① Active Selection Problem

We are given a set of activities $S = \{a_1, a_2, a_3 \dots a_n\}$.

Each activity has a start time s_i and finish time f_i.

The activities are sorted such that $f_1 \le f_2 \le \dots \le f_n$.

The goal is to select the maximum number of compatiable activities.

② Dynamic Programming.

Yes, this problem has optimal substructure. because the optimal solution between activity a and p can be constructed from optimal solution of subproblems (a, k) and (k, p).

Therefore dynamic programming can be used.

The recursive idea is to select an activity k between a and p. The ~~pem~~ problem is divided into two subproblems (a, k) and (k, p). We compute the maximum number of activities recursively.

RecuSelect (s, f, a, p).

max = 0

for k = a+1 to ~~p~~ p-1

if activity k is compatible:

count = ReCuSelect (s, f, a, k) + ReCuSelect (s, f, k, p) + 1.

if count > max

max = count → return max

This recursion divides the problems into smaller subproblems.

(0, n)

(0, k) (k, n)

Example of recurrence calculation:

Suppose we have:

$a_1(1,3)$, $a_2(2,5)$, $a_3(4,6)$, $a_4(6,7)$

$c[1,3] = 1$ One compatible activity exits.

$c[0,3] = c[0,1] + c[1,3] + 1$

$c[0,3] = 0 + 1 + 1$

$c[0,3] = 2.$

$c[0,4] = c[0,3] + c[3,4] + 1$

$c[0,4] = 2 + 0 + 1$

$c[0,4] = 3$

This shows dynamic programming computation.

③ Tabulation: We store results in table

	0	1	2	3	4
0	0	0	0	0	0
1		0	1	1	2
2			0	1	2
3				0	1
4					0

Table store subproblems.

④ Greedy Approch.

Greedy selects earliest finish time.

Greedy Schedule (s, f, n)

sort by finish time

select activity 1 →

last = 1

for i = 2 to n.

if $s_i \ge f[last]$

Select activity i

last = i

~~return~~ return activities.

Example:

$a_1(1,3)$

$a_2(2,5)$

$a_3(4,6)$

$a_4(6,7)$

Select a1

Select a3

Select a4

Total = 3

Time →

a1 |----|

a2 |------|

a3 |----|

a4 |--|

Greedy gives optimal result.

Proof by induction:

Base case:

First activity is optimal.

Step:

Next activity is optimal.

Conclusion

Greedy is optimal.

Proof by contradiction

Assume greedy is not optimal.

Better solution exits.

But greedy leaves more time.

Contradiction.

Greedy is optimal.