Algoritmi e Strutture Dati

Algoritmi Golosi (Greedy)

Maria Rita Di Berardini, Emanuela Merelli[1]

[1]Dipartimento di Matematica e Informatica
Università di Camerino

A.A. 2007/08

Problema della selezione di attività

Problema: programmazione di più attività in competizione che richiedono l'uso esclusivo di una risorsa comune

Sia $S = \{a_1, a_2, \cdots, a_n\}$ un insieme di n attività che devono usare in maniera mutuamente esclusiva una data risorsa

Ogni attività a_i ha un tempo di inizio s_i ed un tempo di fine f_i, con $0 \leq s_i < f_i < \infty$, e si svolge nell'intervallo aperto $[s_i, f_i)$

Due attività a_i e a_j si dicono compatibili se gli intervalli $[s_i, f_i)$ e $[s_j, f_j)$ non si sovrappongono (e quindi se $s_i \geq f_j$ o $s_j \geq f_i$)

Obiettivo: selezionare il più grande sottoinsieme di attività mutuamente compatibili

Problema della selezione di attività: un esempio

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

$\{a_3, a_9, a_{11}\}$ è formato da attività mutuamente compatibili

$\{a_1, a_4, a_8, a_{11}\}$ hanno cardinalità massima

$\{a_2, a_4, a_9, a_{11}\}$

Sottostruttura ottima della selezione di attività

- Il problema della selezione di attività ha una sottostruttura ottima
- Può essere risolto con un algoritmo di PD
- Soddisfa anche la proprietà della scelta golosa: può essere risolto con un algoritmo goloso
- Iniziamo con il dimostrare la sottostruttura ottima

Sottostruttura ottima della selezione di attività

Per dimostrare la sottostruttura ottima occorre definire un appropriato spazio di sottoproblemi

Sia $S_{ij} = \{a_k \in S \mid f_i \leq s_k < f_k \leq s_j\}$ il sottoinsieme delle attività che iniziano dopo la fine di a_i (i.e. tali che $f_i \leq s_k$) e finisco prima dell'inizio di a_j(i.e. tali che $f_k \leq s_j$)

Per rappresentare tutto l'insieme S introduciamo due attività fittizie:

- a_0 con tempo di fine $f_0 = 0$
- a_{n+1} con tempo di inizio $s_{n+1} = \infty$

Allora $S = S_{0\,n+1}$

Assumiamo, inoltre, che le attività in S siano ordinate in modo crescente rispetto ai tempi di fine e quindi che $f_0 \leq f_1 \leq f_2 \leq ... \leq f_n < f_{n+1}$

Sottostruttura ottima della selezione di attività

Assumiamo $S_{ij} \neq \emptyset$ (se $S_{ij} = \emptyset$ il problema è banale) e che una soluzione ottima A_{ij} per S_{ij} contenga l'attività a_y (a_y è tale che $f_i \leq s_y < f_y \leq s_j$)

Possiamo utilizzare a_y per individuare due sottoproblemi:

- S_{iy} (iniziano dopo la fine di a_i e finiscono prima dell'inizio di a_y)
- S_{yj} (iniziano dopo la fine di a_y e finiscono prima dell'inizio di a_j)

tali che $S_{ij} = S_{iy} \cup \{a_y\} \cup S_{yj}$ (esercizio); inoltre:

$$A_{ij} = A_{iy} \cup \{a_y\} \cup A_{yj}$$

dove A_{iy} e A_{yj} sono le soluzioni per S_{iy} e S_{yj} contenute all'interno della soluzione ottima A_{ij} per S_{ij}

Sottostruttura ottima della selezione di attività

Assumiamo, per assurdo, che A_{iy} non sia ottima

Questo significa che esiste un insieme di attività mutuamente compatibili in S_{iy} più grande di A_{iy}, ossia che esiste un insieme A'_{iy} di attività mutuamente compatibili in S_{iy} tale che $|A'_{iy}| > |A_{iy}|$

Allora $A'_{ij} = A'_{iy} \cup \{a_y\} \cup A_{yj}$ è una soluzione migliore della soluzione ottima A_{ij}, il che è chiaramente una contraddizione

In maniera simile possiamo dimostrare che A_{yk} è ottima

Sottostruttura ottima della selezione di attività

Questo dimostra che il problema della selezione di attività soddisfa la sottostruttura ottima e, quindi, può essere risolto mediante un algoritmo di programmazione dinamica (un algoritmo abbastanza simile a quello per il prodotto di una sequenza di matrici)

Problema della selezione di attività

Un algoritmo goloso (analogamente agli algoritmi di PD) costruisce la soluzione ottima effettuando una serie di scelte

