TECNICHE ALGORITMICHE: TECNICA GREEDY

[Deme, seconda edizione] cap. 10

Sezione 10.3

Problemi (matematici)

Per prima cosa, definiamo più formalmente un **problema P** come una relazione

$$\mathbf{P} \subseteq \mathbf{I} \times \mathbf{S}$$

Dove **I** è l'insieme delle possibili **istanze** in ingresso ed **S** è l'insieme delle possibili **soluzioni** del problema.

In generale, quando vi viene dato un problema, vi viene data una sua istanza $\mathbf{x} \in \mathbf{I}$ e vi viene chiesto di trovare una sua soluzione **s tale che** $(\mathbf{x}, s) \in \mathbf{P}$

Tipi di Problemi

Problemi di Decisione: problemi che richiedono di **verificare** una certa **proprietà** sull'input. Sono problemi per cui $\boldsymbol{S = \{0, 1\}}$.

Esempio: dato un grafo G, dire se è connesso.

Problemi di Ricerca: data un'istanza di un problema, restituire una **soluzione (ammissibile)**.

Esempio (cammini da sorgente unica): dato un grafo **pesato** G ed un vertice S, trovare un albero A (di radice S) che contenga tutti i vertici raggiungibili da S, in cui ogni cammino $S \rightarrow V$ sia un cammino tra S e V in G.

Problemi di Ottimizzazione: data un'istanza di un problema ed una **funzione obiettivo** valuta(Sol) che per ogni soluzione Sol restituisce un valore di tale soluzione in base ad un determinato **criterio**, restituire una **soluzione** Sol **ottima** (cioè per cui non esista una soluzione Sol' t.c. valuta(Sol')>valuta(Sol) per i problemi di **massimizzazione** o valuta(Sol')<valuta(Sol) per quelli di **minimizzazione**).

Esempio (cammini minimi da sorgente unica): dato un grafo **pesato** G ed un vertice S, trovare un albero A (di radice S) che contenga tutti i vertici raggiungibili da S, in cui ogni cammino $S \rightarrow V$ sia un cammino **minimo** tra S e V in G.

Relazioni tra Problemi di Ottimizzazione e di Ricerca

È facile notare che esiste una relazione tra i problemi di **ottimizzazione** e quelli di **ricerca**.

In particolare **una soluzione di un problema di ottimizzazione è anche una soluzione di un equivalente problema di ricerca** (ma non vale il viceversa).

Infatti, il problema di ottimizzazione «**ordina**» le soluzioni di quello di ricerca secondo il dato **criterio**, e tra queste sceglie quella/quelle «**migliori**»

Si dice che (tutte) le soluzioni del problema di ricerca sono **ammissibili** per quello di ottimizzazione

Le soluzioni del problema di ottimizzazione sono **ottime** secondo il dato **criterio**.

Trovare soluzioni ottime è difficile

In molti problemi di ottimizzazione, trovare una soluzione **ammissibile** è relativamente semplice. Al contrario, trovare una soluzione **ottima** può essere **difficile**.

ESEMPIO. Una soluzione **ammissibile** per il problema dei cammini minimi da sorgente unica in un grafo è **pesato** si trova facilmente con una qualsiasi visita a partire da S. Trovare una soluzione **ottima** non è facile.

Tra qualche lezione, grazie alla tecnica Greedy saremo in grado di farlo.

Ma come funziona la tecnica Greedy, e a quali problemi si applica?

Un problema di ottimizzazione: il problema dello zaino frazionario

Un ladro entra in un magazzino e trova n oggetti. L'i-esimo oggetto o_i ha un valore di $\boldsymbol{c_i}$ euro e pesa $\boldsymbol{p_i}$ chilogrammi. $\boldsymbol{v_i = c_i / p_i}$ è il **valore per unità di peso**.

Gli oggetti sono frazionabili (es. delle polveri), quindi il ladro ne può prendere anche solo una frazione $\boldsymbol{x_i}$, $0 \le \boldsymbol{x_i} \le 1$. In tal caso il valore della parte presa sarà $\boldsymbol{c_i x_i}$.

Il ladro ha solo uno zaino, che può contenere oggetti per un massimo di $\boldsymbol{P}$ chilogrammi.

Quali oggetti e in quale quantità dovrà prendere il ladro per ottenere il massimo guadagno dal furto?

