its running is bounded by polynomial in the input size.

2011/9/18 AA #3.

Thm) Master theorem (⇒ for recursive algorithms)

Let $a \geq 1$ and $b > 1$ be constants, $f(n)$ be a nonnegative function, and $T(n)$ be a function defined on the nonnegative integers by the recurrence.

$$\Rightarrow \quad T(n) = aT(n/b) + f(n)$$

① ⇒ (recursive function에 들어가는 argument 의 size !! not fixed)

along with sufficient base cases, where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

① If $f(n) = O(n^{(\log_b a) - \varepsilon})$ for some constant $\varepsilon > 0$,
then $T(n) = \Theta(n^{\log_b a})$

② If $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some constant $k \geq 0$,
then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

③ If $f(n) = \Omega(n^{(\log_b a) + \varepsilon})$ for some constant $\varepsilon > 0$ and there exists some constant $c < 1$ such that $a \cdot f(n/b) \leq cf(n)$ for all sufficiently large n,
then $T(n) = \Theta(f(n))$

Problem 2) Closest Pair Problem (2D)

Given $P = \{P_1, P_2, \cdots, P_n\}$, a set of points, where p_i represents (x_i, y_i) coordinates, find minimum euclidean distance between two different points of P and their indices.

Alg 1

Sort the given set of points in the ascending order of x-coordinates.
Let P_x be the resulting list.
Return CPair(P_x)

function CPair(P_x)

If $|P_x| \leq 3$ then find the closest pair by enumerating all pairs.
Otherwise Let L_x be the 1st $\lfloor \frac{|P_x|}{2} \rfloor$ of P_x
Let R_x be the rest of P_x
$x^* \leftarrow \max_{p_i \in L_x} x_i$
$(P_{l1}, P_{l2}) \longleftarrow$ CPair(L_x)
$(P_{r1}, P_{r2}) \longleftarrow$ CPair(R_x)
find (P_{m1}, P_{m2}) using $(\pm\delta)$ width on x axis.

→ 러닝타임의 order를 개선하지는 X

Let (P_{m1}, P_{m2}) be a closest pair of points where $P_{m1} \in L_x$ and $P_{m2} \in R_x$

Return minimum distance pair among $\{(P_{l1}, P_{l2}), (P_{r1}, P_{r2}), (P_{m1}, P_{m2})\}$

$\delta \leftarrow \min(d(p_{\ell_1}, p_{\ell_2}), d(p_{r_1}, p_{r_2}))$ where $d(a, b)$ is Euclidean distance btw a and b

find (p_{m_1}, p_{m_2}) using $(\pm \delta)$ width on x-axis

Let S be the set of points in P_x whose x-coords are in $(x^* - \delta, x^* + \delta)$

Let S_y be the list of points in S sorted in the ascending order of y-coords

for each point $p \in S_y$ do.

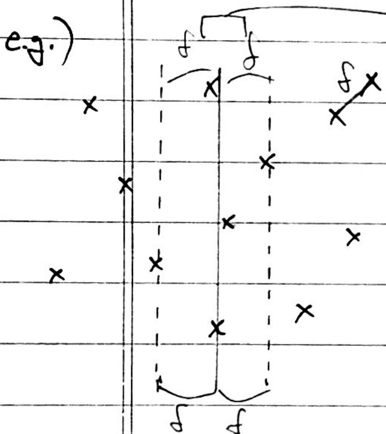
compute the distance between p and next (11 pts) ??? in S_y,

finding a closest pair (p_{m_1}, p_{m_2})

Return best solution.

sorting $O(n \log n)$ (merge sort)

e.g.)



• # of pt ⇓

worst case 이라는 줄지 않는다.

Running time analysis of Alg 1)

Let $T(n)$ denote the worst case # of basic operations performed by Alg 1 when the input has n points. For some constant $c_1, c_2 > 0$ we have

$$T(n) \le c_1, \text{ if } n \le 3 \quad \text{: base case.}$$

$$T(n) \le \underbrace{2T(n/2)}_{\text{recursive part}} + \underbrace{c_2 n \log n}_{\text{find } d(p_{m_1}, p_{m_2})} \quad \text{O/W}$$

T(n)은 inequality 이므로 직접 master theorem 적용할 수 없음.

Where $n/2$, again, is to be interpreted to mean either $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$.

It is easy to see that $T(n)$ is bounded by $T'(n)$ satisfying $T'(n) = 2T'(n/2) + f(n)$

where $f(n) = \Theta(n \log n)$

From the Master theorem, since $f(n) = O(n^{\log_2 2} \log^1 n) = O(n \log n)$

$\Rightarrow k = 1$, case 2,

$T'(n) = \Theta(n \log^2 n)$ since the preprocessing takes $O(n \log n)$ time,

the overall running time is $O(n \log^2 n)$

Alg 2

Sort the given set of points in the ascending order of x-coordinates.) $O(n \log_2 n)$
Let P_x be the resulting list.
Sort the given set of points in the ascending order of y-coordinates. $O(n \log_2 n)$
Let P_y be the resulting list.
Return CPair(P_x, P_y)

function CPair(P_x, P_y)
if $|P_x| \leq 3$ then find the closest pair by exhaustively all pairs and return it.
O/W
Let L_x be the 1st $\lfloor \frac{|P_x|}{2} \rfloor$ of P_x.
Let R_x be the rest of P_x.
Let L_y be the list of points in L_x sorted by y-coordinates. by examining P_y — linear time
Let R_y be the list of points in R_x sorted by y-coordinates. by examining P_y — linear time
$x^* \leftarrow \text{Max}_{p_i \in L_x} x_i$
$(p_{L_1}, p_{L_2}) \leftarrow$ CPair(L_x, L_y) — already sorted by y
$(p_{R_1}, p_{R_2}) \leftarrow$ CPair(R_x, R_y)
$\delta \leftarrow \text{Min}(d(p_{L_1}, p_{L_2}), d(p_{R_1}, p_{R_2}))$
Let S be the set of points in P_x whose x-coords are in $(x^* - \delta, x^* + \delta)$ — by P_x, linear time
Let S_y be the set of points in S, sorted in the ascending order of y-coords.
for each point $p \in S_y$ do
compute the distance btw p and next 11 pts in S_y
finding a closest pair (p_{M_1}, p_{M_2})
Return the minimum distance pair $\leq$ (p_{L_1}, p_{L_2}), (p_{R_1}, p_{R_2}), (p_{M_1}, p_{M_2}) if exists.

RT Analysis of Alg 2)

$T(n) \leq c_1$ if $n \leq 3$
$T(n) \leq 2T(n/2) + c_2 n$ ← linear time.

$\Rightarrow T(n) = \Theta(n \log n)$ ... $T(n) = O(n \log n)$ — subquadratic time.