ALGORITMI GREEDY

- SI TRATTA DI ALGORITMI PER PROBLEMI DI OTTIMIZZAZIONE IN GRADO DI COSTRUIRE UNA SOLUZIONE OTTIMA ATTRAVERSO UNA SEQUENZA "TOP-DOWN" DI SCELTE LOCALMENTE OTTIME

- NON SEMPRE UNA STRATEGIA GREEDY PORTA AD UNA SOLUZIONE OTTIMA (ES. PROBLEMA DELLO ZAINO 0-1)

PROBLEMA DELLO ZAINO (VERSIONE INTERA)
(KNAPSACK PROBLEM)

CAPIENZA DELLO ZAINO $W \ (\in \mathbb{N})$

OGGETTO	PESO	VALORE
1	$w_1 > 0$	v_1
2	$w_2 > 0$	v_2
$\vdots$	$\vdots$	$\vdots$
n	$w_n > 0$	v_n

$A \subseteq \{1, 2, \dots, n\}$

$$\text{weight}(A) = \sum_{i \in A} w_i$$

$$\text{value}(A) = \sum_{i \in A} v_i$$

PROBLEMA DET. $S \subseteq \{1, 2, \dots, n\}$ TALE CHE

- $\text{weight}(S) \leq W$
- $\text{value}(S)$ SIA MASSIMO

PROBLEMA DELLO ZAINO (VERSIONE FRAZIONARIA)

(KNAPSACK PROBLEM)

CAPIENZA DELLO ZAINO $W \ (\in \mathbb{N})$

OGGETTO	PESO	VALORE
1	$w_1 > 0$	v_1
2	$w_2 > 0$	v_2
$\vdots$	$\vdots$	$\vdots$
n	$w_n > 0$	v_n

$A : \{1, 2, \ldots, n\} \longrightarrow [0,1]$

$$\mathrm{weight}(A) = \sum_{i=1}^{n} A(i) \cdot w_i$$

$$\mathrm{value}(A) = \sum_{i=1}^{n} A(i) \cdot v_i$$

PROBLEMA DET. $S : \{1, 2, \ldots, n\} \to [0,1]$ TALE CHE

- $\mathrm{weight}(S) \leq W$
- $\mathrm{value}(S)$ SIA MASSIMO

- ENTRAMBI I PROBLEMI GODONO DELLA PROPRIETÀ DELLA SOTTOSTRUTTURA OTTIMA

- NEL CASO FRAZIONARIO È POSSIBILE UTILIZZARE LA SEGUENTE STRATEGIA GREEDY CONSISTENTE NEL SELEZIONARE LA MASSIMA QUANTITÀ DEL MATERIALE AVENTE VALORE SPECIFICO MASSIMO, COMPATIBILMENTE CON LA CAPIENZA RESIDUA DELLO ZAINO

- TALE STRATEGIA NON FUNZIONA PERÒ PER LA VARIANTE INTERA CHE DIMOSTRA UNA BEN MAGGIORE COMPLESSITÀ COMBINATORIA

OGGETTO	PESO	VALORE	VALORE / PESO
1	10	60	6
2	20	100	5
3	30	120	4

$\boxed{W = 50}$

CASO FRAZIONARIO

$A(1) = 1$ $\quad$ $value(A) = 1 \cdot 60 + 1 \cdot 100 + \frac{2}{3} \cdot 120 = 240$

$A(2) = 1$

$A(3) = \frac{2}{3}$

CASO INTERO

$A = \{1, 2\}$ $\quad$ $value(A) = 60 + 100 = 160$

$A' = \{2, 3\}$ $\quad$ $value(A') = 100 + 120 = 220$ $\quad$ < 240

FIN QUI (16/11/21)

UN PROBLEMA DI SELEZIONE DI ATTIVITÀ

SISTEMA DI ATTIVITÀ (S, s, f)

