(2) Suppose n activities apply for using a common resource. Activity a_i ($1 \leq i \leq n$) has a starting time $S[i]$ and a finish time $F[i]$ such that $0 < S[i] < F[i]$. Two activities a_i and a_j ($1 \leq i, j \leq n$) are compatible if intervals $[S[i], F[i])$ and $[S[j], F[j])$ do not overlap. We assume the activities have been sorted such that $S[1] \leq S[2] \leq ... \leq S[n]$.

A Design an $\mathcal{O}(n^2)$ dynamic programming algorithm to find a set of compatible activities such that the total amount of time the resource is used by these compatible activities is maximized. You need to define the subproblems, establish the inductive formula and show the initial conditions. Pseudocode is not required.

B Apply your algorithm to the following set of activities:

i	1	2	3	4	5	6	7	8	9	10	11
S[i]	2	3	5	6	7	9	10	12	13	14	16
F[i]	6	5	7	10	8	13	16	14	14	18	20

(2(A))

Algorithm 1 A dynamic programming algorithm usable to solve the activity problem above in $\mathcal{O}(n^2)$ time. In this algorithm...

```
1: function MaxActivities(S[1..n], F[1..n])
2:     M ← P ← ∅
3:     Initialize(M, P)
4:     for i from 1 to n do                          ▷ Locate max for m_i
5:         max ← M[i]
6:         for j from 1 to i do
7:             if max < M[j] and F[j] < S[i] then
8:                 max ← M[j]
9:                 P[i] ← j
10:            end if
11:        end for
12:    end for
13:    max ← 1                                       ▷ Find global maximum
14:    for i from 2 to n do
15:        if M[max] < M[i] then
16:            max ← i
17:        end if
18:    end for
19:    return max, P
20: end function
```