Data Structures

Homework Assignment 1

Lusine Tangyan

Problem 1 / Consider the code samples below:

i. a) The code iterates over an array, checks whether the element at that index is negative, if yes it multiplies it by -1, making it positive. In general, it flips the signs of negative numbers to positive.

b) row 1: $\text{for}(\text{int } \overbrace{i=0}^{1};\ \overbrace{i < a.\text{length}}^{a.\text{length}+1};\ \overbrace{i{+}{+}}^{a.\text{length}})$

$i = 0$ - one ~~operation~~ prim. op.

$i < a.\text{length}$ - 2 prim. op.

$i{+}{+}$ - 1 prim op.

Let's denote a.length by $\underline{n}$.

$1 \cdot 1 + (n+1) \cdot 2 + \overbrace{\cancel{(n+1)+1n}}^{n}\ \cancel{3n+3} = 3n+3$

row 2: $\text{if } (\underbrace{a[i] < 0}_{n})$

$a[i] < 0$ - 2 prim. op.

$\underline{2n}$.

row 3: $a[i] *= -1$

In worst case, this line is performed $\underline{n}$ times.

$a[i] *= -1$ - 3 prim. op.

$\underline{3n}$.

Overall, $3n + 3 + 2n + 3n = 8n + 3 = \underline{O(n)}$

11. a) The code iterates over the array a, takes the element from the given index and divides it by 2 ~~while~~ until that element is ≤ 0. Every time it divides the element by 2, it also multiplies the element in the corresponding index in b by 2. So, in general, it does not change the array a because it stores the ~~elements/the/a~~ value of element at every index in variable cur, and in the corresponding indexes in b it ~~takes the~~ displays the number $2^{([\log_2 cur]+1)}$ if $cur > 0$, and 1 if $cur \leq 0$.

0	1
1	2
2	4
3	4
4	8
5	8

b) row 1: $\text{for(int } \overbrace{i=0}^{1}; \overbrace{i < a.length}^{n+1}; \overbrace{i{+}{+}}^{n})$

a.length $= n$, $cur = m$.

$1 \cdot 1 + 2 \cdot (n+1) + 1 \cdot n = 3n+3$

row 2: int cur = a[i]; - 2 prim. op.

$2n$.

row 3: b[i] = 1; - 2 prim. op.

$2n$.

row 4: while (cur > 0) - 1 prim. op.

This check is done $([\log_2 m]+1)$ times and for every element in a. ~~So overall,~~

So, overall, $n \cdot ([\log_2 m]+1)$

row 5: b[i] <<= 1; - 3 prim. op.

$3n \cdot ([\log_2 m]+1)$

row 6: cur >>= 1; - 2 prim. op.

$2n \cdot ([\log_2 m]+1)$

Overall, $3n+3+2n+2n+6n(\lfloor\log_2 m\rfloor+1) = O(n\cdot\log m)$

iii. a) Case 1 / a.length ≥ 100

The code iterates over the first 100 elements of an array a and multiplies all of them.

Case 2 / a.length < 100.

The code iterates over the all elements of an array a and multiplies all of them.

b) row 1: int p=1 - 1 prim. op.

1   1

row 2: for (int i=0; i<100; i++)

In worst case, i<100 is performed 101 times, i++ is performed 100 times. Overall $1+100+100 = 201$

row 3: if (i>=a.length) - 2 prim. op.

$2\cdot 100 = 200$

row 4: break; - 1 prim. op.

1.

row 5: p*=a[i] - 3 prim. op.

$3\cdot 100 = 300$

Overall, $201+200+1+300 = 702 = O(1)$

iv. a) For every index k of the array c, the code assignes to c[k] the sum of the multiples of a[i] and b[j], where $j+i=k$.

b) row 1. for (int $\overset{1}{k=0}$; $\overset{n+k+1}{k<a.\text{length}+b.\text{length}}$; $\overset{n+k}{k\text{++}}$)

k < a.lenght + b.length - 4 prim. op.

a.lenght = n, b.lenght = k

$1\cdot 1 + 4(n+k+1) + 1\cdot(n+k) = 5n+5k+5$

row 2: $c[k] = 0$ - 2 prim. op.

$2 \cdot (n+k)$

row 3: $\text{for (int } i=0;\ \overbrace{i < a.length}^{n+1};\ \overbrace{i{+}{+}}^{n})$

$1 \cdot 1 + 2(n+1) + 1 \cdot n = \underline{3n+3}$

row 4: $\text{for (int } j=0;\ \overbrace{j < b.length}^{k+1};\ \overbrace{j{+}{+}}^{k})$

$n \cdot (1 \cdot 1 + 2(k+1) + 1 \cdot k) = \underline{(3k+3) \cdot n}$

row 5: $c[i+j] += a[i] * b[j];$ - 6 prim. op.