$S = \{a_1, a_2, \dots, a_n\}$ - INSIEME DI ATTIVITÀ IN COMPETIZIONE PER L'USO ESCLUSIVO DI UNA RISORSA

$s : \{1, \dots, n\} \to \mathbb{R}^+$ - $(s_i =_{df} s(i))$ INIZIO ATTIVITÀ

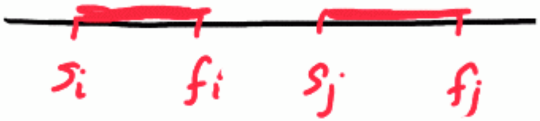
$f : \{1, \dots, n\} \to \mathbb{R}^+$ - $(f_i =_{df} f(i))$ FINE ATTIVITÀ

TALI CHE $s_i < f_i$, PER $i = 1, \dots, n$

$[s_i, f_i[$ - INTERVALLO TEMPORALE PER CUI L'ATTIVITÀ a_i RICHIEDE L'USO ESCLUSIVO DELLA RISORSA

- DUE ATTIVITÀ a_i E a_j SONO COMPATIBILI SE

$[s_i, f_i[\cap [s_j, f_j[= \emptyset$, CIOÈ



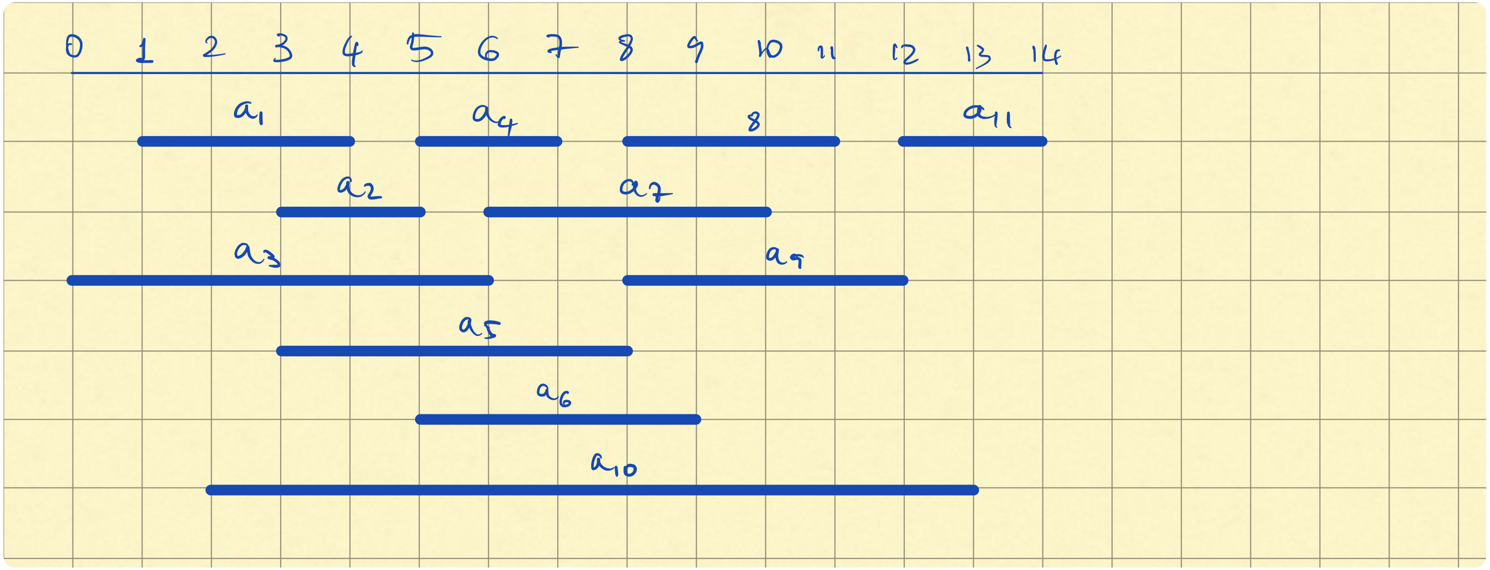
OPPURE

$\boxed{f_i \leq s_j}$ $\boxed{f_j \leq s_i}$

- UN SOTTOINSIEME $A \subseteq S$ È UN INSIEME DI ATTIVITÀ MUTUAMENTE COMPATIBILI SE OGNI COPPIA DI ATTIVITÀ DISTINTE a_i, a_j IN A È COSTITUITA DA ATTIVITÀ COMPATIBILI

