Problemi di ottimizzazione

- Allocazione di risorse (merci in un magazzino)
- Scheduling (ordinamento temporale)
- Pianificazione di investimenti
- Pattern matching: date due sequenze di simboli:

  AACCGATGTACCT

  CGAACGATACGGTTAC

  trovare la piu' lunga sottosequenza comune
- Distanza minima in reti: dato un insieme di città collegate da strade trovare il percorso minimo che collega ogni coppia di città

Soluzione di un problema di ottimizzazione

- Ad ogni problema è associato un costo/valore
- una soluzione e' frutto di una sequenza di scelte, ciascuna delle quali contribuisce a determinare il costo/valore finale
- si è interessati a trovare una soluzione che abbia un costo/valore ottimo (minimo o massimo)

Algoritmi greedy

Si applicano a problemi di ottimizzazione in cui dato un insieme di oggetti {a1,…,an} occorre selezionare un sottoinsieme "ottimo" S di oggetti che verificano una determinata proprietà

Idea: "per trovare un soluzione ***globalmente ottima***, scegli ripetutamente soluzioni ***ottime localmente***"

Esempio: Il problema del resto

Avendo a disposizione monete di vario tipo determinare una collezione *minima* di monete la cui somma sia uguale al resto.

Ad esempio: hai a disposzione monete da

50, 20,10, 2 e 1 cent.euro,

il resto “ottimo” di 87 è formato da:

50+20+10+5+1+1

Struttura degli algoritmi greedy

- Si assume che gli oggetti abbiano associato un *valore di "appetibilità".*
- La soluzione viene costruita *incrementalmente* scegliendo ad ogni passo l'oggetto che ha appetibilita' maggiore e puo' essere aggiunto a quelli già selezionati.

Algoritmi Greedy - Schema generale 1

Se le appetibilita' degli oggetti sono note fin dall'inizio e non vengono modificate

Greedy1 ($\{a_1, a_2, \ldots a_n\}$)

$S \leftarrow \Phi$

" ordina gli a_i in ordine non crescente di *appetibilita'* "

for ogni a_i nell'ordine **do**

if "a_i puo' essere aggiunto a S"

then $S \leftarrow S \cup \{a_i\}$

return S

Algoritmi Greedy - Schema generale 2

Se le appetibilita' degli oggetti possono essere modificate dalle scelte gia' fatte.

Greedy2 ($\{a_1, a_2, \ldots a_n\}$)

$S \leftarrow \Phi$

"valuta le appetibilita' degli a_i "

while "ci sono elementi da scegliere" **do**

"scegli l'a_i piu' *appetibile*"

if "a_i puo' essere aggiunto a S"

then $S \leftarrow S \cup \{a_i\}$

"aggiorna le appetibilita' degli a_i "

return S

Il problema dello zaino

Un ladro vuole rubare dei beni che trasporterà in uno zaino. Può prendere W chili di bottino (W è la capacità dello zaino). Deve scegliere tra n articoli, ognuno dei dei quali ha peso $\mathbf{w_i}$ e valore $\mathbf{v_i}$.

Può prendere qualsiasi articolo, purchè non ecceda la capacità W.

Problema:

Quale è il massimo valore che può mettere insieme e quali articoli deve prendere per *massimizzare* il valore complessivo del bottino?

Due varianti del problema:

- **Lo zaino frazionario** (o continuo): si possono prendere frazioni di ciascun articolo.

- **Lo zaino discreto** (o zaino 0-1): gli articoli sono indivisibili, quindi ciascun articolo o lo si prende oppure no (scelta 0-1)

Lo zaino frazionario è risolvibile con un metodo greedy

Consideriamo come valore di *appetibilità* il valore di ciascun oggetto ($\mathbf{v_i}$) per unità di peso ($\mathbf{w_i}$):

$\mathbf{v_i/w_i}$

Idea dell'algoritmo greedy:

Prendi *il piu' possibile* dell'oggetto con il piu' alto rapporto $\mathbf{v_i/w_i}$.

Se la dotazione dell'oggetto e' esaurita e non hai ancora riempito lo zaino, considera il *prossimo* oggetto con il piu'alto rapporto $\mathbf{v_i/w_i}$. Ripeti il procedimento finchè lo zaino è pieno.

Proprietà della sottostruttura ottima

Se rimuovo una quantità **w** di un articolo **j** da un carico ottimo ottengo un carico ottimo che pesa al piu' **W-w** e che posso mettere insieme avendo a disposizione n-1 articoli con le quantità originarie e $\mathbf{w_j}$ **-w** chili dell'articolo **j**.

Altrimenti: se ci fosse un carico che vale di più, potrei ottenere un carico migliore con la dotazione originaria degli n articoli e peso **W**, aggiungendo **w** chili di **j** a quel carico.

Proprietà della scelta greedy

- Sia h un articolo con il più alto rapporto $\mathbf{v_h/w_h}$.
- C'e' una soluzione ottima L in cui prendo il massimo di h, cioè

$$\mathbf{L_h = \min(W, w_h)}$$

Dopo aver scelto L_h il problema si riduce a trovare una soluzione ottima scegliendo tra n-1 oggetti (h escluso) e potendo mettere insieme un peso non superiore a $W - L_h$. Si ripete il ragionamento considerando la prossima scelta greedy.

