Brenden Nelson
ID#11635716

350 Hw 4

(1) badClosestPair Alg.

1^{st} - split // get A1 + A2. $O(n)$

2^{nd} - badClosestPair on A1 // gets δ_1. $T_{avg}(\frac{n}{2})$

- badClosestPair on A2 // gets δ_2. $T_{avg}(\frac{n}{2})$

3^{rd} - pick smallest δ among points between A1 + A2. // gets δ. $O\left((2\cdot\frac{n}{2})(2\cdot\frac{n}{2})\right)$

$3^{rd} = O(n\cdot n) = O(n^2)$

Step (1) Formula: $T_{Avg}(n) = \overbrace{2\cdot T_{Avg}\left(\frac{n}{2}\right)}^{\text{step } 2^{nd}} + \overbrace{O(n^2)}^{\text{step } 3^{rd}} + \overbrace{O(n)}^{\text{step } 1^{st}}$

$= \frac{2}{n}\sum_{i=1}^{n}\left(2T_{Avg}\left(\frac{n}{2}\right) + O(n^2) + O(n)\right)$

Step (2) guess: $T_{avg}(n) = O(n^2)$. That is, $\exists c > 0$, such that $T_{avg}(n) \le c\cdot n^2$ for all n.

$\boxed{\text{I.H.: } \forall i < n,\ T_{avg}(i) \le c\cdot i^2}$

Step (3) check:

$$T_{Avg}(n) = \frac{1}{n}\sum_{i=1}^{n} 2\cdot T_{Avg}\left(\frac{n}{2}\right) + O(n^2) + O(n)$$
$$= \frac{1}{n}\left\{\sum_{i=1}^{n} 2\cdot T_{Avg}\left(\frac{n}{2}\right) + \sum_{i=1}^{n} O(n^2) + \sum_{i=1}^{n} O(n)\right\}$$
$$= \frac{1}{n}\left\{\sum_{i=1}^{n} 2\cdot T_{Avg}\left(\frac{n}{2}\right) + a\cdot n^3 + a\cdot n^2\right\}$$
$$= \frac{2}{n}\sum_{i=1}^{n} T_{Avg}\left(\frac{n}{2}\right) + a\cdot n^2 + a\cdot n$$
$$\le \frac{2}{n}\sum_{i=1}^{n} c\cdot\left(\frac{n}{2}\right) + a\cdot n^2 + a\cdot n$$
$$\le \frac{2c}{n}\sum_{i=1}^{n} \cdot\left(\frac{n}{2}\right) + a\cdot n^2 + a\cdot n$$
$$\le \frac{2c}{n}\int_0^n f(x)\,dx + a\cdot n^2 + a\cdot n$$
$$\le \frac{2c}{n}\int_0^n \frac{x}{2}\,dx + a\cdot n^2 + a\cdot n$$
$$= \frac{2c}{n}\left\{\frac{x^2}{4}\Big|_0^n\right\} + a\cdot n^2 + a\cdot n$$
$$= \frac{2c}{n}\left\{\frac{n^2}{4}\right\} + a\cdot n^2 + a\cdot n$$
$$= \frac{cn}{2} + a\cdot n^2 + a\cdot n$$
$$\le c\cdot n^2 \text{ when } c >> a.$$

naive Karatsuba Alg

② $\beta = \alpha_1$ // $O(1)$

for ($i = 2$ to n) // $O(n)$

$\beta = \beta \cdot \alpha_i$ (using Karatsuba) // $\times O(n^{1.59})$ } $= O(n^{2.59})$

return β.

① $T_w(n) = \max_{1 \le r \le n} \{ O(n^{2.59}) + \cancel{O(1)} \}$

$= \max_{1 \le r \le n} \{ \underbrace{a \cdot n^{2.59}}_{F(r)} \} \cancel{\pm a}$

$F'(r) = an^{2.59}$.

$= \max \{ F(1), F(n) \} = \max \{ a \cdot n^{2.59} \} = a \cdot n^{2.59} \le a \cdot n^3$

$= O(n^3)$

③ $T(n,m)$: multiply m strings of length n.

Clearly, the worst case is $T(n,n)$ which is bounded by the summation of the first group, 2nd group, + results from both groups.

$T(n,m) = 2 \cdot T(n, \frac{n}{2}) + O(n^{1.59})$, $G(n) = 2 \cdot G(\frac{n}{2}) + G(n \cdot n)^{1.59}$

Guess: $G(n) = O((n^2)^{1.59})$

I.H: $\forall i < n$, $T_w(i) \le C \cdot (i^2)^{1.59}$

Check:

$= 2 \cdot C((\frac{n}{2})^2)^{1.59} + G(\frac{n}{2} \cdot n)^{1.59} \le C \cdot (n^2)^{1.59}$

Clearly, the algorithm would have a worst case complexity of $O((n^2)^{1.59})$ because the n length n amount of numbers would take $O(n^2)$ to compute and for each number, we would have to perform better Karatsuba which would cause the n^2 to be performed $n^{1.59}$ times. Bringing the total complexity of worst case for better Karatsuba to $O((n^2)^{1.59})$

(4) The first step is to think of all airplanes currently in the air as occupying space. Each airplane is at least 1 unit apart from another. This means that there can only be a finite number of airplanes located inside a 1x1x1 unit space. Knowing this information, I can cut the airspace in half so that roughly half the airplanes are in box 1 and the other half are in box 2. Next, I can look at box 1 to find the closest pair of airplanes for box 1 (δ_1), and do the same for box 2 to get (δ_2). After I have $\delta_1 + \delta_2$, I can look at only airplanes located within $\min\{\delta_1, \delta_2\}$ from the line I ~~originally~~ cut. Because there would now be a finite number of airplanes located in this box between box 1 and box 2 all less than $\min\{\delta_1, \delta_2\}$ distance from the middle, I could enumerate every possible pair and compare to find a global δ_3. Once I have $\delta_1, \delta_2, \delta_3$, I can look at $\min\{\delta_1, \delta_2, \delta_3\}$ to find the total closest pair of airplanes. This algorithm is a variation of the closest-pair algorithm which could find the closest pair of airplanes in linear time.

(5) n strings on $\{a,b\}$ each with length $= m$.
$\delta(\alpha,\beta)$ is distance between two points. Each string is a point of A's + B's.

ex: abbabab => (3,4)

now, n-strings are translated into n-points.
run classic close-pair algorithm to get closest different pair.
Trouble = $\Theta(n \log n)$
Want = $O(n \cdot m)$
When [illegible] is higher than [illegible], that means that the length of the strings is much smaller than the total number of strings.
This is a problem because, we want the algorithm to run in time $O(n \cdot m)$. ~~If~~ If this is the case, we can check the length of the strings to see if $m \geq n \log n$. If m is less than $n \log n$, then we could take m^2 as the size of the strings rather than m. This would ensure that [illegible] $= O($[illegible]$)$ or, the algorithm would be upper bound by $O(m \cdot n)$