HW 02 - Complexity II

Sebastián González Villacorta

August 30, 2022

1 Compute the number of elementary steps required to run the following code. Take the assignment of a value to a variable (e.g., A[1,2,4]=1) as elementary step.

```
for (k=1; k¡=n; k++)
 for (j=1; j¡=k; j++)
  for (i=1; i¡=j; i++)
    A[i,j,k]=1;
```

We start by defining an arbitrary value of n and manually computing each cycle to understand the behaviour of the embedded for loops.

$k = 1$
$j = 1$
$i = 1$
$A[1, 1, 1] = 1$

The number of elementary steps (assignments of variable) for the cycle where $k = 1$ is 1, let's continue with this process to see if we can spot a pattern.

$k = 2$	$k = 2$	$k = 2$
$j = 1$	$j = 2$	$j = 2$
$i = 1$	$i = 1$	$i = 2$
$A[1, 1, 2] = 1$	$A[1, 2, 2] = 1$	$A[2, 2, 2] = 1$

When $k = 2$ the number of elementary steps is 3...

$k = 3$	$k = 3$	$k = 3$	$k = 3$	$k = 3$	$k = 3$
$j = 1$	$j = 2$	$j = 2$	$j = 3$	$j = 3$	$j = 3$
$i = 1$	$i = 1$	$i = 2$	$i = 1$	$i = 2$	$i = 3$
$A[1, 1, 3] = 1$	$A[1, 2, 3] = 1$	$A[2, 2, 3] = 1$	$A[1, 3, 3] = 1$	$A[2, 3, 3] = 1$	$A[3, 3, 3] = 1$

Finally we can see that the elementary steps taken when $k = 3$ is 6. From this information we can conclude that the steps for each cycle are as follows.

$$1 = 1$$

$$1 + 2 = 3$$

$$1 + 2 + 3 = 6$$

By analysing this pattern we can clearly see that the number of iterations is going to be equal to n and so is the number of sums to be made in each iteration.

$$n \begin{cases} 1 \\ 1+2 \\ 1+2+3 \\ 1+2+3+4 \\ 1+2+3+4+5 \\ \text{...........................}n \end{cases}$$

We can express this mathematically as the sum of the sums from $i = 0$ to n as follows.

$$\sum_{i=1}^{n} \sum_{i=1}^{n} i$$

By applying previous knowledge we can express this sum as:

$$= \sum_{i=1}^{n} \frac{i(i+1)}{2}$$

$$= \sum_{i=1}^{n} \frac{i^2 + i}{2}$$

$$= \frac{1}{2} \sum_{i=1}^{n} i^2 + i$$

Because of the properties of the sum this expression can be separated into two different sums. $\sum_{i=1}^{n}(a \pm b) = \sum_{i=1}^{n} a \pm \sum_{i=1}^{n} b$

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

Since we already know how to express both of this sums we can rewrite them as:

$$= \frac{1}{2} \left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} \right)$$

And by doing some magic (algebra)...

$$= \frac{1}{2} \left(\frac{n(n+1)(2n+1) + 3n(n+1)}{6} \right)$$

$$= \frac{1}{2}\left(\frac{2n^3 + n^2 + 2n^2 + n + 3n^2 + 3n}{6}\right)$$

$$= \frac{1}{2}\left(\frac{2n^3 + 6n^2 + 4n}{6}\right)$$

And finally we are left with the expression that describes the number of elementary steps.

$$= \left(\frac{n^3 + 3n^2 + 2n}{6}\right)$$

2 Suppose that algorithms A and B solve the same problem and they require consuming $n^3 + 2n + 5$ and $10n^2 + 2n + 5$ time units, respectively. Please compute the values of n for which algorithm B takes longer to run than algorithm A. Furthermore, comment on when it would be pertinent to use algorithms A or B.

We can solve this as an inequality where.

$$10n^2 + 2n + 5 > n^3 + 2n + 5$$

$$10n^2 > n^3$$

$$10n^2 - n^3 > 0$$

$$n^2(10 - n) > 0$$

And thus the solution would be the set of values:

$$n\epsilon[-\infty, 0) \cup (0, 10)$$

But since n is the size of the input and it doesn't make sense that a size is negative we are left with the following result.

$$n\epsilon(0, 10)$$

Algorithm A should be used only in the cases where the input size is smaller than 10, otherwise Algorithm B is a more efficient option.