- DATO UN SISTEMA DI ATTIVITÀ (S, s, f), IL PROBLEMA DELLA SELEZIONE DELLE ATTIVITÀ CONSISTE NEL DETERMINARE UN SOTTOINSIEME DI CARDINALITÀ MASSIMA $A \subseteq S$ DI ATTIVITÀ MUTUAMENTE COMPATIBILI

- DATO UN SISTEMA DI ATTIVITÀ (S, s, f), IL PROBLEMA DELLA SELEZIONE DELLE ATTIVITÀ CONSISTE NEL DETERMINARE UN SOTTOINSIEME DI CARDINALITÀ MASSIMA $A \subseteq S$ DI ATTIVITÀ MUTUAMENTE COMPATIBILI



ES.

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

- a_1 E a_2 NON SONO COMPATIBILI
- $\{a_3, a_9, a_{11}\}$ E' UN INSIEME DI ATTIVITA' MUTUAMENTE COMPATIBILI
- $\{a_1, a_4, a_8, a_{11}\}$
  $\{a_2, a_4, a_9, a_{11}\}$ SONO INSIEMI DI ATTIVITA' MUTUALMENTE COMPATIBILI DI CARDINALITA' MASSIMA

- SOLUZIONE MEDIANTE RICERCA ESAUSTIVA

- SI GENERINO TUTTI I POSSIBILI SOTTOINSIEMI $A \subseteq S$

  $\Omega(2^n)$

- SI VERIFICHI PER CIASCUNO DI ESSI SE SI TRATTA DI UN INSIEME DI ATTIVITÀ MUTUAMENTE COMPATIBILI

  $O(n^2)$

- SI DETERMINI IL SOTTOINSIEME $A \subseteq S$ DI ATTIVITÀ MUTUAMENTE COMPATIBILI DI CARDINALITÀ MASSIMA

COMPLESSITÀ: $\Omega(2^n)$

STUDIO DI UNA SOLUZIONE OTTIMA

- SIA $A \subseteq S$ UNA SOLUZIONE OTTIMA
- SIA $a_i \in A$ (SPESSO SCRIVEREMO $i \in A$)
- a_i INDUCE I DUE SOTTOPROBLEMI

$$S_i^- = \{ k \in S : f_k \leq s_i \}$$

$$S_i^+ = \{ k \in S : s_k \geq f_i \}$$

- E' IMMEDIATO VERIFICARE CHE

$A \cap S_i^-$ E' UNA SOLUZIONE OTTIMA PER S_i^-

$A \cap S_i^+$ E' UNA SOLUZIONE OTTIMA PER S_i^+

CIOE', VALE LA PROPRIETA' DELLA SOTTOSTRUTTURA OTTIMA

STUDIO DELLO SPAZIO DEI SOTTOPROBLEMI

$$j \in S_i^- \begin{cases} (S_i^-)_j^- = S_j^- \\ (S_i^-)_j^+ = S_{ji} = \{k \in S : f_j \le s_k < f_k \le s_i\} \end{cases}$$

$$j \in S_i^+ \begin{cases} (S_i^+)_j^+ = S_j^+ \\ (S_i^+)_j^- = S_{ij} = \{k \in S : f_i \le s_k < f_k \le s_j\} \end{cases}$$

- INTRODUCENDO DUE NUOVE ATTIVITA' DI COMODO a_0, a_{n+1} CARATTERIZZATE DA $f_0 = 0$ E $s_{n+1} > \max f_k$, POSSIAMO SCRIVERE: $S_i^- = S_{0i}$, $S_i^+ = S_{i,n+1}$

- E' FACILE ALLORA VERIFICARE CHE LO SPAZIO DEI SOTTOPROBLEMI RILEVANTI PER IL PROBLEMA DELLE ATTIVITÀ È: $\{ S_{ij} : 0 \leq i,j \leq n+1 \}$

- INDICHIAMO CON $c[i,j]$ LA CARDINALITÀ DI UNA SOLUZIONE OTTIMA AL PROBLEMA S_{ij}. SI HA:

$$c[i,j] = \begin{cases} 0 & \text{SE } S_{ij} = \emptyset \\ \max\limits_{k \in S_{ij}} \left(c[i,k] + c[k,j] + 1 \right) & \text{SE } S_{ij} \neq \emptyset \end{cases}$$

