Jinyang Ruan

011696096

CPT S 515 Advanced Algorithms HW #1

1) $Input\ size\ =\ n,$

   $Time\ =\ O(1)$.

2) 1. Cut the array into $n/7$ groups,

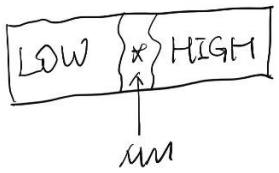
   2. Sort each group (takes $O(n)$ time),

   3. After sorted, we have $n/7$ medians,

   4. Recursively, we use linear time selection to select $n/14 - th$ smallest median (called MM) from the $n/7$ medians,

   5. Swap MM with the first element in the original array of n numbers,

   6. Run partition on updated array,



   7. If $i == r$, return this MM as the $i - th$ smallest,

   If $i < r$, recursively run linear time selection to select the $i - th$ smallest from the LOW,

   If $i > r$, recursively run linear time selection to select the $i - th$ smallest from the HIGH.

   Following the above steps, we have,

$$Tw(n) = Tw(10n/14) + Tw(n/7) + O(n)$$

   Guess

$$Tw(n) = O(n) \leq c \cdot n$$

Check

$$Tw(n) = Tw(10n/14) + Tw(n/7) + O(n)$$

$$Tw(n) \leq c \cdot 5n/7 + c \cdot n/7 + a \cdot n$$

$$Tw(n) = 6/7 \cdot c \cdot n + a \cdot n$$

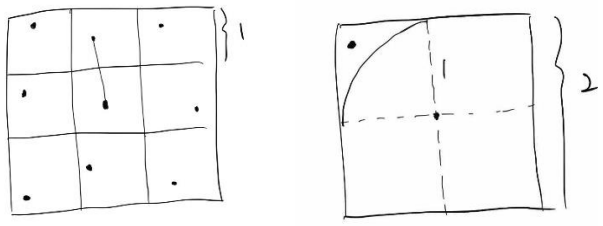
$$Tw(n) \leq c \cdot n \quad when\ c >> a$$

So, we can say when the group size is 7, the algorithm still runs in linear time.

3) 1. Divide the whole graph into $n * n$ squares, there are 2 situations:

a. bug in the square

b. bug on the square



2. Because every 2 bugs must stay away by at least 1, there are maximal 1 bug in 1 square (includes on the sides). in the case a, we only need to compute the length between bug in the square and bugs in the adjacent squares, which are maximal 8, then choose the closest pair as δ.

In the case b, we only need to compute the length between bug on the square and bugs in the adjacent squares which are maximal 4, then choose the closest pair as δ.

3. Recursively doing step 2 on each bug, update the closest pair δ when found a smaller pair.

In the worst case, every bug is in the square, we need to compare 8 pairs for each bug, time complexity is $O(n)$, since the input size is n^2,

$$Tw(n^2) = O(n^2)$$

4) Given: two C programs P_1 and P_2.

Question: decide the similarity between P_1 and P_2.

Analysis:

Programs have two factors which are function and the process of implementation. So, the similarity between two programs should be discussed at function level and the implementation level. Also, similarity should be discussed within a given range of inputs (the range could be anything).

1.The function of program also means the output of program. For one program, different inputs may produce different outputs. Between two programs, same input may produce different outputs, and different inputs may produce the same output, so it is extremely difficult to compare each pair of the input and output to determine the similarity.

So, I would like to translate each program into a mathematical function $f(x_n, y_n)$, each point on the function is a pair of input x_n and output y_n. If I want to decide the similarity in the specific input and output between two programs, I just need to find out the points on each function, and compute the distance between two points, we can consider that the shorter distance means the higher similarity. If I want to decide the similarity within a range of the inputs or between the whole two programs, I can subtract these two functions to get a new function, and integrate the new function (roughly) over the target range, and we can say that the smaller the area (number) that the integral gives, the more similar the two programs are.

2. If the programs are Whitebox, which means I know the source code. It is possible for us to discuss at implementation level. I can further determine the similarity of the program by comparing the syntax and semantics of the code.

Define:

Input of P_1 is x_1, input of P_2 is x_2, output of P_1 is y_1, output of P_2 is y_2.

Combine each input x to its corresponding output y as a pair of number (x_n, y_n).

Function $f(1)$ is translated by P_1, function $f(2)$ is translated by P_2.

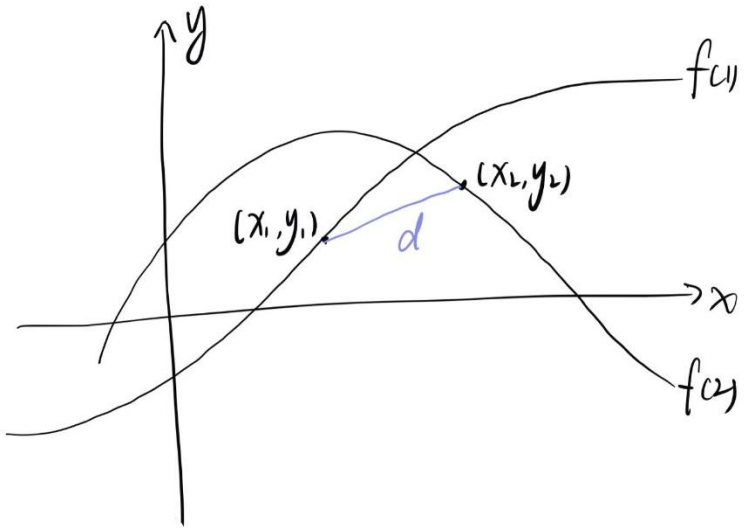
Assumptions:

In two programs, for any input x, there is a unique output y,

Functions are not discrete.

Proof:

1. Decide the similarity on specific inputs and outputs:



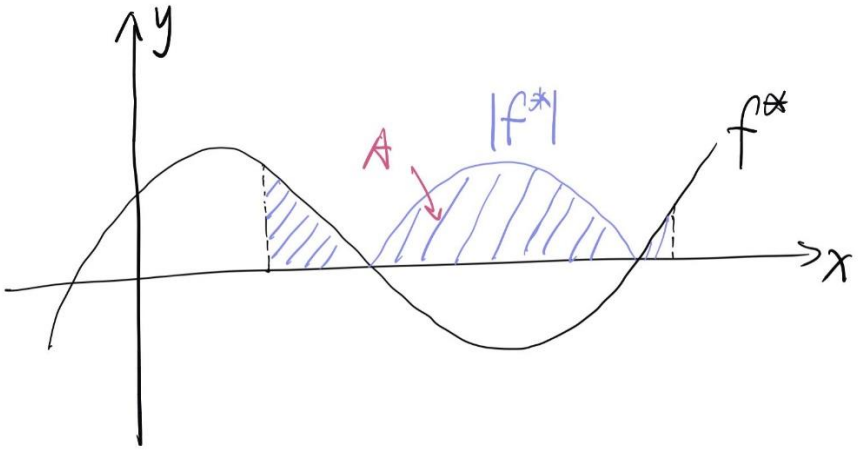
Distance:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The smaller the $d\ is$, the higher similarity is.

2. Decide the similarity within given range/whole two program:

$$f *= f(1) - f(2)$$



Then compute area:

$$A = \int |f^*|$$

The smaller the A is, the higher similarity is.

So far, I decided the similarity between two programs indirectly.

If I know the source code of two programs, I will continue determine the similarity by comparing syntax and semantics of the code.

For example:

"$a = 1$" and "$a = 1$" is same,

"$i = 0; while(i < 100)\{i++\}$" and "$for(int\ i = 0; i < 100; i++)$" looks different, but actually the same.