3 For each algorithm on Table 1, the corresponding function (second column from left to right) quantifies the amount of time, in hundredths of a second, that such algorithm takes to run, for a given value of n Compute the largest value of n that corresponds to running each algorithm for each value of $n\epsilon\{1, 2, 3, ...\}$ so that the corresponding accumulated running time is one full second.

Algorithm A	$f_a(n) = \log_2 n$
Algorithm B	$f_a(n) = n^3 + 5$
Algorithm C	$f_a(n) = 2^n$
Algorithm D	$f_a(n) = \sin(2\pi n)$

3.1 Algorithm A

$$\sum_{i=1}^{n} \log_2 n = 100$$

This can sum be expressed as:

$$\log_2(1) + \log_2(2) + ... + \log_2(n-1) + \log_2(n)$$

By the properties of logarithms we know that $\log(a) + \log(b) = \log(ab)$ and thus...

$$\log_2(1 \times 2 \times ... \times (n-1) \times n) = \log_2(n!)$$

Since we are looking for the largest value of n so that the running time is equal to one second

$$\log_2(n!) = 100$$

$$n! = 2^{100}$$

Doing a numerical analysis we get the following result.

n	$n!$
1	1
2	2
...	...
28	$3.04888E + 29$
29	$8.84176E + 30$

Since the we are looking for the value closest to $2^{100} = 1.26765E + 30$ and considering a flooring of the result we can conclude that the largest value of n is 28.

3.2 Algorithm B

$$\sum_{i=1}^{n} i^3 + 5 = 100$$

We can separate this into two different sums

$$\sum_{i=1}^{n} i^3 + \sum_{i=1}^{n} 5$$

$$= \left[\frac{n(n+1)}{2}\right]^2 + 5n$$

$$= \frac{n^2(n+1)^2}{4} + 5n$$

$$= \frac{n^2(n^2+2n+1)}{4} + 5n$$

$$= \frac{n^4 + 2n^3 + n^2 + 20n}{4} = 100$$
$$= n^4 + 2n^3 + n^2 + 20n = 400$$

By doing a numerical analysis...

$$= (3.78)^4 + 2(3.78)^3 + (3.78)^2 + 20(3.78) = 402.06$$

If we floor the value we get that the largest number of n is 3

3.3 Algorithm C

$$\sum_{i=1}^{n} 2^i$$

We can rename this sum as Sn.

$$\sum_{i=1}^{n} 2^i = 2^1 + 2^2 + ... + 2^{n-1} + 2^n = Sn$$

By multiplying this sum by two we get the following:

$$2Sn = 2^2 + 2^3 + ... + 2^n + 2^{n+1}$$

We substract this two sums

$$Sn - 2Sn =$$
$$2^1 + 2^2 + ... + 2^{n-1} + 2^n$$
$$-2^2 - 2^3 - ... - 2^n - 2^{n+1}$$
$$Sn - 2Sn = 2^1 - 2^{n+1}$$
$$Sn(1-2) = 2 - 2^{n+1}$$
$$Sn(-1) = 2 - 2^{n+1}$$
$$Sn = 2^{n+1} - 2$$

We test the following expression with a couple of known values

$$n = 1$$
$$Sn = 2^{1+1} - 2 = 2$$
$$n = 2$$
$$Sn = 2^{2+1} - 2 = 6$$
$$n = 3$$
$$Sn = 2^{3+1} - 2 = 14$$

With this expression we can get the final result

$$2^{n+1} - 2 = 100$$

$$2^{n+1} = 98$$

$$2 \times 2^n = 98$$

$$2^n = 49$$

$$n = \log_2 49 = 5.614$$

Flooring this result we can conclude that the largest value of n is 5

3.4 Algorithm D

$$\sum_{i=1}^{n} \sin(2\pi n)$$

$$= \sin(2\pi(1)) + \sin(2\pi(2)) + ... + \sin(2\pi(n-1)) + \sin(2\pi(n)) = 0$$

By analysing the cyclical behaviour of the function sin we can see that $\sin(2\pi n)$ will always be equal to zero with every integer value of n. And since we defined n as $n\epsilon\{1, 2, 3, ..\}$ this will be true for any value of n.