Problemi di ottimizzazione - selezione di attività

Contesto: una risorsa che viene usata da più utenti, come la gestiamo in modo efficiente? (la risorsa è mutualmente esclusiva)

$$A = \{e_1, e_2, e_3 \dots e_n\}$$

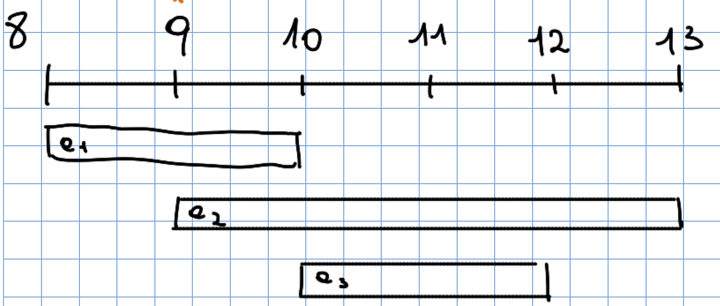
e_i

S_i = tempo di inizio

F_i = tempo di fine

1)

$\underset{S_i \; F_i}{\overset{e_1}{(8, 10)}}$ $\overset{e_2}{(9, 13)}$ $\overset{e_3}{(10, 12)}$



$S_i < F_i$

$e_i, e_J \in A$

sono compatibili se:

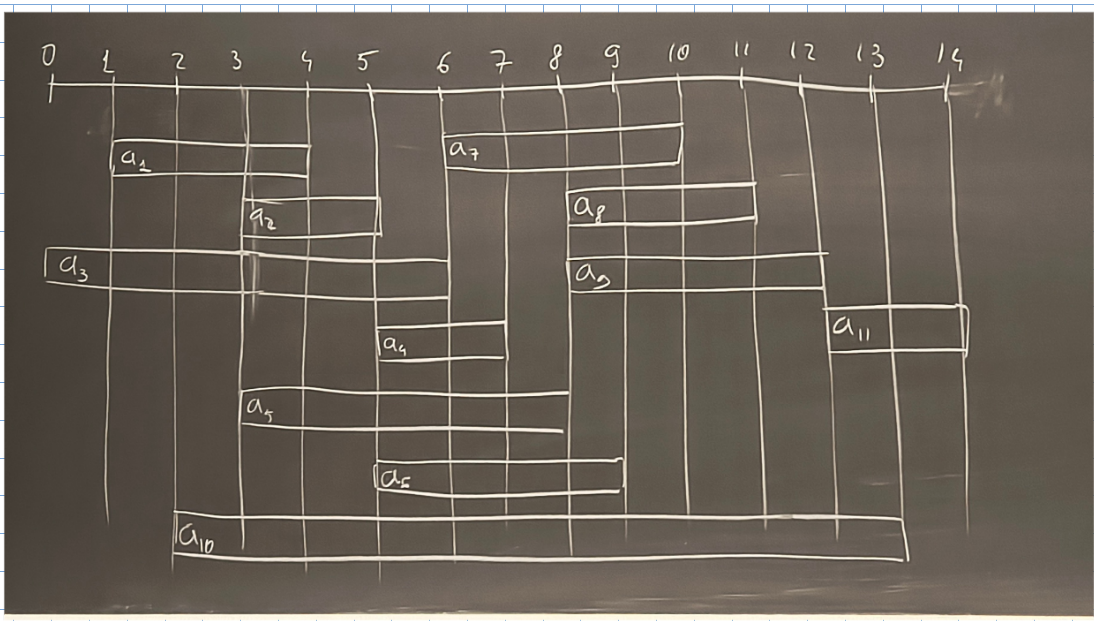
- $S_i < F_i \leq S_J < F$
- $S_J < F_J \leq S_i < F_i$

e_i e e_J sono compatibili se tra di loro non c'è un intersezione sul tempo

Dobbiamo ottimizzare il numero di utenti soddisfatti

Nel nostro caso specifico abbiamo 11 attività

i	1	2	3	4	5	6	7	8	9	10	11
S_i	1	3	0	5	3	5	6	8	8	2	12
F_i	4	5	6	7	8	9	10	11	12	13	14



soluzione
↑
$S \subseteq A$ → definito come

$\forall a_i, a_j \in S \Rightarrow \begin{cases} F_i \le S_j \\ F_j \le S_i \end{cases}$ → insieme delle attività che non si sovrappongono

$S' = \{a_2, a_4, a_9, a_{11}\}$

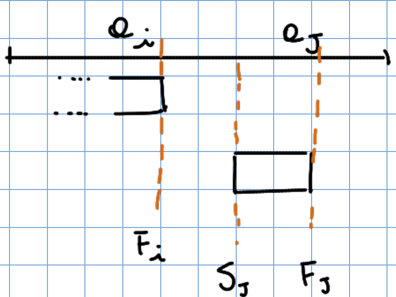
$S'' = \{a_7\}$

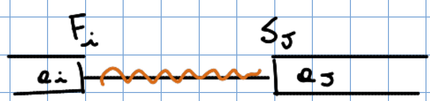
⋮ ci sono altre soluzioni

Dobbiamo trovare l'insieme S con la cardinalità massima

A_{ij}

$i < j$





↳ momento in cui la risorsa è libera → sottoproblema
ricorsione?