A differenza di un algoritmo di PD, la scelta effettuata da un algoritmo goloso non dipende dalle soluzioni dei sottoproblemi

Un algoritmo goloso fa sempre la scelta che sembra ottima in un dato momento, una scelta localmente ottima, nella speranza che tale scelta porterà ad una soluzione globalmente ottima

Gli algoritmi golosi sono, in generale più semplici e meno onerosi rispetto a quelli di PD

Il problema è che non sempre una "tecnica golosa" è in grado di costruire una soluzione ottima per un dato problema

Problema della selezione di attività

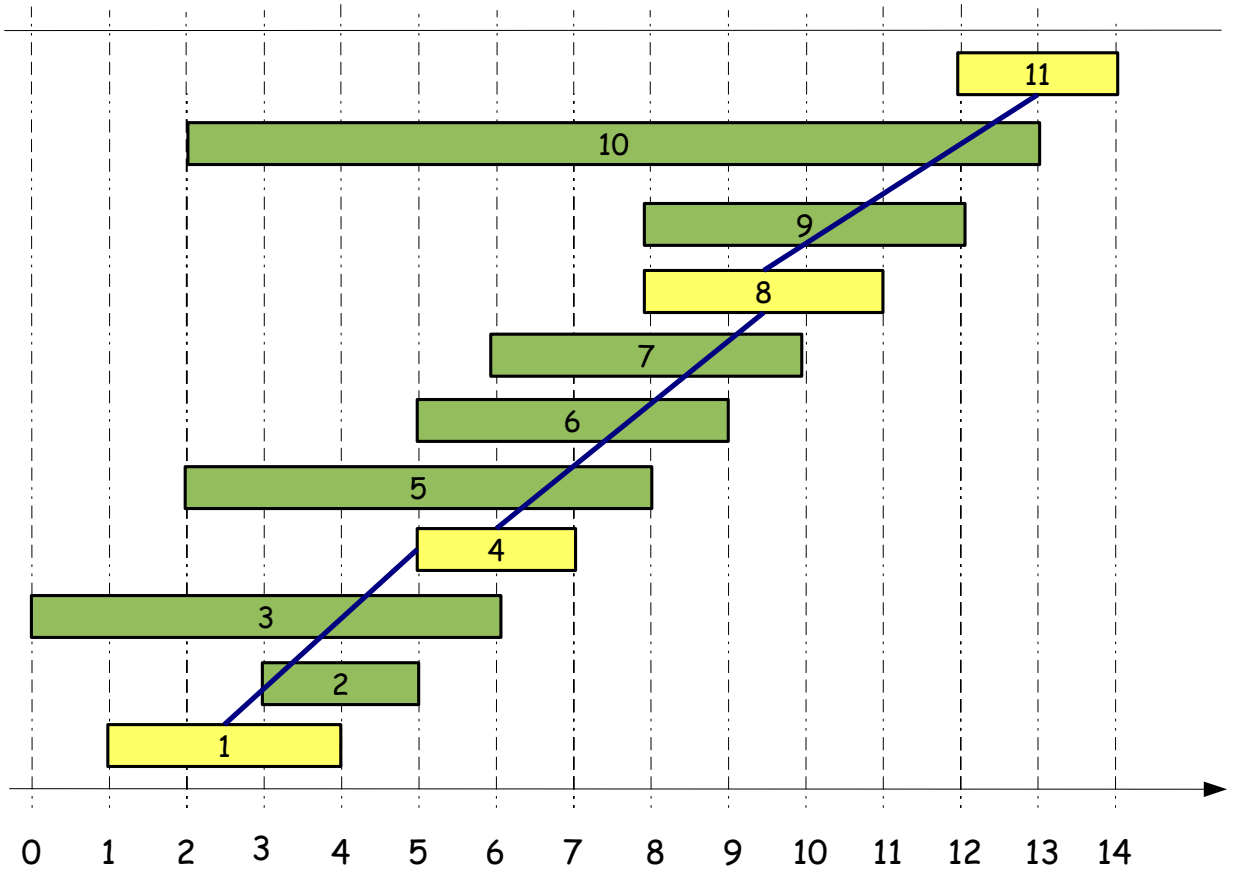
Per il problema della selezione di attività, effettuare la scelta "localmente ottima" significa scegliere l'attività che, in un dato momento, "occupa la risorsa per il minor tempo possibile" e quindi quella con minor tempo di fine

Se le attività in S sono ordinate in maniera crescente rispetto ai tempi di fine, lo algoritmo goloso sceglierà:

1. innanzitutto l'attività a_1, quella che minor tempo di fine
2. poi la prima (e quindi quella più breve) attività compatibile con a_1
3. la prima attività compatibile con la seconda attività scelta
4. e così via