(TALE RICORRENZA PUÒ ESSERE UTILIZZATA BOTTOM-UP SEGUENDO L'ORDINAMENTO CRESCENTE DELLE CARDINALITÀ DEGLI INSIEMI S_{ij})

- NELL'IPOTESI $f_1 \le f_2 \le \ldots \le f_n$, LE CARDINALITA' $c[i,j]$ POSSONO ESSERE CALCOLATE A PARTIRE DALLE COPPIE $(i, i+1)$ PROCEDENDO PER VALORI DI $j-i$ NON DECRESCENTI

COMPLESSITA': $O(n^3)$

- SIN QUI NIENTE DI NUOVO!
- E' POSSIBILE CONVERTIRE LA PRECEDENTE SOLUZIONE IN UNA SOLUZIONE "GREEDY"?

- SI RICONSIDERI LA PRECEDENTE RICORRENZA:

$$c[i,j] = \begin{cases} 0 & \text{SE} \quad S_{ij} = \emptyset \\ \max\limits_{k \in S_{ij}} (c[i,k] + c[k,j] + 1) & \text{SE} \quad S_{ij} \neq \emptyset \end{cases}$$

- PER APPLICARLA, AD OGNI PASSO OCCORRE "INDOVINARE" LA SCELTA GIUSTA DI k TRA AL PIÙ $j-i-1$ VALORI E QUINDI RISOLVERE DUE SOTTOPROBLEMI.
- È POSSIBILE "INDOVINARE" k DIRETTAMENTE E RIDURRE I SOTTOPROBLEMI AD UNO SOLO?

(SCELTA GREEDY)

- DATO $S_{ij} \neq \emptyset$, SI PONGA $\bar{k} = \min S_{ij}$
- CHIARAMENTE $S_{i\bar{k}} = \emptyset$, DA CUI $c[i, \bar{k}] = 0$

  (INFATTI, SE $\ell \in S_{i\bar{k}} \rightarrow f_i \leq s_\ell < f_\ell \leq s_{\bar{k}} < f_{\bar{k}} \leq s_j$

  $\rightarrow \ell \in S_{ij}$ E $\ell < \bar{k}$, ASSURDO)
- QUINDI IN CORRISPONDENZA DI $\bar{k}$ C'E' DA RISOLVERE UN SOLO SOTTOPROBLEMA.
- VALE: $c[i,j] = c[\bar{k}, j] + 1$?

A_{ij} soluz. ott. di S_{ij}

$A'_{ij} = (A_{ij} \setminus \{\min A_{ij}\}) \cup \{\bar{k}\}$

- SIA A_{ij} UNA SOLUZIONE OTTIMA PER S_{ij}.
  ALLORA $A'_{ij} = (A_{ij} \setminus \{\min A_{ij}\}) \cup \{\bar{k}\}$ E' UNA SOLUZIONE OTTIMA PER S_{ij}
- INFATTI, SE $\min A_{ij} = \bar{k}$, ALLORA $A'_{ij} = A_{ij}$ E' OTTIMA
- SE $\min A_{ij} > \bar{k}$, POICHE' $|A'_{ij}| = |A_{ij}|$, E' SUFFICIENTE VERIFICARE CHE TUTTE LE ATTIVITA' IN $A_{ij} \setminus \{\min A_{ij}\}$ SONO COMPATIBILI CON $\bar{k}$.
- SIA $\ell \in A_{ij} \setminus \{\min A_{ij}\}$. PONIAMO $m = \min A_{ij}$, ALLORA $f_m \leq s_\ell$. MA $f_{\bar{k}} \leq f_m$, QUINDI $f_{\bar{k}} \leq s_\ell$, CIOE' $\bar{k}$ ED ℓ SONO COMPATIBILI.

