Cpt S 450 Homework #4

Please print your name!

No late homework!

1. (very hard) Let A be a set of n points in the two-dimension plane. We would like to find the distance of a closest pair in A, using the following badClosestpair algorithm. We first randomly split A into two subsets A_1 and A_2, in linear time. Then, we run badClosestpair on A_1 and on A_2. As a result, we obtain the distance δ_1 of a closest pair in A_1 and the distance δ_2 of a closest pair in A_2. Finally, for each point in A_1 and each point in A_2, we calculate the distance between the two points and pick the smallest δ among all these distances. The result of badClosestpair running over A returns as $\min(\delta_1, \delta_2, \delta)$. Compute the average-case complexity of badClosestpair.

2. We know that, using Karatsuba algorithm, it takes worst time complexity $O(n^{1.59})$ to multiply two bit strings with length n. Suppose that I want to multiply n bit strings $\alpha_1, \cdots, \alpha_n$, each of which is with length n. I do this by the following naiveKaratsuba algorithm:

$\beta = \alpha_1$;
For $i = 2$ to n
  $\beta = \beta \cdot \alpha_i$ (using Karatsuba);
return β.

Compute the worst-case complexity of naiveKaratsuba.

3. (Be very careful – this problem involve a lot of math) We know that, using Karatsuba algorithm, it takes worst time complexity $O(n^{1.59})$ to multiply two bit strings with length n. Suppose that I want to multiply n bit strings $\alpha_1, \cdots, \alpha_n$, each of which is with length n. I do this by the following betterKaratsuba algorithm. I first divide the n bit strings into two groups, each of which has roughly $\frac{n}{2}$ bit strings. Then run betterKaratsuba on each group (when the group has only one bit string, betterKaratsuba simply returns the bit string). Then, we use Karatsuba on the results of betterKaratsuba for the two groups. Compute the worst-case complexity of betterKaratsuba.

4. At some moment, there are n^3 airplanes in the air and we know the x-, y-, and z- coordinates for each airplane. It is also known that all the airplanes are contained in a cube of size n and the distance between any two airplanes is at least 1. Design an algorithm that runs in time $O(n^3)$ and finds the closest pair of airplanes (the pair has the shortest distance between all the n^3 airplanes. Notice that it is essentially a linear time algorithm since the input size is n^3).

5. Let $\Sigma = \{a, b\}$. For a string α over the alphabet Σ, we use $\#_a(\alpha)$ (resp. $\#_b(\alpha)$) to denote the number of a's (resp. b's) in α. Given a pair of two strings α, β, we use $d(\alpha, \beta)$ to denote the *difference* of the pair, which is defined as

$$|\#_a(\alpha) - \#_a(\beta)|^2 + |\#_b(\alpha) - \#_b(\beta)|^2.$$

Now, we are given an array A of n strings over Σ; the length of each string in A is bounded by m. A distinct pair in A is defined by $(A[i], A[j])$ with $i \neq j$. Design an algorithm that runs in time $O(nm)$ and that finds the smallest difference of all the distinct pairs in A.