Problema della selezione di attività

Le attività sono ordinate in maniera crescente rispetto a f_i



Algoritmo greedy per la selezione di attività

GreedyActivitySelector(s, f)

1. $n \leftarrow length[s]$
2. $A \leftarrow \{a_1\}$
3. $i \leftarrow 1$
4. **for** $m \leftarrow 2$ **to** n
5. **do if** $s_m \geq f_i$
6. **then** $A \leftarrow A \cup \{a_m\}$
7. $i \leftarrow m$
8. **return** A

Ad ogni passo l'algoritmo inserisce in A la prima attività compatibile con a_i, quella localmente ottima (N.B: inizialmente $i = 1$)

Questa serie di scelte localmente ottime porta ad una soluzione che è globalmente ottima. Perchè?

Correttezza dell'algoritmo goloso

Assumiamo di dover risolvere il sottoproblema S_{ij} e che a_m sia l'attività in S_{ij} con il minor tempo di fine, ossia

$$a_m \in S_{ij} \text{ tale che } f_m \leq f_k \text{ per ogni } a_k \in S_{ij}$$

a_m rappresenta la scelta golosa, dalla quale vengono originati due sottoproblemi S_{im} e S_{mj}. Solo uno dei due (S_{mj}) è non vuoto

Assumiamo, per assurdo, che $a_k \in S_{im} \neq \emptyset$. Allora:

- $a_k \in S_{im}$ implica $f_i \leq s_k < f_k \leq s_m$

Correttezza dell'algoritmo goloso

Assumiamo di dover risolvere il sottoproblema S_{ij} e che a_m sia l'attività in S_{ij} con il minor tempo di fine, ossia

$$a_m \in S_{ij} \text{ tale che } f_m \leq f_k \text{ per ogni } a_k \in S_{ij}$$

a_m rappresenta la scelta golosa, dalla quale vengono originati due sottoproblemi S_{im} e S_{mj}. Solo uno dei due (S_{mj}) è non vuoto

Assumiamo, per assurdo, che $a_k \in S_{im} \neq \emptyset$. Allora:

- $a_k \in S_{im}$ implica $f_i \leq s_k < f_k \leq s_m$
- poichè $s_m < f_m$, abbiamo $f_k < f_m$ (contraddice il fatto che a_m è l'attività in S_{ij} con minor tempo di fine)

Correttezza dell'algoritmo goloso

Assumiamo di dover risolvere il sottoproblema S_{ij} e che a_m sia l'attività in S_{ij} con il minor tempo di fine, ossia

$$a_m \in S_{ij} \text{ tale che } f_m \leq f_k \text{ per ogni } a_k \in S_{ij}$$

a_m rappresenta la scelta golosa, dalla quale vengono originati due sottoproblemi S_{im} e S_{mj}. Solo uno dei due (S_{mj}) è non vuoto

Assumiamo, per assurdo, che $a_k \in S_{im} \neq \emptyset$. Allora:

- $a_k \in S_{im}$ implica $f_i \leq s_k < f_k \leq s_m$
- poichè $s_m < f_m$, abbiamo $f_k < f_m$ (contraddice il fatto che a_m è l'attività in S_{ij} con minor tempo di fine)

Possiamo, quindi, concludere che S_{im} è vuoto

Correttezza dell'algoritmo goloso

Assumiamo di dover risolvere il sottoproblema S_{ij} e che a_m sia l'attività in S_{ij} con il minor tempo di fine, ossia

$$a_m \in S_{ij} \text{ tale che } f_m \leq f_k \text{ per ogni } a_k \in S_{ij}$$

a_m rappresenta la scelta golosa, dalla quale vengono originati due sottoproblemi S_{im} e S_{mj}. Solo uno dei due (S_{mj}) è non vuoto

Assumiamo, per assurdo, che $a_k \in S_{im} \neq \emptyset$. Allora:

- $a_k \in S_{im}$ implica $f_i \leq s_k < f_k \leq s_m$
- poichè $s_m < f_m$, abbiamo $f_k < f_m$ (contraddice il fatto che a_m è l'attività in S_{ij} con minor tempo di fine)

Possiamo, quindi, concludere che S_{im} è vuoto
Fatta la scelta golosa, risolviamo solo S_{mj} (l'unico sottoproblema $\neq \emptyset$);
S_{mj} viene risolto ricorsivamente, di nuovo, tramite una scelta golosa

Correttezza dell'algoritmo goloso

Per dimostrare la correttezza dell'algoritmo greedy dimostriamo che:

per ogni $S_{ij} \neq \emptyset$, esiste **sempre** una soluzione ottima di S_{ij} contenente la scelta golosa, ossia l'attività $a_m \in S_{ij}$ con minor tempo di fine

