CSE 551 – Assignment 2

Submitted By:
Ankit Sharma
1219472813
ashar263@asu.edu

<u>Solution for Question (1)</u>

$F'(n) = F'(n-1) + F'(n-2) + F'(n-3) + F'(n-4)$

For finding $F'(n)$ using above definition and matrix chain multiplication method we use a 1x4 matrix with values as:

$[F'(n-3) \quad F'(n-2) \quad F'(n-1) \quad F'(n)]$ …………..**eq(i)**

$= [F'(n-3) \quad F'(n-2) \quad F'(n-1) \quad F'(n-1) + F'(n-2) + F'(n-3) + F'(n-4)]$

$= [F'(n-4) \quad F'(n-3) \quad F'(n-2) \quad F'(n-1)] \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

Let $\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = A$. Therefore, above equation now becomes:

$= [F'(n-4) \quad F'(n-3) \quad F'(n-2) \quad F'(n-1)] \; A$

Now expanding $F'(n-1)$ with a similar approach like we used for eq(i), we get:

$= [F'(n-5) \quad F'(n-4) \quad F'(n-3) \quad F'(n-2)] \; A^2$

.

.

. (continuing this till the last step)

.

.

$= [F'(0) \quad F'(1) \quad F'(2) \quad F'(3)] \,.\, A^{n-3}$

Substituting the given values for $= F'(0), \quad F'(1), \quad F'(1) \; and \quad F'(3)$ in the above equation, we get:

$= [0 \quad 1 \quad 1 \quad 1] \; A^{n-3}$

Above equation can be rewritten as:

$= [0 \quad 1 \quad 1 \quad 1] \; A^{n-3} \; (A^{-1}A^1)(A^{-1}A^1)(A^{-1}A^1)$, because $A^{-1}A^1 = 1$

$= [0 \quad 1 \quad 1 \quad 1] \; (A^{-1}A^{-1}A^{-1}) \; A^n$ …………….**eq(ii)**

Now, $A^{-1} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Therefore, $A^{-1}A^{-1}A^{-1} = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 2 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix}$

Substituting this value in eq(ii):

$$= \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & 1 \\ 2 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix} A^n$$

$$= \begin{bmatrix} 1 & 1 & -1 & 0 \end{bmatrix} A^n$$

Therefore,
$[F'(n-3) \quad F'(n-2) \quad F'(n-1) \quad F'(n)] = [1 \quad 1 \quad -1 \quad 0]\, A^n$ **……..eq(iii)**

Now we know that, from the proof of $F(n) = F(n-1) + F(n-2)$ using matrix chain multiplication technique, value of A^n can be determined in $O(logn)$ time.

Therefore, $\boldsymbol{F'(n) = F'(n-1) + F'(n-2) + F'(n-3) + F'(n-4)}$ can also be computed in $\mathbf{O}(\boldsymbol{logn})$ time using eq(iii)**.**

<u>Solution for Question (2)</u>

Data: *Input A and n*
Result: *Return B*

```
for i = 1, 2, ..., n do
        for j = i + 1, i + 2, ..., n − 1 do
                Add up array entries A[i] through A[j]
                Store the result in B[i; j]
        end
end
Return B;
```

- **Give a bound of the form O(f(n)) on the running time of this algorithm**

Outer for-loop (line 1) iterations = n
Inner for-loop (line 2) iterates up to = $n-2$
Number of 'Add up array entries $A[i]$ through $A[j]$' operations are up to = $n-2$
Store the result in $B[i;\ j]$ operation = 1

Total operations = $n*(n-2)*(n-2+1) = n^3-3n^2+2n \in\ \boldsymbol{O}(\boldsymbol{n^3})$

Therefore, running time of this algorithm on an input of size n is $\boldsymbol{O}(\boldsymbol{n^3})$

- **Show that the running time of the algorithm on an input of size $\boldsymbol{n}$ is also $\boldsymbol{\Omega}(\boldsymbol{n}\text{^}\boldsymbol{3})$**

To prove, $n^3-3n^2+2n \in\ \Omega(n^3)$

To prove above equation, by the definition of Ω- notation, we need to prove that:

$cn^3 \leq\ n^3-3n^2+2n\ \forall\ n \geq n_o$
where c and $n_o$are positive constants.

