Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

In questa esercitazione vengono proposti alcuni probemi risolubili con tecniche di programmazione dinamica.

Il primo ripercorre esattamente l'esempio visto a lezione, si tratta semplicemente di tradurre lo pseudo-codice in un programma C. L'esercizio presenta il classico problema noto come problema dello zaino, partendo da una versione semplificata.

Gli esercizi successivi presentano delle varianti al problema dello scheduling che possono essere risolte usando queste tecniche.

1 Compagnie aeree

Per fidelizzare i clienti, la compagnia aerea Swindle-fly offre ai suoi viaggiatori dei *jet-points*: per ogni tratta è definito un *valore in jet-points*; i viaggiatori possono *accumulare* jet-points nelle varie tratte e usarli, al posto dei dollari, per acquistare delle tratte; i jet-points di un viaggio si accumulano additivamente, cioè i punti acquisiti con una tratta si sommano ai punti precedentemente acquisiti.

Il programma di fidelizzazione è scaduto a fine anno e i jet-points non spesi non saranno più utilizzabili per comprare tratte aeree. La compagnia consente però *generosamente* ai suoi clienti di usare i propri jet-points rimasti per acquistare dei buoni in dollari chiamati *swindles*; questi swindles potranno essere usati nel 2014 sull'acquisto di tratte offerte dalla compagnia Swindle-fly.

Si supponga che il costo in jet points degli swindles sia come riportato nella tabella seguente:

Swindles (valore in dollari)	Jet-points
100	20.000
107	22.000
124	24.000
133	26.000
139	28.000
155	30.000
172	32.000
178	34.000
184	36.000
190	38.000
195	40.000

Se il totale di jet-points è 60.000 allora il valore massimo in dollari che può essere realizzato acquistando swindles è pari a 311 $ (uno swindle da 133 $ e uno swindle da 178 $). Se il totale di jet-points è 100.000 allora il valore massimo in dollari che può essere realizzato acquistando swindles è pari a 528 $ (due swindle da 172 $ e uno swindle da 184 $). Se il totale di jet-points è 140.000 allora il valore massimo in dollari che può essere realizzato acquistando swindles è pari a 740 $ (tre swindle da 172 $, uno swindle da 124 $ e uno swindle da 100 $).

Come è possibile calcolare il massimo valore in swindles acquistabili con j jet-points, avendo in input una tabella tipo quella dell'esempio?

2 Esercizio: *scheduling* di intervalli pesati

Un intervallo è dato da una tripla di interi positivi: (i, f, v), dove i è il tempo di inizio, f è il tempo di fine, v è il valore (o peso) dell'intervallo. Ad esempio, un intervallo può rappresentare una lezione che inizia al tempo i, finisce al tempo f e coinvolge v studenti. Dato un insieme I di intervalli, una soluzione al problema dello scheduling è data da un sottoinsieme $S \subseteq I$ di intervalli che non si sovrappongono fra loro. Il valore di una soluzione S è dato dalla somma dei valori degli intervalli di S, ovvero:

$$\sum_{(i,f,v) \in S} v$$

Dovete scrivere un programma che, dato un insieme I di intervalli in input, fornisca in output il valore massimo fra tutte le possibili soluzioni, insieme ad una qualunque delle soluzioni ottimali (ovvero una soluzione che ha tale valore massimo). Potete decidere autonomamente il formato di input e output, magari ispirandovi all'esercizio sullo scheduli di intervalli (non pesati).

Ricordate la tecnica di programmazione dinamica vista a lezione, costruita a partire dalle seguenti osservazioni riguardanti la struttura del problema e la possibiltà di ridurlo in sottoproblemi.

Innanzitutto, ordinate gli elementi di i in base al tempo di fine, quindi avrete $I = \{(i_1, f_1, v_1), (i_2, f_2, v_2), \ldots, (i_n, f_n, v_n)\}$ con $f_1 \leq f_2 \leq \ldots \leq f_n$. Per ogni indice j tra 1 e n, definite $p(j)$ come il più grande indice $i < j$ tale che l'intervallo di indice i non si sovrappone all'intervallo di indice j. In altre parole, $p(i)$ denota l'*ultimo* intervallo che finisce prima dell'intervallo denotato da j.

Detto $Opt(j)$ il valore di una qualsiasi soluzione ottimale S_j costruita usando gli intervalli di indici $\{1, 2, \ldots, j\}$, vale quindi questa proprietà ricorsiva:

$$Opt(j) = max\{v_j + Opt(p(j)), Opt(j-1)\}$$

Inoltre, notate che se $j \in S_j$, allora $Opt(j) = v_j + Opt(p(j))$, altrimenti $Opt(j) = Opt(j-1)$.

