CAB301 Practical 2

Time: 13:00 Monday

**Question 1**: Definitions of big- notation:

a) Time grows in proportion to input:

b) Time does not change with input, i.e., consistent or constant where is a number independent of . Hence, complexity is .

c) . Since the highest power is , the complexity is of class .

**Question 2**: Efficiency function of basic control structures

For all parts, the parameter characterising the input size is the number , which is just a whole number.

The basic operation is the assignment .

Part a) Single for loop:

**for** **to** **do**

Set to be the time taken to run .

The efficiency function is:

This is added to itself for every number from to .

There are (upper – lower + 1) numbers between and , so is added to itself times.

Therefore, the efficiency function is:

Which belongs to complexity class . (in LaTeX, \mathcal{O})

Part c) Single for loop and a nested for loop, branched by an if statement

**if**   
 **for** **to** **do**

**else**

**for** **to** **do**

**for** **to** **do**

Find the complexity of each for loops (we’ll deal with the if statement later).

For the first for loop:

For the second for loop:

Factor in the probability of these cases. Assume that , where is the probability of happening (i.e., the first for loop is used).

The efficiency function is:

When never happens:

Think of a scenario where you are buying lottery tickes.

One ticket is worth return).

If you win, you get ( return).

Let’s assume the probability of winning is 50%, we can take it easily.

Find the average value of each ticket.

If probability of winning , then on average for every 2 tickets, 1 wins.

Total value of 2 tickets:

Average value per ticket:

per ticket

If the probability of winning is 10%, less likely, but maybe.

On average, 10 tickets 1 wins, 9 lose

Per ticket:

Here, is the probability of winning.

If the probability of winning is , no.

This is about, for every 11 tickets, 1 win 10 losses. The total is now:

d) Single for loop, followed by a nested for loop:

**Question 3**: Algorithm Analysis. Only need big- complexity.

Part b) Algorithm to find minimum distance between any two elements.

**ALGORITHM** MinDistance()

**for** **to** **do**  
 **for** **to** **do**

**return**

The parameter characterising the input is the number of elements in the array , called .

The basic operation is the assignment , which we assume a running time of .

Set up the summation:

When there are additions or subtractions in the sum, can split them into two parts:

* Depends on
* Independent of

Note that since we are adding a different thing every time, we cannot multiply by , as the value in the summation depends on the running .

Solve the following:

Add the above together:

Divide both sides by 2:

If familiar with the arithmetic series, find that:

Put everything back together, and find the worst case efficiency function:

We observer that the highest power multiply is , therefore, the time complexity of this algorithm is .

Alternative approach:

**Question 5:**

|  |  |
| --- | --- |
| **Size** | **Count** |
| 1000 | 252939 |
| 2000 | 1013126 |
| 3000 | 2265761 |
| 4000 | 3983470 |
| 5000 | 6234042 |
| 6000 | 8998359 |
| 7000 | 12387202 |
| 8000 | 16020750 |
| 9000 | 20373482 |
| 10000 | 24980992 |
| 11000 | 30314370 |
| 12000 | 35529968 |
| 13000 | 42508146 |
| 14000 | 48723778 |
| 15000 | 56416624 |
| 16000 | 64528928 |
| 17000 | 72418258 |
| 18000 | 80408154 |
| 19000 | 89808813 |
| 20000 | 99785925 |

Start by forming a hypothesis (assumption of what the complexity could be).

Try finding the ratio:

If , then when input doubles, algorithm takes twice as long

If , then when input doubles, algorithm takes four times as long:

If , then when input doubles, algorithm takes eight times as long:

Try a couple of :

* :
* :

Since , we can assume that the algorithm is of complexity class .

To make prediction for any , need to find :

Therefore,

And for 25,000 elements:

operations.