Hp: le attività in S_{ij} sono memorizzate in ordine crescente rispetto ai tempi di fine

Sia A una soluzione ottima per $S_{ij} \neq \emptyset$ e sia $a_m \in S_{ij}$ con minor tempo di fine. Abbiamo due possibilità:

- $a_m \in A$ – prova terminata
- $a_m \notin A$: forniamo un modo per costruire (a partire da A) una soluzione ottima A' tale che $a_m \in A'$

Correttezza dell'algoritmo goloso

Sia A una soluzione ottima per $S_{ij} \neq \emptyset$ e sia $a_m \in S_{ij}$ con minor tempo di fine. Assumiamo, inoltre, $a_m \notin A$

Sia a_k l'attività in A con il minor tempo di fine; a_k è tale che $f_k \leq f_j$ per ogni $a_j \in A$

Alcune proprietà di a_k:

1. $s_k < f_k \leq f_j$ e, quindi, $s_k < f_j$ per ogni $a_j \in A$
2. a_k è compatibile con ogni altra attività $a_j \in A$. Poichè $s_k < f_j$, deve essere $s_j \geq f_k$

Consideriamo ora il sottoinsieme di attività $A' = (A - \{a_k\}) \cup \{a_m\}$ e dimostriamo che A' è una soluzione **ammissibile** (contiene solo attività mutuamente compatibili)

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

- Innanzitutto, $a_j \in A'$ e $a_k \notin A'$ implica $a_j \neq a_k$

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

- Innanzitutto, $a_j \in A'$ e $a_k \notin A'$ implica $a_j \neq a_k$
- Quindi $a_j \in A - \{a_m, a_k\}$ e, per le proprietà di a_k, $s_j \geq f_k$.

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

- Innanzitutto, $a_j \in A'$ e $a_k \notin A'$ implica $a_j \neq a_k$
- Quindi $a_j \in A - \{a_m, a_k\}$ e, per le proprietà di a_k, $s_j \geq f_k$.
- Poichè $A \subseteq S_{ij}$ e $a_m \in S_{ij}$ con minor tempo di fine, $f_k \geq f_m$

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

- Innanzitutto, $a_j \in A'$ e $a_k \notin A'$ implica $a_j \neq a_k$
- Quindi $a_j \in A - \{a_m, a_k\}$ e, per le proprietà di a_k, $s_j \geq f_k$.
- Poichè $A \subseteq S_{ij}$ e $a_m \in S_{ij}$ con minor tempo di fine, $f_k \geq f_m$
- Allora, $s_j \geq f_m$ ed a_j è compatibile con a_m

Correttezza dell'algoritmo goloso

Tutte le attività in A' diverse da a_m sono mutuamente compatibili perchè lo erano in A

Dimostriamo che a_m è compatibile con ogni altra attività $a_j \in A'$. Sia quindi $a_j \in A'$ con $a_j \neq a_m$

- Innanzitutto, $a_j \in A'$ e $a_k \notin A'$ implica $a_j \neq a_k$
- Quindi $a_j \in A - \{a_m, a_k\}$ e, per le proprietà di a_k, $s_j \geq f_k$.
- Poichè $A \subseteq S_{ij}$ e $a_m \in S_{ij}$ con minor tempo di fine, $f_k \geq f_m$
- Allora, $s_j \geq f_m$ ed a_j è compatibile con a_m

Quindi A' è una soluzione ammissibile con cardinalità pari a quella di una soluzione ottima, ossia A' è ottima

Ricapitolando:

Fin qui, abbiamo:

- verificato la sottostruttura ottima per il problema della selezione di attività
- identificato la scelta golosa e dimostrato come ogni scelta golosa origina un solo sottoproblema non banale (non vuoto)
- dimostrato che la scelta golosa è sempre sicura, ossia che per ogni sottoproblema non banale esiste una soluzione ottima che contiene la scelta golosa

In altri termini, abbiamo dimostrato

- come una soluzione ottima del problema originario può essere costruita mediante una sequenza di scelte ottime
- quindi l'algoritmo greedy proposto è corretto

Elementi della strategia golosa

Un algoritmo goloso costruisce una soluzione ottima di un dato problema effettuando una serie di scelte

Ogni decisione viene presa tenendo conto della scelta che, al momento, sembra la migliore

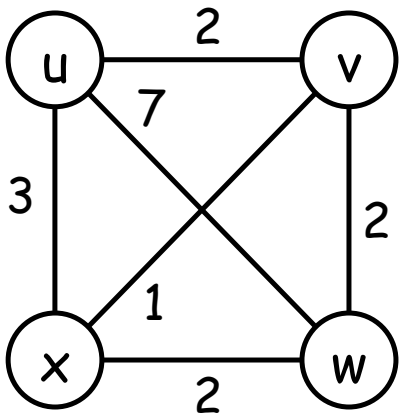
Questa strategia euristica non sempre produce una soluzione ottima

