

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  $S[1] \leq S[2] \leq \dots \leq S[n]$ .

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:    $M \leftarrow P \leftarrow \emptyset$ 
3:   INITIALIZE( $M, P$ )
4:
5:    $M[1] \leftarrow 1$ 
6:    $P[1] \leftarrow 0$ 
7:   for  $i$  from 2 to  $n$  do                                      $\triangleright$  Locate max for  $m_i$ 
8:      $max \leftarrow 1$ 
9:     for  $j$  from 1 to  $i$  do
10:      if  $max < M[j]$  and  $F[j] < S[i]$  then
11:         $max \leftarrow M[j]$ 
12:         $P[i] \leftarrow j$ 
13:      end if
14:    end for
15:     $M[i] \leftarrow max$ 
16:  end for
17:
18:   $max \leftarrow 1$                                               $\triangleright$  Find global maximum
19:  for  $i$  from 2 to  $n$  do
20:    if  $M[max] < M[i]$  then
21:       $max \leftarrow i$ 
22:    end if
23:  end for
24:
25:  return ( $max, P$ )
26: end function

```

---

With this algorithm in mind, the answer is as follows:

**Inductive Formula:**  $\{m(i) = \max_{1 \leq j \leq i} (m_j) + 1 \mid F[j] < S[i]\}$

**Initial Conditions:**  $m(1) = 1$

The **subproblem** can be thought of as follows. Each activity,  $i$ , will be added to a chain of activities resulting in the maximum set that includes activity  $i$ . Activity  $i$  is always included in its maximum. Its maximum,  $m_i$ , must also be based off of previous activities which are compatible.

## 2.B

*After Initialization.*

i	1	2	3	4	5	6	7	8	9	10	11
M[i]	1	0	0	0	0	0	0	0	0	0	0
P[i]	0	0	0	0	0	0	0	0	0	0	0

*Upon Completion.*

i	1	2	3	4	5	6	7	8	9	10	11
M[i]	1	0	0	0	0	0	0	0	0	0	0
P[i]	0	0	0	0	0	0	0	0	0	0	0