(formalmente, ci viene richiesto di valorizzare x_i per ogni $1 \le i \le n$ in maniera che $\Sigma\, x_i c_i$ sia massima)

Formalizziamo il problema dello zaino frazionario

$$\max \sum_{i=1..n} x_i c_i$$

$$\sum_{i=1..n} x_i p_i \leq P$$

$$0 \leq x_1 \leq 1$$

…

$$0 \leq x_n \leq 1$$

Ottimalità: tutte le soluzioni (ammissibili) che massimizzano il valore totale dello zaino (la **funzione obiettivo**) sono **soluzioni ottime**

Ammissibilità: tutte le valorizzazioni dei vari x_i t.c il peso totale sia minore di P e che x_i sia compreso tra 0 e 1 sono **soluzioni ammissibili**

Zaino frazionario – esempio

i	1	2	3
p_i	10	20	30
c_i	60	100	120
v_i	6	5	4

Con $P = 50$, una soluzione ottima per questo problema è

$$X = \langle x_1 = 1, x_2 = 1, x_3 = 2/3 \rangle$$

Proprietà della sottostruttura ottima

Un problema ha la **proprietà della sottostruttura ottima** se **una soluzione ottima del problema «include» (contiene/è ottenibile efficientemente a partire da) le soluzioni ottime dei suoi sottoproblemi**.

Se un problema di ottimizzazione gode della proprietà della sottostruttura ottima, in genere può essere risolto con tecniche di tipo **greedy** o di **programmazione dinamica** (che vedremo prossimamente).

ESEMPIO. Il problema dello zaino frazionario gode della proprietà della sottostruttura ottima (vedi dimostrazione di seguito). Quindi

Se $X = \langle x_1 = 1, x_2 = 1, x_3 = 2/3 \rangle$ è ottima per il problema precedente, allora $X = \langle x_2 = 1, x_3 = 2/3 \rangle$ è ottima per il sottoproblema ottenuto eliminando l'oggetto 1 e con $P = 50 - 10 = 40$.

Sottostruttura Ottima nel Problema dello Zaino (dimostrazione)

Indichiamo con $\boldsymbol{Q_{n,P}}$ il problema dello zaino frazionato con $\boldsymbol{n}$ **oggetti** e **capacità** dello zaino $\boldsymbol{P}$.

(**ipotesi**) Sia $\boldsymbol{X_n = \langle x_1, x_2, \dots, x_n \rangle}$ una soluzione **ottima** per $Q_{n,P}$, di **valore** $\boldsymbol{C_{n,P} = \sum_{i=1..n} x_i c_i}$.

Dobbiamo dimostrare che $\boldsymbol{X_{n-1} = \langle x_1, x_2, \dots, x_{n-1} \rangle}$ di valore $C_{n-1,P-x_n p_n}$ **è una soluzione ottima** per il sottoproblema $Q_{n-1,P-x_n p_n}$

Come lo facciamo? Per **assurdo**.

Ipotizziamo che esista X'_{n-1} tale che il suo valore $\boldsymbol{C'_{n-1,P-x_n p_n} = \sum_{i=1..n-1} x'_i c_i} > \boldsymbol{C_{n-1,P-x_n p_n} = \sum_{i=1..n-1} x_i c_i}$, dobbiamo raggiungere una **contraddizione**.

Sottostruttura Ottima nel Problema dello Zaino (dimostrazione)

Se esistesse X'_{n-1}, potremmo usarlo per trovare una soluzione X'_n, semplicemente **aggiungendo tutto l'n-esimo oggetto a $\boldsymbol{X'_{n-1}}$**.

Il valore di X'_n sarebbe quindi

$C'_{n-1,P-x_n p_n} + x_n c_n$ ma
$C'_{n-1,P-x_n p_n} + x_n c_n > C_{n-1,P-x_n p_n} + x_n c_n = \boldsymbol{C_{n,P}}$

Quindi **il valore di $\boldsymbol{X'_n}$ sarebbe maggiore di quello di $\boldsymbol{X_n}$**. Ma questo è **impossibile**, dato che X_n è una soluzione ottima e non possono esistere soluzioni di valore maggiore.

cvd.

Proprietà della scelta greedy

La sottostruttura ottima ci fornisce un'importante relazione tra soluzioni di un problema e soluzioni dei suoi sottoproblemi, ma non ci spiega come risolvere e/o scomporre un problema in sotto-problemi.

Un problema gode della **proprietà della scelta greedy** se è sempre possibile **scegliere** in esso (in genere) una variabile x_h, attribuire un valore **ammissibile** e **localmente ottimo** a tale variabile, «rimuoverla» dal problema, ed ottenere un **sottoproblema** la cui soluzione unita al valore di x_h sia una soluzione **ottima** per il problema originario.

