CCC '02 S4 – Bridge Crossing (Editorial)

All Subtasks:

The solution for this problem utilizes a dynamic programming approach. Two main arrays are maintained, dp and $group$. Specifically, $dp[i]$ is the optimal time for the first i people. $group[i]$ contains the length of the group with i as the last person. For each i (person), we are trying to find which group would be best to place in order to keep the time minimal. Since we are given M as the total number of people there can be 1 group and we can loop over $[1...M]$. So we loop over $[1...q]$ as i and $[1...M]$ as j, the answer for $dp[i]$ lies within the answer of $dp[i-j]$. However can only place i in one of the groups. So we maintain a max variable to check if we can place i in a group that produces a smaller result. The final answer will be $dp[q]$.

Time Complexity: $O(NM)$

Example:

Let us say that our sample input is:

```
2 5
alice 1
bob 5
charlie 5
dobson 3
eric 3
```

Our dp array would be:

```
People = alice bob charlie dobson eric
DP     =   0    1     5      6     9
```

To form the groups, we maintain a $group$ array. $group[i]$ contains the length of the group with i as the last person. This is how we built our groups.

```
Index   = 0 1 2 3 4
People  = a b c d e
group[] = 0 0 1 2 3
```

To get the group that e is in, we can tell that $group[e]$ is 3. This means e is in a group that starts from index 3. Index(3) is d. So the group for any i is $[group[i] \ldots index[i]]$.