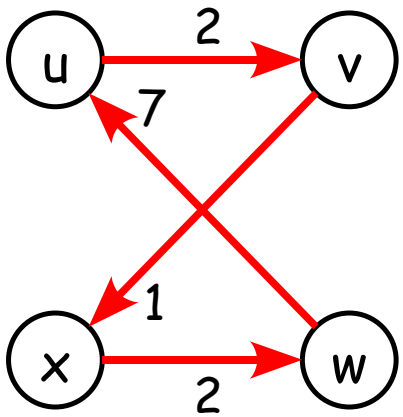
Ma quando lo fa, ci consente di scrivere algoritmi molto efficienti

Un problema non risolvibile mediante algoritmi golosi

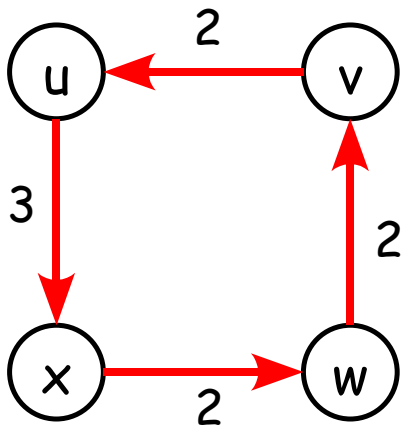
Il problema del commesso viaggiatore: dato un insieme di n città trovare la strada più breve per visitarle tutte una ed una sola volta e tornare alla città di partenza

Assumendo di rappresentare le città e le relative distanze come nodi ed archi di un grafo:





Soluzione greedy
costo 2+1+2+7=12



Soluzione ottima
costo 3+2+2+2=9

Progettazione mediante strategia golosa

Progettiamo algoritmi golosi seguendo la seguente sequenza di passi:

1. definire il problema di ottimizzazione in maniera tale che come ogni scelta genera un solo sottoproblema
2. dimostrare che esiste sempre una soluzione ottima che fa la scelta golosa, quindi la scelta golosa è sempre sicura (proprietà della scelta golosa)
3. dimostrare che è possibile costruire una soluzione ottima combinando la scelta golosa con una soluzione ottima del sottoproblema che la scelta golosa genera (forma "golosa" della sottostruttura ottima)

Queste due proprietà sono due ingredienti chiave di questa tecnica di progettazione di algoritmi golosi

La proprietà della scelta ottima

Proprietà della scelta golosa: stabilisce che una soluzione globalmente ottima può essere ottenuta facendo una scelta localmente ottima

Ad ogni passo, facciamo sempre la scelta che sembra migliore per il sottoproblema corrente, senza considerare le soluzioni dei sottoproblemi

La scelta fatta da un algoritmo goloso può dipendere dalle scelte fatte in precedenza (che determinano il problema corrente) ma non dalle scelte future (soluzioni dei sottoproblemi)

A differenza della PD (che, in generale, procede in maniera bottom-up), una strategia golosa procede di solito in maniera top-down (ossia, dall'alto verso il basso): facendo una scelta golosa dopo l'altra, riducendo ogni istanza di un determinato problema in un problema più piccolo

Programmazione Dinaminca Vs Algoritmi golosi

Non sempre è possibile risolvere un problema con una della tecniche: programmazione dinamica o algoritmi golosi (commesso viaggiatore)

Non sempre problemi che soddisfano la proprietà della sottostruttura ottima possono essere risolti con entrambi i metodi

Ci sono problemi che non possono essere risolti con algoritmi golosi ma possono essere risolti con programmazione dinamica

Se un dato problema può essere risolto mediante algoritmi golosi è inutile scomodare la programmazione dinamica

Il problema dello zaino: due versioni

Il problema dello zaino 0-1

Un ladro entra in un magazzino e trova n oggetti; l'i-esimo oggetto vale v_i euro e pesa w_i Kg, dove v_i e w_i sono dei numeri interi

Il ladro vorrebbe realizzare il furto di maggior valore compatibilmente con il peso W del suo zaino

Ciascun oggetto può essere preso o lasciato; il ladro non può prendere frazioni di oggetti né più volte lo stesso oggetto (di qui il nome "zaino 0-1")

Quali oggetti dovrà prendere??

Il problema dello zaino: due versioni

Il problema dello zaino frazionario

Un ladro entra in un magazzino e trova n oggetti; l'i-esimo oggetto vale v_i euro e pesa w_i chilogrammi, dove v_i e w_i sono dei numeri interi

Il ladro vorrebbe realizzare il furto di maggior valore compatibilmente con il peso W del suo zaino

Le condizioni sono le stesse, con la differenza che il ladro può prendere frazioni di oggetti, anzichè fare una scelta binaria (0-1)

Quali oggetti dovrà prendere??