$6 \cdot n \cdot k$

Overall, $5n + 5k + 5 + 2(n+k) + 3n + 3 + n(3k+3) +$
$+ 6 \cdot n \cdot k = \underline{O(n \cdot k)}$

V. a) The code prints all the differences $a[j]-a[i]$, where j is an index starting from the index $\underline{i+1}$. In general, it takes $a[i]$ and prints all the differences with the following elements.

b) row 1: $\text{for (int } \overbrace{i=0}^{1};\ \overbrace{i < a.length}^{n+1};\ \overbrace{i{+}{+}}^{n})$

$1 + 2(n+1) + n = \underline{3n+3}$

~~row 2: for (int j = i+1; j < a.length; j++)~~

~~$i=0$ - $n-1$~~
~~$i=1$ - $n-2$~~
~~$\vdots$~~
~~$i=n-1$ - 1~~ ~~$\Rightarrow \frac{(n-1) \cdot n}{2}$~~

~~This line is done $\frac{n(n-1)}{2}$ times.~~

~~$\frac{n \cdot (n-1)}{2}$ (~~

row 2) for (int j = i+1; j < a.length; j++)

$$i = 0 \quad - \quad 1 \cdot 1 + 2 \cdot n + 1 \cdot (n-1)$$
$$i = 1 \quad - \quad 1 \cdot 1 + 2(n-1) + 1 \cdot (n-2)$$
$$\vdots$$
$$i = n-1 \quad - \quad 1 \cdot 1 + 2 \cdot 1 + 1 \cdot 0$$

=>

$$1 \Rightarrow \quad n \cdot 1 + 2\left(n + (n-1) + \dots + 1\right) + 1 \cdot \left((n-1) + (n-2) + \dots + 0\right) =$$
$$= n + 2\left(\frac{n \cdot (n+1)}{2}\right) + 1 \cdot \left(\frac{n(n-1)}{2}\right)$$

row 3: System.out.println(a[j] - a[i]); - 4 prim. op.

$$i = 0 \quad - \quad n-1$$
$$i = 1 \quad - \quad n-2$$
$$\vdots$$
$$i = n-1 \quad - \quad 0$$

=> $\frac{(n-1) \cdot n}{2}$ times the body of for loop in line 2 is done.

Overall, $\frac{n(n-1)}{2} \cdot 4 = \underline{2n(n-1)}$.

Overall, $3n + 3 + n + 2\left(\frac{n(n+1)}{2}\right) + \frac{n(n-1)}{2} + 2n(n-1) =$

$$= \underline{O(n^2)}$$

c) 1) iii - $O(1)$ ii - $O(n \log m)$

2) i - $O(n)$ iv - $O(n \cdot k)$

3) v - $O(n^2)$

ii and iv cannot be incorporated into the ranking, because their Big-Oh estimate of running times are functions dependent on two different variables. That is why, we cannot compare functions of one variables with functions of two variables.

d) No Big-Theta relations.

Problem 2 / Order the following functions by asymptotic growth rate.

1) $1250 \times 2^{45} = O(1)$
2) $10 + \log n = O(\log n)$
3) $60\log n + 10n = O(n)$ $\qquad \lim\limits_{n\to\infty} \frac{15n}{60\log n + 10n} \approx 1{,}5$
4) $15n = O(n)$ $\qquad \lim\limits_{n\to\infty} \frac{5\log n + 48n}{15n} = \frac{48}{15} = 3{,}2$
5) $5\log n + 48(n) = O(n)$
6) $3n + 2n\log n = O(n\log n)$
7) $2n^2 = O(n^2)$
8) $8n^2 + 2n^3 = O(n^3)$ $\qquad \lim\limits_{n\to\infty} \frac{5n^3}{8n^2+2n^3} = \lim\limits_{n\to\infty} \frac{5\cancel{n^3}}{\cancel{n^3}(\underbrace{\frac{8}{n}}_{\to 0}+2)} = 2{,}5$
9) $5n^3 = O(n^3)$
10) $17n^3 + n^4 = O(n^4)$
11) $2^n = O(2^n)$
12) $\frac{9}{2}^n = O(\frac{9}{2}^n)$ $\quad (\frac{9}{2} > 2)$

$\lim\limits_{n\to\infty} \frac{5n^3}{8n^2+2n^3} = 2{,}5$, which means that the growth rate of $5n^3$ is 2,5 times greater than the growth rate of $8n^2+2n^3$

$\lim\limits_{n\to\infty} \frac{5\log n + 48n}{15n} = 3{,}2 \Rightarrow$ the growth rate of $5\log n + 48n$ is 3,2 times the growth rate of $15n$.

$\lim\limits_{n\to\infty} \frac{15n}{60\log n + 10n} = 1{,}5 \Rightarrow$ the growth rate of $15n$ is 1,5 times the growth rate of $60\log n + 10n$.