We can re-write the above equation as:
$c'n*c'n*c'n \leq\ n^3-3n^2+2n$, where $c' = c^{1/3}$

$c'n*c'n*c'n \leq\ n*(n-2)*(n-1)$

We can see that if c' is a positive fraction less than 1 like 0.01 and $n \geq n_o$ where $n_o = 3$, above equation holds true as $c'n$ will be less than $n, (n-1)\ and\ (n-2)$

Hence proved, $\boldsymbol{n^3-3n^2+2n} \in\ \boldsymbol{\Omega}(\boldsymbol{n^3})$

- **Algorithm to solve this problem, with an asymptotically better running time.**

$\boldsymbol{Data}$: $Input\ A\ and\ n$

$memSum = 0$
$for\ i\ =\ 1,2,\ldots,n\ do$
  $memSum = A[i]$
  $for\ j\ =\ i\ +\ 1, i\ +\ 2, \ldots, n\ -\ 1\ do$
    $memSum = memSum + A[j]$
    $Store\ the\ result\ in\ B[i;\ j]$
  end
end
$Return\ B;$

We replace adding all array entries $A[i]\ to\ A[j]$ to just adding $A[j]$ to $memSum$ which stores sum of all the previous iterations of $A[i], A[i+1], \dots \dots \dots \dots .. A[j-1]$ for a given i value.

Since, $memSum = memSum + A[j]$ can be done in $O(1)$, we are only left with two nested for-loop running $n\ times$ and $(n-2)\ times$.

Therefore, running time of this modified algorithm on an input of size n is $\boldsymbol{O}(\boldsymbol{n^2})$

Solution for Question (3)

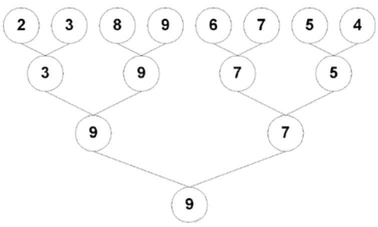
Given: Unsorted sequence of numbers $\{a_1, a_2, \dots \dots \dots \dots \dots ., a_n\}$

Below is the algorithm to find 2^{nd}smallest number in an unordered sequence –

High level description:

1. $Find\ smallest\ number\ \boldsymbol{min}\ using\ tournament\ approach.$
2. $Store\ all\ numbers\ that\ were\ compared\ to 'min' in\ an\ comp[\quad]\ array$
3. $Find\ smallest\ number$ ***min2*** $element\ in\ the\ comp[\quad]\ array$
4. $Return$ ***min2***

In step 1, $\boldsymbol{n-1}$ comparisons are done in tournament approach as all elements need to be compared. Below is a diagram, explaining tournament approach for finding max element.



In step 2, since the problem is being divided by 2 in every iteration, no of calls = $\lceil \log_2 \mathrm{n} \rceil$
Therefore, max numbers of elements that can be in $comp[\,] = \lceil \log_2 \mathrm{n} \rceil$
Hence, max numbers of comparisons that can be done in $comp[\,] = \lceil \log_2 \mathrm{n} \rceil - 1$

Total comparisons in step 1 and 2 = $(n-1) + \lceil \log_2 \mathrm{n} \rceil - 1$

$$= n + \lceil \log_2 n \rceil - 2$$

Therefore, above algorithm computes the 2nd smallest number in an unordered (unsorted) sequence of numbers $\{a_1, a_2, \dots \dots \dots \dots \dots ., a_n\}$ in $\boldsymbol{n} + \lceil \mathbf{log_2\, n} \rceil - \mathbf{2}$ comparisons in the worst case.