Il problema dello zaino: due versioni

Il problemi sono molto simili ...

ed entrambi verificano la sottostruttura ottima,

ma **solo** il problema dello zaino frazionario può essere risolto con un algoritmo goloso

Sottostruttura ottima

Problema 0-1:

- consideriamo il carico più prezioso che pesa al più W chili
- fatta una qualsiasi scelta (ossia, togliendo da questo carico un qualsiasi oggetto o_j), la soluzione che rimane deve contenere il carico più prezioso (il cui peso non supera $W - w_j$) che possiamo realizzare con gli $n - 1$ oggetti rimanenti (tutti tranne ob_j)

Problema frazionario:

- consideriamo il carico più prezioso che pesa al più W chili
- assumiamo che il carico ottimo contenga una parte dell'oggetto ob_j il cui peso è pari a w
- se togliamo dal carico ottimo questa parte dell'oggetto ob_j, ciò che resta è il carico più prezioso (il cui peso non supera $W - w$) che possiamo realizzare con gli $n - 1$ oggetti originali più $w_j - w$ chili dell'oggetto ob_j

Un algoritmo goloso per lo zaino frazionario

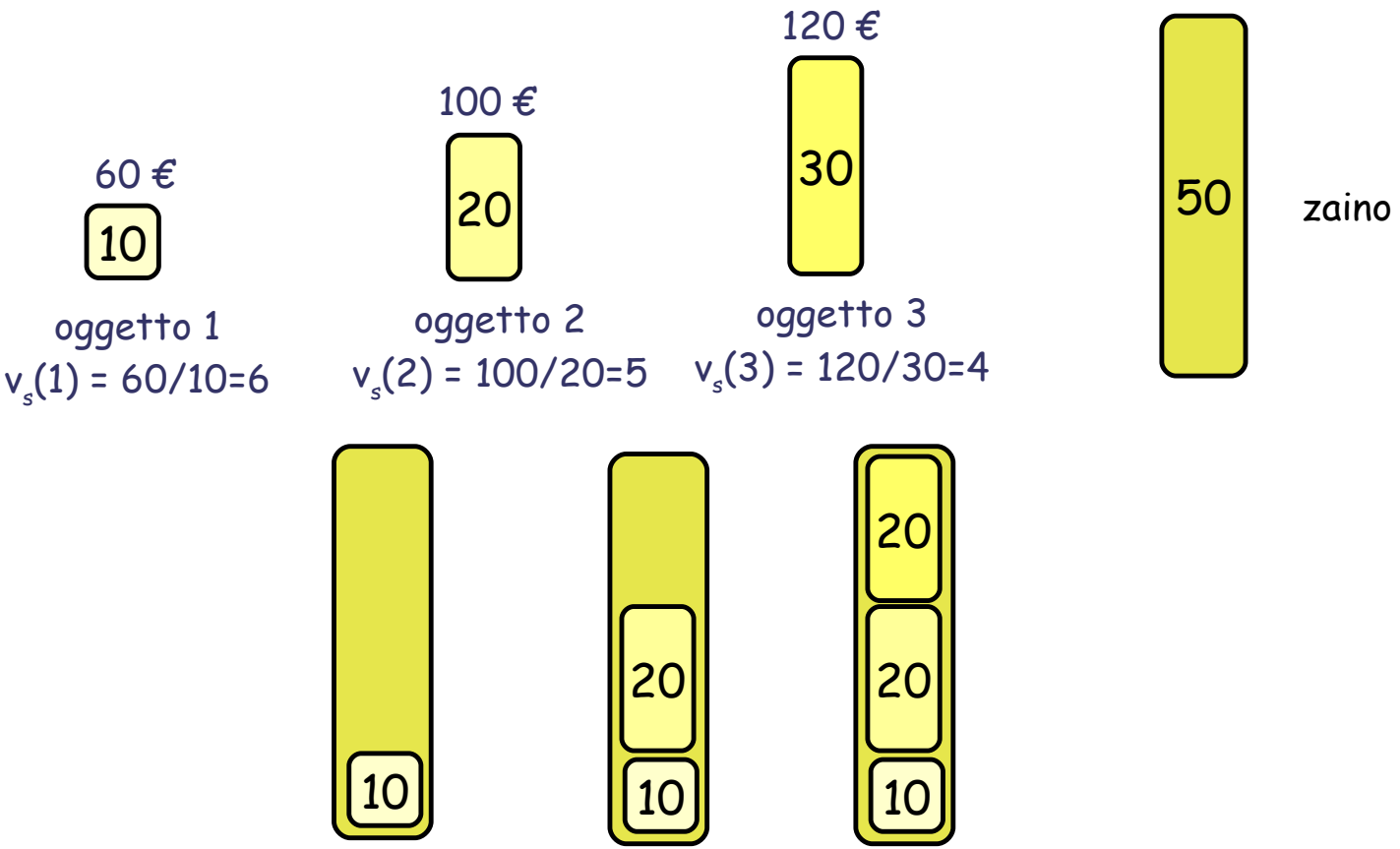
Strategia golosa per il problema dello zaino frazionario:

1. calcoliamo innanzitutto il valore per unità peso, o **valore specifico**, di ciascun oggetto; tale peso specifico è definito come

$$\frac{v_i}{w_i}$$

2. inizialmente, il ladro prende la maggiore quantità possibile dell'oggetto con peso specifico maggiore (la strategia golosa per il problema dello zaino 0-1, consiste nel prendere l'oggetto con peso specifico maggiore)
3. esaurita la scorta di questo oggetto, se nello zaino c'è ancora posto ripete l'operazione

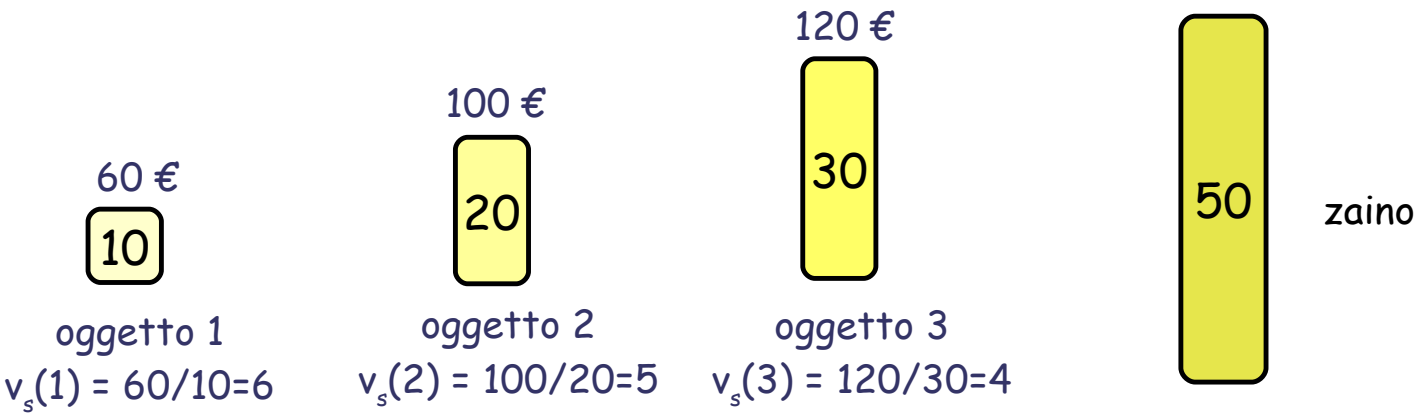
Un algoritmo goloso per lo zaino frazionario

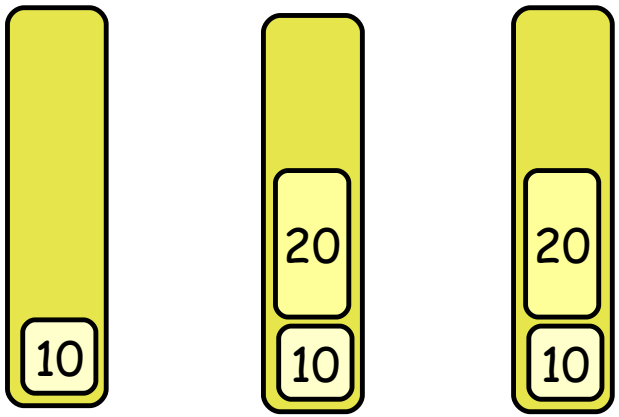


Otteniamo una soluzione composta dali oggetti 1 e 2 (interi), più una parte (2/3) dell'oggetto 3 il cui peso è pari a 50 Kg e con valore

60 + 100 + (2/3) 120 = 60 + 100 + 80 = 240 € (soluzione ottima)

Un algoritmo goloso per lo zaino 0-1





La capacità residua dello zaino non ci consente di inserire il terzo oggetto

otteniamo una soluzione il cui valore è **60+100=160**

La **soluzione ottima** include gli oggetti 2 e 3 ed ha un peso di 50Kg ed un valore pari a **100+120= 220 €**