esempio

$A_{1,7} = \emptyset$

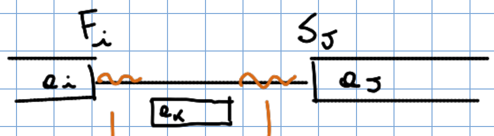
$A_{1,11} = \{a_4, a_5, a_7, a_8, a_9, a_6\}$ → soluzione del sottoproblema $A_{1,11}$

tutte le attività che si hanno tra quando a_1 inizia e a_{11} finisce

$A_{i,j} \to e_k \in A_{ij}$
$\uparrow$
$S_{i,j}$
↳ soluzione di A_{ij}

$A_{i,j} \to \underbrace{e_k \in A_{ij}, S_{ij}}$
$\uparrow$
$S_{i,j}$

se impongo che $e_k \in S_{ij}$ allora tutte le altre attività devono essere compatibili con e_k



lo spazio disponibili occasione di due sottoproblemi più piccolo

$$S_{ij} = S_{ik} \cup \{e_k\} \cup S_{kj}$$

$$S^*_{ij} = \underline{S^*_{ik}} \cup \{e_k\} \cup \underline{S^*_{kj}}$$

se questa è ottima anche queste lo sono?

Si! Di seguito la dimostrazione

$$S^*_{ij} = S_{ik} \cup \{e_k\} \cup S^*_{kj}$$

$$|S_{ij}| \leq |S^*_{ij}|$$

$$S_{ij} = S^*_{ik} \cup \{e_k\} \cup S^*_{kj}$$

$$|S^*_{ij}| = |S_{ik}| + |S^*_{kj}| + 1 \leq \underbrace{|S^*_{ik}| + |S^*_{kj}| + 1}_{= |S_{ij}|}$$

2) Soluzione ricorsiva per il costo ottimo

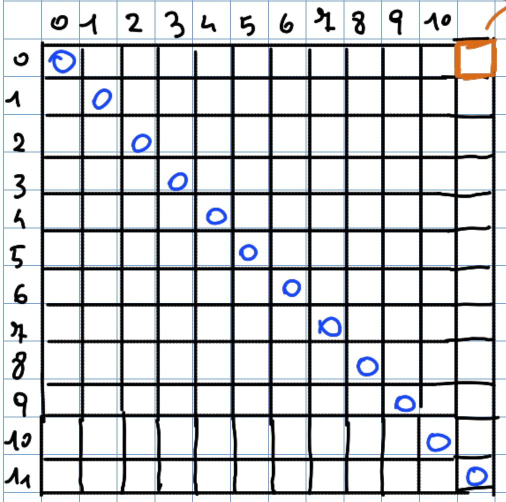
$$S(i,j) = \begin{cases} 0 & \text{se } A_{ij} = \emptyset \\ \underset{\substack{i<k<j \\ \text{con } e_k \in A_{ij}}}{\max} \left(S(i,k) + S(k,j) + 1 \right) & \text{se } |A_{ij}| > 0 \end{cases}$$

$O(n) \to$ max

Come determiniamo il k migliore? con il max

Soluzione Bottom-up iterativo

arrivati qui avremo una soluzione



$$|S_{ij}| = |S_{ik}| + |S_{kj}| + 1$$
$$i<k<j$$

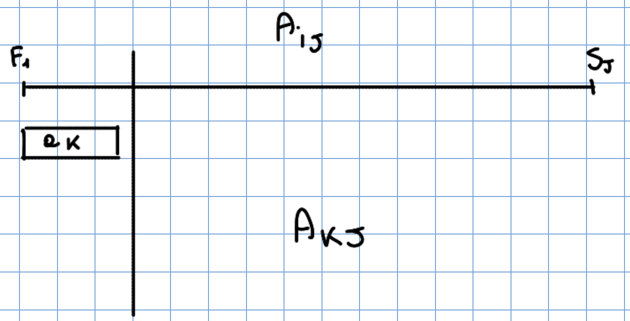
→ questo caso deve essere garantito e viene garantito risolvendolo per diagonali

Risolviamo noi il pomo 3-4 "Siamo a mare" !!

Arriveremo ad essere $O(n^3)$

Esiste un modo per trovare la soluzione senza iterare tutte le k?

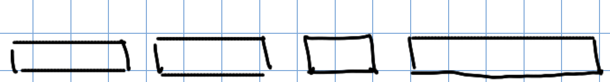
No nella maggior parte dei casi, come risolviamo in quei casi?