<u>Solution for Question (4)</u>

Given, $n = 2p$
and, Multiplications in $VW = \frac{3n}{2} = 3p$

Let first calculate for a 4x4 matrix where,

$V_1 = (v_1', v_2', v_3', v_4')$
$V_2 = (v_1'', v_2'', v_3'', v_4'')$
$V_3 = (v_1''', v_2''', v_3''', v_4''')$
$V_4 = (v_1'''', v_2'''', v_3'''', v_4'''')$

and

$W_1 = (w_1', w_2', w_3', w_4')$
$W_2 = (w_1'', w_2'', w_3'', w_4'')$
$W_3 = (w_1''', w_2''', w_3''', w_4''')$
$W_4 = (w_1'''', w_2'''', w_3'''', w_4'''')$

Then,

$$\begin{bmatrix} v_1' & v_2' & v_3' & v_4' \\ v_1'' & v_2'' & v_3'' & v_4'' \\ v_1''' & v_2''' & v_3''' & v_4''' \\ v_1'''' & v_2'''' & v_3'''' & v_4'''' \end{bmatrix} \begin{bmatrix} w_1' & w_2' & w_3' & w_4' \\ w_1'' & w_2'' & w_3'' & w_4'' \\ w_1''' & w_2''' & w_3''' & w_4''' \\ w_1'''' & w_2'''' & w_3'''' & w_4'''' \end{bmatrix} = \begin{bmatrix} V_1W_1 & V_1W_2 & V_1W_3 & V_1W_3 \\ V_2W_1 & V_2W_2 & V_2W_3 & V_2W_4 \\ V_3W_1 & V_3W_2 & V_3W_3 & V_3W_4 \\ V_4W_1 & V_3W_2 & V_3W_3 & V_3W_4 \end{bmatrix}$$

Values of V_xW_y are calculated using the vector product VW by the formula:
$$\sum_{1 \le i \le p} (v_{2i-1} + w_{2i}) * (v_{2i} + w_{2i-1}) - \sum_{1 \le i \le p} v_{2i-1} * v_{2i} - \sum_{1 \le i \le p} w_{2i-1} * w_{2i}$$

Generalizing above equation for a nXn matrix we get:

$$\begin{bmatrix} V_1W_1 & V_1W_2 & \dots\dots & V_1W_n \\ V_2W_1 & V_2W_2 & \dots\dots & V_2W_n \\ \dots & \dots\dots & \dots\dots & \dots\dots \\ V_nW_1 & V_nW_2 & \dots\dots & V_nW_n \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_n \end{bmatrix} [W_1 \quad W_2 \quad W_3 \dots\dots\dots \quad W_n] \text{ ....}\mathbf{eq(i)}$$

We are given that V_1W_1 will need $3p$ multiplications using the $VW\ vector$ vector product formula.

Now, for $V_1W_{i(2 \le i \le n)}$ we will only require $2p$ multiplication as $\sum_{1 \le i \le p} v_{2i-1} * v_{2i}$ term is independent of w and need not be calculated as its already done in V_1W_1

Similarly, for $V_{i(2 \le i \le n)}W_1$ we will only require $2p$ multiplication as $\sum_{1 \le i \le p} w_{2i-1} * w_{2i}$ term is independent of v and need not be calculated as its already done in V_1W_1

Expanding on this thought process we can denote multiplications needed for calculating elements for matrix with dimensions nXn in eq(i), we get:

$$\begin{bmatrix} 3p & 2p & 2p & \dots. & 2p \\ 2p & p & p & \dots. & p \\ 2p & p & p & \dots. & p \\ \dots & \dots. & \dots. & \dots. & \dots. \\ 2p & p & p & p & p \end{bmatrix}_{n\,X\,n}$$

Calculating total multiplications needed:

$$3p + 2p(n-1) + 2p(n-1) + p(n-1)(n-1)$$

$$= 3p + 2pn - 2p + 2pn - 2p + pn^2 + p - 2pn$$

$$= pn^2 + 2pn$$

Substituting value $p = \frac{n}{2}$

$$= \frac{n^3}{2} + n^2$$

Therefore, vector product VW formula for the multiplication of two $n\ X\ n$ matrices require ($\frac{n^3}{2} + n^2$) multiplications.