ESEMPIO. Il problema dello zaino frazionario gode della proprietà della scelta greedy. In un problema la scelta greedy ricade sulla variabile con rapporto costo/peso (v_h) maggiore. Nell'esempio, la prima variabile ad essere scelta sarà x_1.

Scelta greedy nel Problema dello Zaino frazionario (dimostrazione informale)

Sia h l'oggetto con v_h maggiore.
Senza perdita di generalità, assumiamo che non esista nessun altro oggetto con rapporto valore/peso uguale e che $P \geq p_h$.

Dimostriamo che ogni soluzione ottima del problema **deve contenere l'oggetto $\boldsymbol{h}$ per intero**.

Supponiamo, per **assurdo**, che $\boldsymbol{C_{n,P} = \sum_{i=1..n} x_i c_i}$ sia il valore di una soluzione **ottima** X in cui $\boldsymbol{x_h = 1 - t}$.

Allora possiamo scegliere un'altra variabile $x_j \geq tp_h/p_j$, ed ottenere una nuova soluzione ammissibile X' in cui $\boldsymbol{x'_h = x_h + t}$ **e** $\boldsymbol{x'_j = x_j - tp_h/p_j}$. Con $\boldsymbol{tp_h}$ che è la quantità (non la percentuale) di peso aggiunto ad h e tolto da j. Il valore di X' sarà quindi
$\boldsymbol{C'_{n,P} = C_{n,P} - tp_h v_j + tp_h v_h}$

ma $v_h > v_j$ per ipotesi, $t > 0$ e $p_h > 0$,

$-tp_h v_j + tp_h v_h > 0$ quindi

$\boldsymbol{C'_{n,P} > C_{n,P}}$ che contraddice l'ipotesi che X sia una soluzione ottima

Scelta greedy: Appetibilità

La scelta greedy deve essere fatta in maniera efficiente, altrimenti l’algoritmo non sarebbe «greedy».

Per fare ciò, bisogna definire per le variabili del problema un criterio o valore di **appetibilità**, che permette di effettuare la **scelta locale greedy** in tempi ragionevoli.

L’appetibilità **ordina** le variabili, in maniera che in ogni sottoproblema sia possibile **effettuare efficientemente la scelta greedy**.

L’ordinamento può essere **fisso** o **variare** durante lo svolgimento dell’algoritmo.

Esempio. Nel problema dello zaino frazionario, l’appetibilità ordina gli oggetti in base all’ordinamento non crescente dei v_i.

La tecnica Greedy

«Meglio un uovo oggi che una gallina domani» (A. Bertossi).

La tecnica Greedy risolve problemi di **ottimizzazione** che godono delle proprietà della sottostruttura ottima e della scelta greedy.

Greedy in inglese vuol dire **goloso, ingordo**.

Invece che affrontare il problema per intero, lo scompone in sottoproblemi. Un (sotto)problema viene scomposto in

- una **scelta locale** (scelta di una variabile e attribuzione del valore)
- uno o più suoi **sottoproblemi**.

La scelta locale viene presa in maniera **ottima localmente**, senza preoccuparsi del resto del problema (da qui il nome greedy)

Il sottoproblema viene a sua volta risolto in maniera greedy, finché non si raggiunge un sottoproblema banale.

La soluzione ottima globale è la **composizione** delle soluzioni ottime localmente ottenute.

Schema Astratto di Algoritmo Greedy

1. Scegli la variabile del problema con **appetibilità maggiore**
2. Attribuisci ad essa **il valore ammissibile più promettente** a livello **locale** (scelta greedy)
3. Ottieni il **sottoproblema** risultante dall’eliminazione della variabile scelta al punto 1
4. Se il problema è risolto, restituisci I valori delle variabili assegnati al punto 2, altrimenti vai al punto 1

Algoritmo Greedy per lo Zaino frazionario

1. Scegli la variabile x_i con v_i maggiore
2. $x_i = \min(1, P/p_i)$
3. Elimina x_i dal problema e $P = P - x_i p_i$
4. Se il problema non contiene più nessuna variabile o $P = 0$, restituisci $X = \langle x_1, \dots, x_n \rangle$, altrimenti vai al punto 1.

