

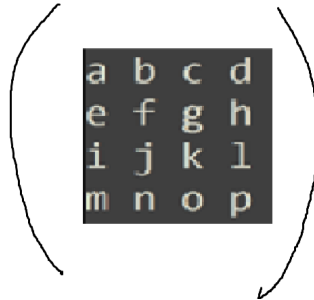
Student Number: 1226909816

Name: Wei Hng Yeo

CSE450 Assignment 2

- 1) Yes, it can be done in  $O(\log(n))$  time.

We can derive a  $4 \times 4$  matrix which will be used as the base case matrix. Taking the example matrix below as a reference.



|   |   |   |   |
|---|---|---|---|
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |
| m | n | o | p |

Then taking matrix and multiply by the base case matrix to get matrix  $M_1$ . We will get the a to be  $F'(4)$  in  $M_1$ . Then as we take the matrix  $M_1$  to multiply by the base case matrix to get  $M_2$ . We will see  $F'(5)$  to be a in  $M_2$ .

The above results in  $O(n)$  to compute the  $n$ th term of  $F'$ .

To get  $O(\log(n))$ , we will need to use exponentiation by squaring. So instead of iteratively multiplying each matrix product by the base case matrix. Let the base case matrix be  $M_0$ . We can use divide and conquer to split the matrix multiplication into squaring the matrix. So instead of taking  $MN = M_0 * M_0 * \dots$ , we instead do  $MN = MN' * MN'$  where  $N' = N / 2$ .

- 2)  $T(n) = T(n - 1) + n / 2$ , where  $T(1) = 1$   
 $= T(n - 2) + (n - 1) / 2 + n / 2$   
 $= T(n - 3) + (n - 2) / 2 + (n - 1) / 2 + n / 2$   
 $= T(n - (n - 1)) + (2 + 3 + 4 + 5 + \dots + n) / 2$   
 $= T(1) + ((2 + 3 + 4 + 5 + \dots + n) / 2)$   
 $= 1 + ((2 + 3 + 4 + 5 + \dots + n) / 2)$   
 $= 1 + ((n - 1)(n + 2) / 4)$

Thus, the complexity will be  $O(n^2)$ .

- 3) Firstly, we divide the players into 2 equal groups. Then let them play within the groups for the first  $((n/2) - 1)$  days.

Then, schedule the games between the groups for the other  $n / 2$  days.

Let  $arr\_p\_d$  be an  $(n - 1)$  by  $n$  array, where  $arr\_p\_d[d, i]$  records who player  $P_i$  plays with on day  $d$ .

The function  $recursive\_scheduler(arr\_p\_d, i, j)$  will compute the schedule and assign the results to  $arr\_p\_d$  for players  $i$  to  $j$  for days 1 to  $j - i$ .

As for the base case:  $recursive\_scheduler(arr\_p\_d, i, i + 1)$  which is just 2 players. Schedule them to play each other on the first day.

Subsequent recursive case:  $recursive\_scheduler(arr\_p\_d, i, j)$  where  $j > i + 1$ . Then let  $k = i + n/2$ . Then call  $recursive\_scheduler(arr\_p\_d, i, k-1)$  and  $recursive\_scheduler(arr\_p\_d, k, j)$ . This schedule the first half of the players and the second half of the players from days 1 to  $((n / 2) - 1)$ .

We only need to schedule each player in the first half against all the other players in the second half. We will schedule by, for  $m = 0$  to  $((n / 2) - 1)$ , we schedule player  $i$  against player  $k + m$  on day  $((n / 2) + m)$ . We also schedule player  $i + 1$  against player  $k + ((1 + m) \% (n / 2))$  on day  $((n / 2) + m)$ .

Lastly,  $recursive\_scheduler(arr\_p\_d, 1, n)$  is executed and then a matrix of player vs day will be shown the result. So, we will print the matrix out column by column to see which player is pair with which other player on day  $d$ .

Therefore, a total of  $n - 1$  days is required to complete the tournaments.

Time Complexity:  $\Theta(n^2)$

Space Complexity:  $\Theta(n^2)$

- 4) We can do this problem in  $O(n)$ , we simply need 2 variables one to store the largest number (1st\_num) and another one to store the second largest number (2nd\_num). We need to store the first element in the array to be in 1st\_num and second element in the array 2nd\_num depending on whether the first or second number in the array is larger. Then for each iteration through the array, if the current number is larger than the number, we have in the variable used to store the largest number 1st\_num, we copy the value in the 1st\_num to 2nd\_num then assign that new value to 1st\_num.

Below is a pseudo code.

```
if (a[0] < a[1])
    1st_num = a[1]
    2nd_num = a[0]
else
    1st_num = a[0]
    2nd_num = a[1]
```

```
for (each element in array starting from index 2) {  
    if (a[i] > 1st_num) {  
        2nd_num = 1st_num  
        1st_num = a[i]  
    }  
}  
  
return 2nd_num
```