<u>Solution for Question (5)</u>

$$\begin{aligned} T(n) &= n^2 + 16T\left(\frac{n}{4}\right) \\ &= n^2 + 16[\ \left(\frac{n}{4}\right)^2 + 16T\left(\frac{n}{4^2}\right)\ ] \\ &= n^2 + 16\left(\frac{n}{4}\right)^2 + 16^2T\left(\frac{n}{4^2}\right) \\ &= n^2 + 16\left(\frac{n}{4}\right)^2 + 16^2[\ \left(\frac{n}{4^2}\right)^2 + 16T\left(\frac{n}{4^3}\right)\ ] \\ &= n^2 + 16\left(\frac{n}{4}\right)^2 + 16^2\left(\frac{n}{4^2}\right)^2 + 16^3T(\frac{n}{4^3}) \\ &= n^2 + (1)n^2 + (1)n^2 + 16^3T(\frac{n}{4^3}) \end{aligned}$$

After x iterations we will get:

$= x\, n^2 + 16^x T(\frac{n}{4^x})$ ……………(i)

This recurrence will end when $\frac{n}{4^x} = 1$

We are given that $n = 4^k$, therefore recurrence ends when $x = k$ in eq(i). Substituting values in eq(i) we get,

$= kn^2 + 16^k T(1)$, where $T(1) = 1$ is given

Since, $n = 4^k \Rightarrow k = \log_4 n$

$= (\log_4 n)n^2 + 16^{\log_4 n}(1) = (\log_4 n)n^2 + 4^{2\log_4 n} = (\log_4 n)n^2 + n^2$

Therefore, $\boldsymbol{T(n) = n^2(\log_4 n + 1)}$

<u>Solution for Question (6)</u>

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

$= 3^n \left[\, 3^{\frac{n}{2}}\, T\left(\frac{n}{2^2}\right) \right]$

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

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

$= 3^{n+\frac{n}{2}+\frac{n}{2^2}+ \dots\dots\dots + \frac{n}{2^{x-1}}}\, T\left(\frac{n}{2^x}\right) \dots\dots\dots\dots\dots eq(i)$

This recurrence will end when $\frac{n}{2^x} = 1$

We are given that $n = 2^k$, therefore recurrence ends when $x = k$ in eq(i). Substituting value in eq(i) we get,

$= 3^{n+\frac{n}{2}+\frac{n}{2^2}+ \dots\dots\dots + \frac{n}{2^{k-1}}}\, T(1)$, where $T(1) = 1$ is given ………………eq(ii)

Now,

$n + \frac{n}{2} + \frac{n}{2^2} + \dots\dots\dots.. + \frac{n}{2^{k-1}} = \frac{n\left(1 - \frac{1}{2}^k\right)}{1 - \frac{1}{2}} = 2n(1 - \frac{1}{2^k})$

We are given $n = 2^k \Rightarrow k = \log_2 n$. Substituting value of k in above equation we get:

$$2n\left(1 - \frac{1}{2^k}\right) = 2n\left(1 - \frac{1}{2^{\log_2 n}}\right) = 2n\left(1 - \frac{1}{n}\right) = 2(n-1)$$

Therefore, $n + \frac{n}{2} + \frac{n}{2^2} + \dots\dots\dots.. + \frac{n}{2^{k-1}} = 2(n-1)$

Substituting this value in eq(ii), we get:

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

$\boldsymbol{T(n) = 9^{n-1}}$