PERTANTO

- $A'_{ij} \setminus \{\bar{k}\}$ E' UNA SOLUZIONE OTTIMA PER $S_{\bar{k}j}$, QUINDI

  $|A'_{ij} \setminus \{\bar{k}\}| = c[\bar{k},j]$

- POICHE' A'_{ij} E' UNA SOLUZIONE OTTIMA PER S_{ij} SI HA $|A'_{ij}| = c[i,j]$, DA CUI:

  $c[i,j] = c[\bar{k},j] + 1$

- LA SCELTA $\bar{k} = \min S_{ij}$ E' GREEDY RELATIVAMENTE ALLA SEGUENTE INTUIZIONE:

  "SCEGLIENDO TRA TUTTE LE ATTIVITA' COMPATIBILI QUELLA CHE TERMINA PRIMA, SI MASSIMIZZA LA DISPONIBILITA' DELLA RISORSA PER LE RIMANENTI ATTIVITA'."

- SI OSSERVI CHE PER SELEZIONARE $\bar{k} = \min S_{ij}$ NON E' NECESSARIO AVERE RISOLTO PRECEDENTEMENTE IL PROBLEMA $S_{\bar{k}j}$.
- QUINDI UNA SOLUZIONE OTTIMA PUÒ ESSERE COSTRUITA IN MANIERA TOP-DOWN

```
RECURSIVE_ACTIVITY_SELECTOR (s, f, i, j)
   FOR k := i+1 TO j-1 DO
      IF f_i ≤ s_k < f_k ≤ s_j THEN
         RETURN {k} ∪ RECURSIVE_ACTIVITY_SELECTOR (s, f, k, j)
   RETURN ∅
```

- POICHE' TUTTE LE CHIAMATE A R_A_S SONO DEL TIPO R_A_S(S, s, f, i, $n+1$), ESSA PUÒ ESSERE SEMPLIFICATA COSÌ:

```
RECURSIVE_ACTIVITY_SELECTOR (s, f, i)
   n := |s|
   FOR k := i+1 TO n DO
      IF f_i ≤ s_k THEN
         RETURN {k} ∪ RECURSIVE_ACTIVITY_SELECTOR (s, f, k)
   RETURN ∅
```

- OGNI ATTIVITA' VIENE CONSIDERATA ESATTAMENTE UNA VOLTA, QUINDI R_A_S E' LINEARE

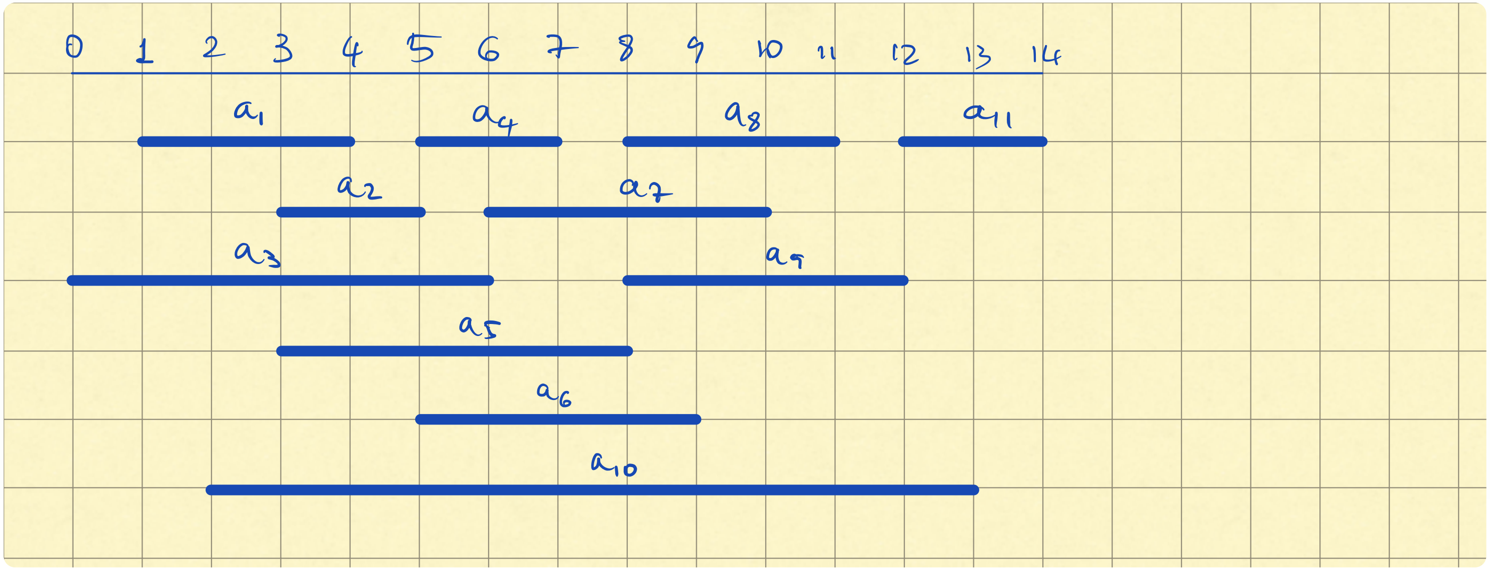
- LA "QUASI" RICORSIONE DI CODA DI R_A_S PUO' ESSERE FACILMENTE ELIMINATA, DANDO LUOGO AL SEGUENTE ALGORITMO ITERATIVO:

```
GREEDY_ACTIVITY_SELECTOR (s,f)
  n := |s|
  A := {1}
  i := 1
  for m := 2 to n do
      if s_m ≥ f_i then
          A := A ∪ {m}
          i := m
  return A
```