Knapsack(W, **w**,**v**)

Ordina {1,…,n} per $\mathbf{v_i}/\mathbf{w_i}$ non crescente

$C \leftarrow W$

for i = 1 **to** n **do**

$L_i \leftarrow 0$

$i \leftarrow 1$

while (i ≤ n) and (C > 0) **do**

$L_i \leftarrow \min(C, \mathbf{w_i})$

$C \leftarrow C - L_i$

$i \leftarrow i+1$

return L

Knapsack(W, **w**,**v**) (L valori frazionari)

Ordina {1,…,n} per $\mathbf{v_i}/\mathbf{w_i}$ non crescente

$C \leftarrow W$

for i = 1 **to** n **do**

$L_i \leftarrow 0$

$i \leftarrow 1$

while (i ≤ n) and (C > 0) **do**

if ($\mathbf{w_i}$ > C)

then $L_i \leftarrow C$ ($L_i \leftarrow C/\mathbf{w_i}$)

$C \leftarrow 0$

else $L_i \leftarrow \mathbf{w_i}$ **(**$L_i \leftarrow 1$**)**

$C \leftarrow C - \mathbf{w_i}$,

$i \leftarrow i+1$

return L

Esempio

	peso **w**	valore **v**	**v**/**w**
articolo 1	10	60	6
articolo 2	20	100	5
articolo 3	30	120	4

Esecuzione algoritmo

i	C	L_1	L_2	L_3
/	50	0	0	0
1	**40**	**10**	0	0
2	**20**	10	**20**	0
3	**0**	10	20	**20**

Soluzione: V = 10*6 + 20 *5+20* 4 = 240

Zaino 0-1

Stesso problema, ma gli articoli vanno presi *interamente*:

$\mathbf{L_i} = 1$ se prendiamo l'articolo i

$\mathbf{L_i} = 0$ se non prendiamo l'articolo i

Vale la proprietà della sottostruttura ottima anche per lo zaino0-1: se ad un carico ottimo di peso **W** tolgo un oggetto **j**, ottengo un carico ottimo di peso $\mathbf{W - w_j}$

GreedyKnapsack0-1(W, **w**,**v**)

Ordina {1,…,n} per $\mathbf{v_i}/\mathbf{w_i}$ non crescente

$C \leftarrow W$

for i = 1 **to** n **do**

$L_i \leftarrow 0$

$i \leftarrow 1$

while (i ≤ n) and (C > 0) **do**

if ($\mathbf{w_i}$ > C)

then $L_i \leftarrow 0$

else $L_i \leftarrow \mathbf{1}$

$C \leftarrow C - \mathbf{w_i}$,

$i \leftarrow i+1$

return L

Rivediamo l'esempio

	peso **w**	valore **v**	**v**/**w**
articolo 1	10	60	6
articolo 2	20	100	5
articolo 3	30	120	4

Esecuzione algoritmo

i	C	L_1	L_2	L_3
/	50	0	0	0
1	**40**	**10**	0	0
2	**20**	10	**20**	0
3	**20**	10	20	**0 (w=30)**

Soluzione: V = 10*6 + 20 *5= 160

E’ottima la soluzione?

NO!!

Se prendo l’articolo 2 e l’articolo 3 ottengo:

$$V = 100 + 120 = 220$$

La strategia greedy non trova una soluzione ottima per il problema dello zaino 0-1

Algoritmo Zaino 0/1

```
int main(int argc,char **argv)
{
        int  x[N];
        int  i, ricavi, OreTot = 10;
        float scelta[N];
        int ore[] = {1, 2, 3, 4, 5};
        int ricavo[] = {2, 3, 5, 6, 50};
        int OreIniziali = OreTot;
        for (i=0; i < N; i++) {
            scelta[i] = (float) ricavo[i] / ore[i];
        };
        ricavi = 0;
        ordina(scelta, ore, ricavo, N);
        for (i=0; i < N; i++) {
                if (ore[i] <= OreTot) {
                        OreTot -= ore[i]; x[i] = 1; ricavi += ricavo[i];
                }
                else x[i] = 0;
        };
```

Algoritmo Zaino 0/1

```
        printf("\nCoppie totali ");
        for (i=0; i < N; i++) {
                printf ("\n (%f, %d, %d) ", scelta[i], ore[i], ricavo[i]);
        };

        printf("\nCoppie scelte: (Ore Scelte = %d su %d): ",
                                        OreIniziali - OreTot, OreIniziali);

        for (i=0; i < N; i++) {
                if (x[i] == 1) {
                        printf (" (%d, %d) ", ore[i], ricavo[i]);
                }
        };

        printf("\nRicavi Totali (ore usate %d): %d", OreIniziali - OreTot, ricavi);

  return 0;
}
```

Non vale il principio della scelta greedy:

la scelta se prendere o no un oggetto non dipende dalla sua appetibilità.

Per trovare una soluzione ottima bisogna *comparare* la soluzione del sottoproblema in cui si e' scelto di prendere un articolo con la soluzione in cui si e' scelto di *non* prendere quell'articolo.