① Pseudocode

```
optN(0) = N4[0]
opt.S(0) = SF[0]
for i = 1 ... n
    optN(i) = N[i] + min(optN(i-1), (m + optS(i-1)))
    optS(i) = S[i] + min(opt(i-1), (m + optN(i-1)))
end
Return min(optN(n), optS(n))
```

Complexity:

T(n) = O(n) Linear Time works as complexity until a single for loop will perform as much iterations as the number of elements of arrays.

(2)

1- Sort the seasons according their finishing time

2- Select the first activity from the sorted array and print

3- Do following for remaining activities in the sorted array

if the start time of season is greater than or equal to finish time of previously selected activity then select this activity and print it.

It takes $O(n \log n)$ time if input activities may not be sorted. It takes $O(n)$ time when it is given that input activities are always sorted

(4) → Initialize first row and first column to be 0.

→ The score of the best local alignment is the largest value in the entire array.

→ To find the actual local alignment

* Start at an entry with the maximum score
* traceback as usual
* stop when we reach an entry with a score of 0

There are two loops so Time Complexity is $O(n^2)$

(5)

* Firstly find two minimum elements in array ad add them
* Insert the sum of this two elements
* Delete these two element from list
* Do this until there are one element, then return it, it will be the result.

Time Complexity is $O(n^2)$