**Exercise 15.1**

**1. Give a dynamic-programming algorithm for the activity-selection problem, based on recurrence (15.2). Have your algorithm compute the sizes as defined above and produce the maximum-size subset of mutually compatible activities.**

**Ans)**

Recurrence:

c[i,j] = { 0 Sij = ϕ

max{c[i,k] + c[k,j] + 1 : ak in Sij Sij =! ϕ

}

DYNAMIC-ACTIVITY-SELECTOR(A,s,f,n):

let c[0..n+1,0..n+1] be the table where c[i,j] is the size of the subset between activities ai and aj

let act[0.. n+1, 0..n+1] be an auxiliary table

for i =0 to n:

for j =0 to n:

c[i,j] = 0

AUXILLIARY-DYNAMIC-ACTIVITY-SELECTOR(A,c,s,f,n)

 AUXILLIARY-DYNAMIC-ACTIVITY-SELECTOR(A,c,s,f,n):

if

**3. Not just any greedy approach to the activity-selection problem produces a maximum-size of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest starting time.**

Ans)

Consider a set of activities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 |
| s | 0 | 3 | 5 | 10 |
| f | 2 | 5 | 7 | 11 |
| duration | 2 | 2 | 2 | 1 |

Consider the set of activities above. The activity with the least duration is activity 4. If we select this activity first, we will not be able to select any other activity.

Activity Set using this approach: {a4}

Optimal activity set: {a1, a2, a3, a4}

Consider a set of activities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 |
| s | 0 | 3 | 7 | 10 |
| f | 4 | 8 | 9 | 11 |

Here, a4 has no overlaps. All other activities have 1 overlap at time 0. So, we select a4 at first

Activity Set using this approach: {a4}

Optimal activity set: {a1,a3,a4}

Consider a set of activities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 |
| s | 0 | 3 | 7 | 10 |
| f | 100 | 5 | 9 | 11 |

Here, if we consider activities based on their start time, a1 has the smallest start time. Therefore, we select this first

Activity Set using this approach: {a1}

Optimal activity set: {a2,a3,a4}

4.

Let h be the number of lecture halls

SCHEDULE-LECTURE-HALL(s,f,n,h):

Let h be an array of linked lists. Each element of h describes a booked lecture hall

// assuming the activities are sorted on their finish times

j = 0

while h[j].head!=NULL || h[j].head.f < si:

j++

if h[j].head == NULL:

h[j].head = new Node(i,fi)

else:

LIST-INSERT(h[j] , new Node(i,fi))

return h

5.