23CCE201 - Data Structures

Quiz

Harish.J
CB.EN.U4CCE23016

1) Friends in Social Networks - Each n user spends some time on a social media site. For each $i = 1, \dots n$ user i enters the site at time a_i and leaves at time $b_i > a_i$. How many distinct pairs of users are ever on the site at the same time?
Given input (a_1, b_1) (a_2, b_2) ... (a_n, b_n). Explain an algorithm to compute the number of pairs of users on the site at the same time.

Logic:

*) To solve this problem, we could employ an undirected graph, where each vertex represent the user. The input is given in terms of the entry time and exit time of the user ⇒ $[[a_1, b_1], [a_2, b_2], [a_3, b_3], \dots [a_n, b_n]]$ ($a_i \rightarrow$ entry time, $b_i \rightarrow$ exit time)

*) We could devise an adjacency matrix for the user vertices and compute the weights as the time spent simultaneously on the site.

*) We could use a nested loop to traverse through the input and take the entry & exit times of two users. Now, we should compare if the maximum value of their ~~inp~~ entry times is less than the minimum value of their exit times. If this is true, there is an overlap and we could store the weight of the edge in the adjacency matrix as ~~the~~ an array as $[\max(a_i, a_j), \min(b_i, b_j)]$.

*) A counter pointer can be initialized and incremented whenever there is an overlap between the users and have a count of total number of users that were simultaneously using the site.

*) When we need to find the number of user at the same time, we could traverse the adjacency matrix and compute ~~the~~ if the weight in terms of the time interval contains the desired time input. If yes, increment the counter and show the output to the user.

Algorithm: input → [[a1,b1],[a2,b2], . . . [an,bn]]

Adjacency matrix (input)

```
1) Initialise the graph as array[n][n], where n is
   the length of input array, initially everything as 0.
2) Initialize a variable count = 0.
3) Initialize a for loop from range(0 to n):
    3.1) In the nested for loop from range (0 to i):
        3.1.1) Get the entry and exit times with input array
               ai = input[i][0], bi = input[i][1] and
               aj = input[j][0], bj = input[j][0]
        3.1.2) ~~start~~ set start = max(ai, aj)
               set end = min(bi, bj)
        3.1.3) If start <= end:
               • Store the interval as the weight of the graph
               • graph[i][j] = [start, end]
               • graph[j][i] = [start, end]
               • Increment the overall counter.
4) Return the adjacency matrix and counter.
```

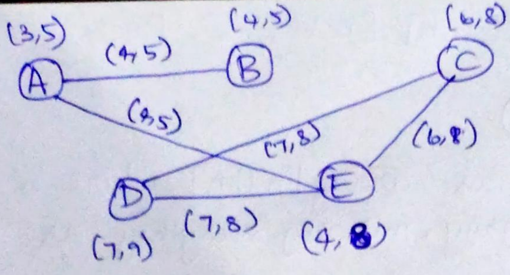
Particular Time (time, input)

```
1) Call the function adjacency matrix to store the
   correct representation.
2) Initialize counter = 0
3) Iterate from 0 to n using a variable i:
    3.1) In nested loop iterate from 0 to i+1 with j:
        3.1.1) Check if the given time lies in the weight of
               the graph.
               If time in graph[i][j]:
                   • counter = counter + 1.
4) Return counter.
```

Visualization:

Input = [[3,5],[4,5],[6,8][7,9][4,8]]

A B C D E → user



adjacency matrix:

	A	B	C	D	E
A	0	[4,5]	0	0	[4,5]
B	[4,5]	0	0	0	0
C	0	0	0	[7,8]	(6,8)
D	0	0	[7,8]	0	[7,8]
E	[4,5]	0	[6,8]	[7,8]	0