Proprietà della scelta golosa

Sia $S = \{1, ..., n\}$ un insieme di n oggetti ciascuno con valore v_i, peso w_i e, quindi, con peso specifico v_i/w_i

Assumiamo che gli oggetti in S siano ordinati in maniera decrescente rispetto ai valori v_i/w_i

A questo punto 1 è l'oggetto con maggior peso specifico e denotiamo con q la maggiore quantità di tale oggetto che è possibile inserire nello zaino vuoto

Dobbiamo dimostrare che esiste sempre una soluzione ottima dello zaino frazionario contenente una quantità q dell'oggetto m

Proprietà della scelta golosa

Ad ogni soluzione A del problema può essere associata una tupla

$$t_A = \langle q_1, ..., q_n \rangle$$

Ciascun q_i è tale che $0 \leq q_i \leq w_i$ e rappresenta la porzione dell'oggetto i selezionata ($q_i = 0$ per ogni i non contenuto nello zaino)

- l'insieme degli oggetti residui è $R(t_A) = \{i \in S \ : \ q_i = 0\}$
- gli oggetti nello zaino $Z(t_A) = \{i \in S \ : \ q_i > 0\}$
- il peso di t come $W(t_A) = \sum_{i=1}^{n} q_i w_i \leq W$
- il valore di t come $V(t_A) = \sum_{i=1}^{n} q_i v_i$

Proprietà della scelta golosa

Teorema: è sempre possibile costruire una soluzione ottima dello zaino frazionario contenente la maggiore quantità q dell'oggetto 1 (quello di maggior peso specifico)

Proof: sia A una soluzione ottima del problema ed $t_A = \langle q_1, ..., q_n \rangle$ la tupla ad essa associata

Caso 1: $q_1 = q$, non abbiamo nulla da dimostrare

Proprietà della scelta golosa

Caso 2: $0 \leq q_1 < q$

Consideriamo la soluzione A' la cui tupla associata $t_{A'} = \langle q'_1, ..., q'_n \rangle$ è tale che:

$$q'_i = q_i \qquad \text{per } i > 2$$

$$q'_1 = q = q_1 + (q - q_1)$$

$$q'_2 = q_2 - \frac{v_1}{v_2}(q - q_1)$$

Dimostriamo che A' è ottima, ossia

1. è ammissibile, ossia che $W(t_{A'}) \leq W$
2. e tale che $V(t_{A'}) = V(t_A)$

Ammissibilità

Consideriamo

$$q_1' w_1 + q_2' w_2 =$$

$$q w_1 + (q_2 - \frac{v_1}{v_2}(q - q_1)) w_2 =$$

$$q w_1 + q_2 w_2 - v_1 (q - q_1) \frac{w_2}{v_2}$$

Ora

$$\frac{v_2}{w_2} \leq \frac{v_1}{w_1} \text{ implica } \frac{w_2}{v_2} \geq \frac{w_1}{v_1}$$

Quindi:

$$v_1 (q - q_1) \frac{w_2}{v_2} \geq v_1 (q - q_1) \frac{w_1}{v_1} = (q - q_1) w_1$$

e

$$-v_1 (q - q_1) \frac{w_2}{v_2} \leq -(q - q_1) w_1$$

Ammissibilità

Allora

$$q_1' w_1 + q_2' w_2 = q w_1 + q_2 w_2 - v_1 (q - q_1) \frac{w_2}{v_2} \leq$$

$$q w_1 + q_2 w_2 - (q - q_1) w_1 = q_1 w_1 + q_2 w_2$$

Questo ci permette di concludere che

$$\begin{aligned} W(t_{A'}) &= \sum_{i=1}^{n} q_i' w_i \\ &= q_1' w_1 + q_2' w_2 + (\sum_{i=2}^{n} q_i' w_i) \\ &\leq q_1 w_1 + q_2 w_2 + (\sum_{i=2}^{n} q_i w_i) \\ &= \sum_{i=1}^{n} q_i w_i = W(t_A) \leq W \end{aligned}$$

Valore ottimo

In maniera simile,

$$q'_1 v_1 + q'_2 v_2 =$$

$$q v_1 + (q_2 - \frac{v_1}{v_2}(q - q_1)) v_2 =$$

$$q v_1 + q_2 v_2 - v_1(q - q_1) = q_1 v_1 + q_2 v_2$$

e

$$\begin{aligned} V(t_{A'}) &= \sum_{i=1}^{n} q'_i v_i \\ &= q'_1 v_1 + q'_2 v_2 + (\sum_{i=2}^{n} q'_i v_i) \\ &= q_1 v_1 + q_2 v_2 + (\sum_{i=2}^{n} q_i v_i) = \\ &= \sum_{i=1}^{n} q_i v_i = V(t_A) \end{aligned}$$