Esempio di funzionamento

INPUT
6 6-9 4 1-5 2 2-7 4 11-15 1 10-14 2 3-12 7

OUTPUT
8 1-5 2 6-9 4 10-14 2

3 Esercizio: problema dello zaino

3.1 Versione semplificata

Un altro classico problema risolubile con tecniche di programmazione dinamica è conosciuto come *problema dello zaino*.

Iniziamo considerandone una versione semplice.

Abbiamo uno zaino che sopporta un peso massimo P e un insieme $T = \{1, 2, \ldots, n\}$ di *tipi* di oggetti. Ogni tipo i di oggetti ha un peso p_i e un valore v_i, entrambi interi positivi. Per ogni tipo è disponibile una fornitura illimitata di oggetti. Vogliamo riempire lo zaino non superando P con il peso complessivo degli oggetti, ma allo stesso tempo massimizzando la somma dei valori degli oggetti nello zaino.

È chiaro che in linea di principio potremmo provare a enumerare tutti gli insiemi di oggetti che stanno nello zaino, e cercare quello con valore massimo. Se però i tipi sono molti, e la capacità dello zaino grande, il numero di soluzioni da esaminare diventa improponibile.

Esiste una relazione interessante tra una soluzione del nostro problema e la soluzione dello stesso problema per uno zaino più piccolo (cioè meno resistente): se ho una soluzione ottima (cioè il cui valore è massimo tra tutte le soluzioni possibili) per uno zaino che regge P, e tolgo dalla soluzione un oggetto qualsiasi di tipo t, ottengo una soluzione ottima per uno zaino che regge $P - p_t$. Infatti, se per assurdo esistesse una soluzione migliore con peso inferiore a $P - p_t$, potrei aggiungerle un oggetto di tipo t e ottenere così una soluzione migliore per il problema originale (il che è impossibile, avendo assunto che la soluzione fosse ottima).

Questo significa che se conosciamo la soluzione ottima per uno zaino di peso Q, possiamo ottenere nuove soluzioni per zaini di grandezza superiore aggiungendo un oggetto di tipo t, per ogni tipo t in T. In particolare, se ho una soluzione di valore V per uno zaino di peso Q, allora so che esiste una soluzione di valore $V + v_t$ per uno zaino di peso $P + p_t$, per ogni t in T. *Tutte le soluzioni ottime si ottengono in questo modo.*

Queste osservazioni consentono di ridurre il problema dello zaino per il peso P al problema dello zaino per i pesi $< P$. Questo suggerisce di usare la programmazione dinamica: costruite un vettore s di lunghezza P, tale che $s[i]$ contenga il valore delle soluzioni ottime per uno zaino di peso i. Il vettore si può costruire partendo da $i = 0$ e incrementando i, sfruttando la relazione vista sopra. Una volta completato il vettore s, il valore ottimo di una soluzione per P è chiaramente dato da $s[P]$.

Provate a ragionarci su... se volete ulteriori dettagli, e un esempio completo, vi invito a leggere la spiegazione preparata dal prof. Sebastiano Vigna!

3.2 Versione generale

Abbiamo ancra uno zaino che sopporta un peso massimo P, ma invece di avere un insieme di tipi di oggetti disponibili in quantità illimitata, abbiamo ora un insieme *finito* di oggetti $E = \{1, 2, \ldots, m\}$, ciascuno dotato di peso e valore. Dobbiamo nuovamente riempire lo zaino nel miglior modo possibile.

Più precisamente, ogni oggetto di E è rappresenato da una coppia (p, v), dove p è il peso dell'oggetto e v il suo valore. la somma degli oggetti inseriti nello zaino non deve superare peso P. Il valore di un riempimento è dato dalla somma dei valori contenuti nello zaino. Una soluzione ottimale è data da un sottoinsieme di oggetti di E il cui peso complessivo non superi P e con valore massimo tra quelli possibili.

In questo caso non possiamo più utilizzare la riduzione precedente: infatti, se tolgo un oggetto e da una soluzione ottima per uno zaino che porta P, *non è detto che quanto rimane sia una soluzione ottima per uno zaino che porta* $P - p_e$. Infatti, utilizzando e potrei ottenere una soluzione migliore per quel peso, e a questo punto non potrei aggiungerlo nuovamente: la dimostrazione per assurdo non funziona più.

