Greedy Approach

In [ ]:
```
#   Always select the activity that finishes earliest and does not conflict wit
def greedy_select(s, f):
    selected = [0]
    last = f[0]
    for i in range(1, len(f)):
        if s[i] >= last:
            selected.append(i)
            last = f[i]
    return selected

# E.g.
s = [1, 3, 0, 5, 8, 5]
f = [2, 4, 6, 7, 9, 9]
print(greedy_select(s, f))
```

Greedy choice property: Selecting the earliest-finishing activity leaves the maximum possible time for remaining activities .

Suppose there exists a better solution that does not include the earliest-finishing activity → we can replace some activity in that solution with the earliest-finishing one without decreasing the total number → contradiction.

Dynamic Programming Approach

Recurrence:

In [ ]:
```
#   c[i][j] = max{ c[i][k] + c[k][j] + 1 } over all k such that i < k < j and ac
#   c[i][j] = 0 if no such k exists
for length in 2 to n+1:
    for i in 0 to n-length:
        j = i + length
        c[i][j] = 0
        for k in i+1 to j-1:
            if f[i] <= s[k] and f[k] <= s[j]:
                c[i][j] = max(c[i][j], c[i][k] + c[k][j] + 1)
```

Greedy: O(n) time, O(1) space, optimal, very simple

Dynamic Programming: $O(n^3)$ time, $O(n^2)$ space, also optimal, but much slower

1. Mathematical Induction

Proof by induction on r: Base case (r = 1):

Greedy always picks the activity with the absolute earliest finish time → $f(g_1)$ is minimal among all activities → $f(g_1) \le f(o_1)$.

Inductive hypothesis (IH): Assume true for all $t \le r-1$, i.e., $f(g_t) \le f(o_t)$ for t = 1 to r-1.

Inductive step (prove for r):

Consider $g_r$ (the r-th greedy choice).

By IH, $f(g_{r-1}) \le f(o_{r-1})$.

$o_r$ is compatible with $o_{r-1}$ → $s(o_r) \ge f(o_{r-1}) \ge f(g_{r-1})$.

So $o_r$ is a candidate when greedy was choosing after $g_{r-1}$.

But greedy picked $g_r$ instead → by greedy rule (earliest finish), $f(g_r) \le f(o_r)$.

By induction, the lemma holds for all $r \le k$.

2. Stay Ahead Argument

From the lemma above: greedy "stays ahead" — at every prefix, its r-th activity finishes no later than optimal's r-th activity.

This means greedy always leaves at least as much room for future activities as optimal does.

Consequence: After k steps, greedy has selected k activities.

If $m > k$ (optimal has more), then $o_{k+1}$ exists and is compatible with $\{o_1, \ldots, o_k\}$.

But $f(g_k) \le f(o_k)$ → $s(o_{k+1}) \ge f(o_k) \ge f(g_k)$ → $o_{k+1}$ is also compatible with greedy's prefix $\{g_1, \ldots, g_k\}$.

→ Greedy could have selected $o_{k+1}$ (or something at least as good) → greedy would have continued → contradiction to $|G| = k$.

Thus $m \le k$. But optimal is maximum → $m = k$ → greedy is optimal.

3. Contradiction

Assume for contradiction that greedy is not optimal, i.e., there exists an optimal solution O with $|O| = m > k = |G|$.

From the stay-ahead lemma: $f(g_r) \le f(o_r)$ for all $r \le k$.

In particular, $f(g_k) \le f(o_k)$.

Since O has $m > k$ activities, $o_{k+1}$ exists.

$o_{k+1}$ is compatible with $o_1 \ldots o_k$ → $s(o_{k+1}) \ge f(o_k) \ge f(g_k)$.

→ $o_{k+1}$ is compatible with $g_1 \ldots g_k$.

→ When greedy finished selecting $g_k$, it could have continued and picked at least one more compatible activity (like $o_{k+1}$ or the earliest-finishing one after $g_k$).

→ Greedy would have $|G| \geq k+1$ → contradiction to $|G| = k$.

Therefore, no such O with $m > k$ exists → greedy is optimal.