COMPLESSITA': $O(n)$
(A MENO DELL'ORDINAMENTO DI S)

ES.

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



ESERCIZI

1) CHE COSA SUCCEDE SE ANZICHÈ SELEZIONARE L'ATTIVITÀ CHE TERMINA PRIMA, SI SELEZIONA L'ATTIVITÀ CHE INIZIA PIÙ TARDI?

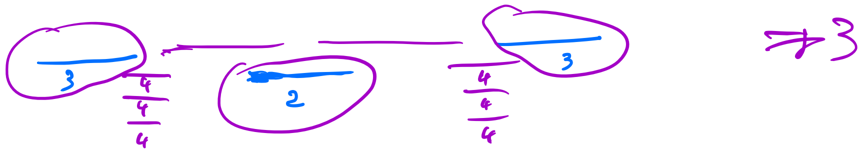
2) VERIFICARE CHE LE SEGUENTI SCELTE "GREEDY" NON FUNZIONANO PER IL PROBLEMA DELLA SELEZIONE DELLE ATTIVITÀ:

"TRA TUTTE LE ATTIVITÀ COMPATIBILI

a) SI SELEZIONI L'ATTIVITÀ DI DURATA MINIMA

b) SI SELEZIONI L'ATTIVITÀ CHE INTERSECA IL MINOR NUMERO DI ATTIVITÀ

c) SI SELEZIONI L'ATTIVITÀ CHE INIZIA PRIMA"


3
4
4
4
2
4
4
4
3
⇏ 3

- RIASSUMENDO, LA STRATEGIA GREEDY CONSISTE NEI SEGUENTI PASSI:

1. VERIFICARE LA PROPRIETÀ DELLA SOTTOSTRUTTURA OTTIMA

2. VERIFICARE CHE ESISTE SEMPRE UNA SOLUZIONE OTTIMA CHE INCLUDE LA SCELTA GREEDY

3. VERIFICARE CHE DOPO LA SCELTA GREEDY IL PROBLEMA INIZIALE È RICONDOTTO AD UN SOTTOPROBLEMA DELLO STESSO TIPO LA CUI SOLUZIONE OTTIMA PUÒ ESSERE COMBINATA CON LA SCELTA GREEDY PER DARE LUOGO AD UNA SOLUZIONE OTTIMA DEL PROBLEMA INIZIALE

PROGRAMMAZIONE DINAMICA E STRATEGIA GREEDY

- TALI METODOLOGIE UTILIZZANO ENTRAMBE LA PROPRIETÀ DELLA SOTTOSTRUTTURA OTTIMA
- E' QUINDI POSSIBILE CHE:
  - SI UTILIZZI LA PROGRAMMAZIONE DINAMICA QUANDO E' SUFFICIENTE UNA PIÙ EFFICIENTE STRATEGIA GREEDY
  - PER ERRORE SI DIA UNA SOLUZIONE BASATA SULLA STRATEGIA GREEDY QUANDO INVECE E' NECESSARIO UTILIZZARE LA PROGRAMMAZIONE DINAMICA