Dimostrazione della scelta greedy del tipo: $e_k \in A_{ij} : F_k = \min\{F_w \mid e_w \in A_{ij}\}$

S^*_{ij} una sol ottima per A_{ij} $e_k \in S^*_{ij}$?

lo dimostriamo sempre per contraddizione

$S^*_{ij} = \{$  $\}$ $e_\ell \in S^*_{ij} : F_\ell \min\{F_w : e_w \in S^*_{ij}\}$

$e_k, e_\ell \in A_{ij}$

$S^*_{ij} / \{e_f\} \cup \{e_k\} = S_{ij}$

levo e_f e metto e_k lo posso fare perché $F_k \leq F_f$

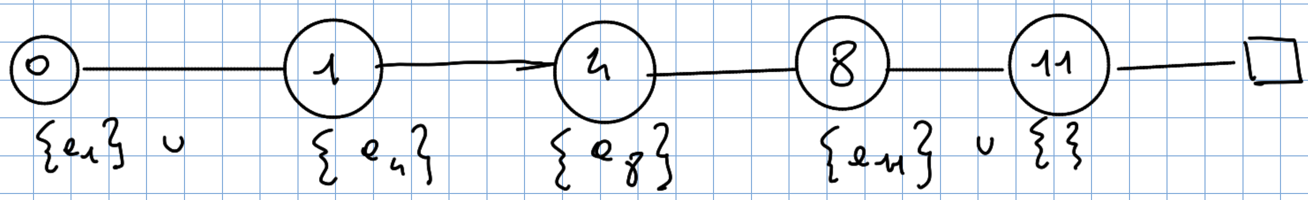
$|S_{ij}| = |S^*_{ij}| - 1 + 1 = |S^*_{ij}|$

Prendo una soluzione ottima costruendo intorno ad e_f

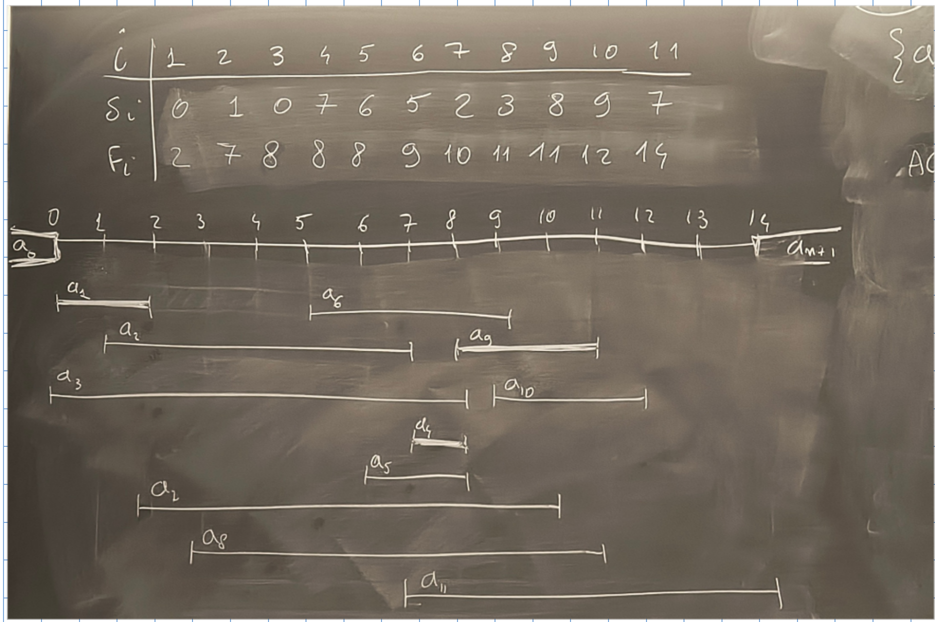
$$S(i,j) = \begin{cases} 0 & \text{se } |A_{ij}| = 0 \\ 1 + S(k,j),\ F_k = \min\{F_w : e_w \in A_{ij}\} & \text{se } |A_{ij}| > 0 \end{cases}$$

```
Activity-Selector(A, i, ~~j~~, n)
  m ← i + 1                         e_m ∈ A_ij ?
  while m ≤ n Do
      if (S_m ≥ F_i) then
          return {e_m} ∪ AS(A, m, n)
  return { }
```

```
Activity-Selector(A, n)             e_m ∈ A_i,m+1
  S = {e_1}
  i ← 1
  FOR m ← 2 to n Do                 O(n)
      if (S_m ≥ F_i) then
          S = S ∪ {e_m}
          i ← m
```



Esempio con altri numeri



Cambia il problema: Dobbiamo occupare le risorse per più tempo possibile

$$S(i,j) = S(i,k) + S(k,j) + (F_k - S_k)$$

Si dimostra come prima, per contraddizione

$$S_{i,j} = \begin{cases} 0 & |A_{ij}| = \emptyset \\ \max\limits_{\substack{i<k<j \\ \text{con } a_k \in A_{ij}}} & (S(i,k) + S(k,j) + (F_k - S_k)) \end{cases}$$

Troviamo un algoritmo greedy? Non esiste, devo usare l'algoritmo $O(n^3)$