cpt_s 350

Homework 4

11641327 Yu-Chieh Wang

2020/3/8

1. Divide and Conquer.

   Step 1: Split A to A_0 and A_1. $O(n)$

   Step 2: get the smallest distance δ_0 from A_0. $T(\frac{n}{2}-1)$

   Step 3: get the smallest distance δ_1 from A_1. $T(n-\frac{n}{2})$

   Step 4: get the smallest distance δ from A. $T(n)$

   Therefore, we get the formula as follow:

   $$T_{AVG} = O(n) + T(\frac{n}{2}) + T(\frac{n}{2}) + T(n)$$

   Then, we guess the time complexity is $O(n^2)$ and check.

   $$T_{AVG} = O(n) + T(\frac{n}{2}) + T(\frac{n}{2}) + T(n)$$

   $$= O(n) + 2 \cdot O(\frac{n}{2})^2 + O(n^2)$$

   $$\leq a \cdot n + 2 \cdot a \cdot (\frac{n}{2})^2 + a \cdot n^2$$

   $\leq c \cdot n^2$ when c>>a.

2. Compute the worst-case complexity of naiveKaratsuba.

   Since the time complexity of Karatsuba is $O(n^{log_2 3})$, we get the following formula:

$T(2) = T(1)^{log_2 3}$ (multiply 2 bit strings)

$\vdots$

$T(n-1) = T(n-2)^{log_2 3}$

$T(n) = T(n-1)^{log_2 3}$ (multiply n bit strings)

As a result, to get the $T(n)$, it will do recursive n times, so the time complexity is $n^{(log_2 3)^n} = n^{(1.59)^n}$.

3. Compute the worst-case complexity of betterKaratsuba.

Step 1: divide every n bit string as two $\frac{n}{2}$ bit strings. $O(n)$

Step 2: run betterKaratsuba on each strings. $2 \cdot T(\frac{n}{2})$

Step 3: run Karatsuba on the results of betterKaratsuba. $O(n^{log_2 3})$

Therefore, we get the time complexity as follow:

$$T_{WORST} = O(n) + 2 \cdot T(\frac{n}{2}) \cdot O(n)$$

$$= a \cdot n + 2 \cdot a \cdot (\frac{n}{2})^{log_2 3} \cdot a \cdot n^{log_2 3}$$

$$\leq O(n^3)$$

4. Design an algorithm that runs in time $O(n^3)$ and finds the closest pair of airplanes.

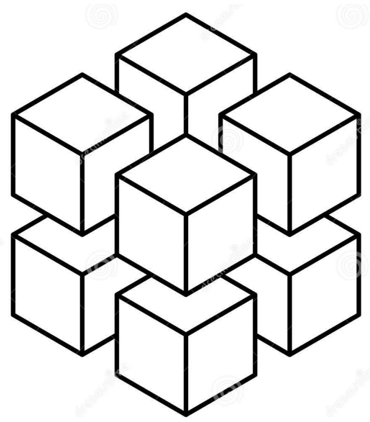
(Using divide and conquer algorithm)

Step 1: See the sky as a huge cube.

Step 2: Keep dividing the cube in half until the cube size is 1

(do it recursively)

Step 3: Each cube is able to contain two airplanes. Therefore,



check each cube. If there are two airplanes in a cube, check their distance. If not, combine the nearby cube and check again.

Step 4: Finally, when all smeller cubes combine as a huge cube, all distances between airplanes are checked.

5. Design an algorithm that runs in time $O(nm)$ and that finds the smallest difference of all the distinct pairs in A.

   Given 1: a 2-D array A of n strings which only contain alphabet {a, b}.

(length of A[i] $\leq$m)

A = { “abbabbbaaabbbba”	(A[0])
“bbbbbbbbbbaaaaa”	(A[1])
:	:
“aaaaabbbbb” }	(A[n])

Given 2: a formula to denote the difference between every two strings.

$$d(\alpha, \beta) = |\#_a(\alpha) - \#_a(\beta)|^2 + |\#_b(\alpha) - \#_b(\alpha)|^2$$

Since we know how to denote the difference by the given formula, we can run a for loop to compute the difference between every two strings. The pseudo-code is as follow:

$d(\alpha, \beta)${

Count how many ‘a’ in A[x]; // $\#_a(\alpha)$

Count how many ‘b’ in A[x]; // $\#_b(\alpha)$

Count how many ‘a’ in A[x+1]; // $\#_a(\beta)$

Count how many ‘b’ in A[x+1]; // $\#_b(\beta)$

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

```
}
main(){
    D = d(1,2)
    For (i=0; i<n; i++){
        For (j = i+1; j≤n; j++){
            If (d(i, j) < D){
                D = d(i, j)
            }
        }
    }
    Return D
}
```