Greedy per scelta di elementi

Nella maggior parte dei casi, gli algoritmi greedy affrontano problemi dove, dato un insieme di elementi **A**, è necessario selezionare un sottoinsieme **S** di elementi **ammissibile** e **ottimo**. In questo caso, $x_i \in \{0, 1\}$ ed è possibile riscrivere lo schema in questo modo

Greedy **(A)**
    S <- insieme vuoto
    while (S non è una soluzione)
        estrai da A l'elemento *più appetibile* a
        if S U {a} è *ammissibile*
            aggiungi a all'insieme S

ATTENZIONE: questo è uno schema generale che fornisce **un'idea**, gli algoritmi greedy non sono per forza particolarizzazioni di questo.

ATTENZIONE2: le appetibilità degli elementi dell'insieme A possono essere note fin dall'inizio e **restare costanti**, oppure possono venire **modificate** dalle scelte già fatte.
Si hanno allora rispettivamente due particolarizzazioni dello schema precedente.

Greedy scelta di elementi – appetibilità fisse

Greedy1 **({a_1, a_2, …a_n})**

S <- insieme vuoto

ordina gli a_i in ordine non crescente di appetibilità

for each a_i nell'ordine

if $\{a_i\} \cup S$ è ammissibile

aggiungi a_i ad S

Esempio: l'algoritmo di Kruskal che vedremo poi.

Greedy scelta di elementi – appetibilità modificabili

Greedy2 **({a_1, a_2, …a_n})**

S <- insieme vuoto

valuta le appetibilità degli a_i

while ci sono ancora elementi da scegliere

scegli l'a_i più appetibile

if $\{a_i\} \cup S$ è ammissibile

aggiungi a_i ad S

aggiorna le appetibilità degli a_i

Esempio: gli algoritmi di Dijkstra e di Prim che vedremo poi.

Un esempio di <u>non</u> applicabilità: il problema del resto

Input del problema:

- un insieme di interi che rappresentano i tagli disponibili delle banconote o monete (ad es. **1, 2, 5, 10, 20, 50,** … euro);
- un intero **R** che rappresenta un “resto”, ad es. **84** euro;

Output del problema (soluzione):

- un (multi-)insieme di cardinalità minima di monete la cui somma sia uguale al resto.

Un insieme di monete è quindi:

- **ammissibile** se la loro somma è uguale al resto, ad esempio:
  {20, 20, 20, 20, 2, 1, 1} (**7** pezzi)
- **ottimo** se il numero di monete è il **minimo** possibile, ad es.:
  {50, 20, 10, 2, 2} (**5** pezzi)

Bozza di algoritmo greedy per il resto

***GreedyResto* (A <- {tagli disponibili}, R)**

S <- insieme vuoto

while A ≠ ∅ **and** valore(S) ≠ R

x <- **moneta con valore più elevato in A**

if $\mathbf{valore}(\boldsymbol{S} \cup \{\boldsymbol{x}\}) \leq \boldsymbol{R}$

$S \leftarrow S \cup \{x\}$

else

$\mathrm{A} \leftarrow \mathrm{A} - \{\mathrm{x}\}$

return S

Sottostruttura ottima nel problema del resto

Il **problema del resto non ha**, per un **insieme arbitrario** di tagli, **la proprietà della sottostruttura ottima**.

tagli disponibili = {1, 5, 6}   R = 15

la sequenza di scelte greedy {6, 6, 1} è una soluzione ottima del sottoproblema R = 13, ma da essa non si può ricavare una soluzione ottima per il problema R = 15 ({5, 5, 5}).

Fallimento Greedy

tagli disponibili = **{1, 5, 6}**

- caso 1: **R = 13** scelte greedy: **6, 6, 1**; è la soluzione ottima.
- caso 2: **R = 15** scelte greedy: **6, 6, 1, 1, 1**;
  mentre la soluzione ottima è **5, 5, 5**.

Fallimento Greedy – II

In questo caso la tecnica greedy **non trova** neppure **una soluzione ammissibile**, anche quando la soluzione esiste.

Esempio: tagli disponibili = **{10, 5, 2}**

caso 1: **R = 19** scelte greedy: **10, 5, 2, 2**; soluzione ottima.

caso 2: **R = 6** scelte greedy: **5, fallisce**

mentre la soluzione esiste ed è **2, 2, 2**.

Cosa devo aver capito fino ad ora

- Cos'è la tecnica greedy
- A che genere di problemi può essere applicata
- Condizioni necessarie per l'applicazione della tecnica greedy
- Schema di algoritmo greedy:
  - Con appetibilità fisse
  - Con appetibilità modificabili

…se non ho capito qualcosa

- Alzo la mano e chiedo
- Ripasso sul libro
- Chiedo aiuto sul forum
- Chiedo o mando una mail al docente