Selezione attività

greedy_activity_selector(s, f) // O(n) a meno dell'ordinamento di S

```
n ← |s|
A ← {1}
i ← 1
FOR m ← 2 to n do
    IF s_m ≥ f_i then        => Quella che termina prima
        A ← A ∪ {m}
        i ← m
return A
```

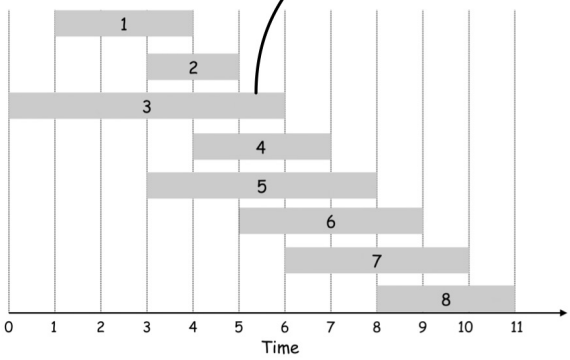
Nel caso di attività pesate si cerca di massimizzare il peso delle attività selezionate.

Soluzione => Prog. dinamica

Weighted_activity_selector(n, w, $\vec{p}$) // O(n) + O(n log n) per ordinamento + calcolo della fun. p

(p: indice massimo di attiv. compatibili ma precedenti)

```
OPT[0] ← 0
FOR j ← 1 to n do
    OPT[j] ← max(w(j) + OPT[p(j)], OPT[j-1])
return OPT[n]
```



Scelgo il max tra il peso di $a_3 = 6$ e il peso fino ad $a_2 = 3$

j	w[j]	p(j)	opt[j]
0	-	-	0
1	3	0	3
2	2	0	3
3	6	0	6
4	3	1	6
5	5	0	6
6	4	2	7
7	4	3	10
8	3	5	10

Valore massimo

Oltre al valore massimo del peso delle attività scelte si potrebbe avere necessità di trovare quali sono le attività che hanno generato quel valore:

Find_solution(j) // O(n) => Scelgo l'attività in corrispondenza di un cambio di valore in OPT[j]

```
IF    j = 0    THEN
    RETURN
ELSE IF   w(j) + OPT[p(j)] > OPT[j-1]  THEN
    PRINT j
    FIND_SOLUTION (P[j])
ELSE
    FIND_SOLUTION(j-1)
```