In altre parole, la riduzione al problema di peso inferiore in questo caso non si può più utilizzare. Dobbiamo quindi trovare un modo più furbo di ridurre il problema e non c'è altro modo di farlo se non *riducendolo rispetto a* due *parametri*. Riduciamo cioè contemporaneamne rispetto al peso e rispetto al numero di oggetti tra cui scegliere. Vogliamo trovare il valore di una soluzione ottima per tutti gli zaini di peso inferiore a P e *per tutti gli insiemi di oggetti* $\{1, 2, \ldots, j\}$ *con* $j \leq m$.

Supponiamo di avere infatti una soluzione ottima di valore V per uno zaino di peso P che utilizza gli oggetti $1, 2, \ldots, j$. Ci sono due possibilità: o la lista utilizza l'ultimo oggetto j, oppure no. Nel primo caso so di avere una soluzione ottima di valore $V - v_j$ per uno zaino di peso $P - p_j$ usando la lista di oggetti $1, 2, \ldots, j - 1$; nel secondo, so di avere una soluzione ottima per uno zaino di peso P con la lista di oggetti $1, 2, \ldots, j - 1$. In ogni caso, uno dei parametri che definisce il sottoproblema viene ridotto.

È chiaro che a questo punto un vettore non sarà sufficiente: avremo bisogno di una matrice $s[i][j]$ che contiene nella posizione di indici i e j il valore della sottosoluzione ottima per uno zaino di peso i che utilizza al più i primi j oggetti di E (chiaramente, $0 \leq i \leq P$ e $0 \leq j \leq m$). Quando avremo riempito tutta la tabella, la posizione $s[P][m]$ conterrà la soluzione ottima del problema originale.

Anche per questo caso generale, potete trovare ulteriori dettagli ed un esempio completo nella dispensa del Prof. Vigna.

4 Esercizio: varianti dello scheduling di intervalli

Considerate la seguente variante del problema dello scheduling di intervalli.

Avete un insieme $I = \{1, 2, \ldots, n\}$ di richieste; ciascuna richiesta i ha una durata d_i, che denota il tempo richiesto per soddisfarla. Facendo riferimento all'esempio delle lezioni da collocare in un aula, in questo caso l'orario di inizio e fine della lezione non è prefissato, ma si indica soltanto la durata della lezione. Inoltre, avete un tempo massimo T in cui collocare le richieste.

Una soluzione a questo problema è data da un sottoinsieme $S \subseteq I$ tale che la somma delle durate degli elementi di S non superi T (ovvero, c'è tempo sufficiente per soddisfare tutte le richieste di S, in un ordine qualsiasi). La durata di una soluzione è data dalla somma delle durate delle richieste contenute in S, ovvero:

$$\sum_{i \in S} d_i$$

Vogliamo massimizzare il tempo di occupazione della risorsa, ovvero consideriamo ottimali le soluzioni che hanno durata complessiva massima tra la varie soluzioni possibili.

Dovete scrivere un programma che, dato in input un tempo massimo T ed un insieme I di richieste, fornisca in output la durata complessima di una soluzione massimale, insieme ad una qualunque di tali soluzioni ottimali.

Suggerimento: Questo problema può essere riformulato come caso particolare del problema dello zaino...

4.1 Un'altra variante

Aggiungete alla durata di ogni richiesta anche un valore. Allora ciascuna richiesta sarà rappresentata da una coppia (d, v). Il primo elemento della coppia, d, rappresenta una durata, ovvero il tempo richiesto per soddisfare la richiesta, e v è il valore della richiesta. Facendo riferimento all'esempio delle lezioni da collocare in un aula, v può essere il numero di studenti coinvolti. Inoltre, avete sempre un tempo massimo T in cui collocare le richieste.

Una soluzione a questo problema è data da un sottoinsieme $S \subset I$ tale che la somma delle durate degli elementi di S non superi T (ovvero, c'è tempo sufficiente per soddisfare tutte le richieste di S, in un ordine qualsiasi). Il valore di una soluzione S è dato dalla somma dei valori delle richieste contenute in S, ovvero:

$$\sum_{(d,v) \in S} v$$

Dovete scrivere un programma che, dato in input un tempo massimo T ed un insieme I di richieste, fornisca in output il valore massimo fra tutte le possibili soluzioni, insieme ad una qualunque delle soluzioni ottimali (ovvero una soluzione che ha tale valore massimo).

Suggerimento: Anche questo problema può essere riformulato nei termini del problema dello zaino...