Problema dello zaino - knapsack

prof.ssaGrandillo

Dato uno zaino che può sopportare un determinato peso M e dati N oggetti, ognuno dei quali caratterizzato da un peso e un valore, il problema si propone di scegliere quali di questi oggetti mettere nello zaino per ottenere il maggiore valore senza eccedere il peso sostenibile dallo zaino stesso.

Il problema ha 3 formulazioni: problema dello **zaino 0-1**, problema dello zaino con limiti e **senza limiti.** https://it.wikipedia.org/wiki/Problema_dello_zaino

problema dello zaino SENZA limiti

Ogni oggetto può essere inserito nello zaino un numero arbitrario di volte (si suppone che ci siano infiniti esemplari di ogni oggetto). L'Algoritmo in **Programmazione Dinamica bottom-up** <<calcola per il primo oggetto il meglio che si riesce a fare con quell'oggetto su zaini di dimensioni da 0 a M; i risultati vengono memorizzati nel vettore soluzioni. Poi per il secondo oggetto (e per tutti i successivi) si procede allo stesso modo, aggiornando, dove si ottengono dei risultati migliori, i valori massimi per gli zaini di dimensioni da 0 a M>> (*Guida per le Selezioni Territoriali OII del prof.Bugatti* p. 67)

Viene utilizzato il vettore di interi **soluzioni []** per rappresentare in ogni posizione **j** il miglior valore che può contenere uno zaino di capienza **j <= M**.

La valorizzazione del vettore è operata dinamicamente; si inizia a considerare l'oggetto **0** per stabilire quanti esemplari di **peso[0]** possono essere inseriti nello zaino, memorizzando in **soluzioni [j]** :

- **zero**, per **j** < **peso[0]** (in realtà il vettore è già inizializzato a zero e non viene aggiornato cfr. *NOTA*)
- **valore[0],** per le posizioni **j** con **peso[0] <= j < 2*peso[0]** (= **peso[0] + peso[0]**)
- **2*valore[0],** per le posizioni **j** tali che **2*peso[0] <= j < 3*peso[0]** e così via.

Si considera poi l'oggetto **1**: lasciando inalterati i valori delle posizioni con **j** < **peso[1]**

e modificando le sole posizioni con **j** >= **peso[1]** per le quali con l'oggetto **1** si ottiene un valore maggiore.

Alla fine dell'elaborazione, considerati tutti gli N oggetti, la risposta richiesta (il maggior valore per il sacco di capienza M) si troverà in posizione M del vettore **soluzioni**, cioè in **soluzioni[M]**.

```
#include <iostream>
using namespace std;
#include <fstream>
  int peso  [100];
  int valore[100];
  int N, M;
  int soluzioni[101] = {0};
int knapsack_bottom_up(){
   for (int i = 0; i < N; i++){
       for (int  j = 0; j <= M - peso[i] ; j++)

         if ( soluzioni [ j + peso[ i ] ] < soluzioni[ j ] + valore[ i ] )

             soluzioni [ j + peso[ i ] ] = soluzioni[ j ] + valore[ i ];
    }
     return soluzioni[M];
}
int main() {
 ifstream in ("input.txt");
 ofstream out("output.txt");
    in >> N >> M;
    for (int i = 0; i < N; i++)
       in >> peso[i] >> valore[i];
    out << knapsack_bottom_up();
    return 0;
}
```

input.txt **output.txt**

```
4  24          54
26  30
6  7
4  8
10  23
```

N<100 M<=100

nell'esempio: N = 4, M = 24

NOTA: Per ogni oggetto **i** vengono elaborate le posizioni **da 0 + peso[i] ad M** (0+peso[i], 1+peso[i] … M) lasciando inalterate le precedenti perchè in uno zaino di capienza minore di peso[i] non è possible inserire l' oggetto **i**.

Si modifica la posizione **j** + peso[i] di **soluzioni** , cioè **soluzioni [j** + peso[i] **]** , solo se il valore attuale è minore del valore attuale di **soluzioni [j]** incrementato di **valore [i]**.

Problema dello zaino SENZA limiti

Nell' immagine che segue si riportano le configurazioni successive di **soluzioni []** per l'input indicato: la riga **i** riporta il contenuto di **soluzioni** alla fine della **i-esima** iterazione.

Per l'oggetto **0** lo zaino resta vuoto perchè **peso[0]=26 > 24= M**. Per l'oggetto **1** di peso[1]=**6** e valore[1]=**7**, è possibile inserire **4 esemplari** per un valore complessivo di **28** = 4***7**.

Per l'oggetto **2** lo zaino raggiunge per ogni **j** > peso**[1]=4** un maggior valore rispetto al mantenimento di ciascun oggetto **1**; per esempio:

- per j=0 **j+peso[2]** vale 4=0+4 e **soluzioni [j + peso[i]]** cioè **soluzioni [4]** vale **0** mentre **soluzioni [j] + valore[i] = soluzioni[0]+ valore[2]** vale 8=0+8, quindi il valore complessivo può essere migliorato inserendo il primo esemplare dell' oggetto **2**;
- per j=4 **j+peso[2]** vale 8=4+4 e **soluzioni [j + peso[i]]** cioè **soluzioni[8]** vale **7** mentre **soluzioni[j] + valore[i] = soluzioni[** 4 **]+ valore[2]** vale 16=8+8, quindi il valore complessivo può essere migliorato, anche in questo caso, inserendo il secondo esemplare dell' oggetto **2**.

Si procede così per ogni **j <= 20 = M - peso[2]=24-4** e poi per l'oggetto **3** per ogni **j <= 14 = M - peso[3]=24-10**

i / N	pesi	valori	M																							p[i]	v[i]	i	j	M - p[i]	j+pi	s[j+p[i]]	<	s(j)+v(i)		
4			24	28 = 4 oggetti di peso 6 (peso complessivo=24) e valore 7 (valore totale=28)																																
0	26	30																								26	30	0	1-18	-2	il for su j NON viene eseguito, riga 0 resta tutta a zero					
1	6	7		48 = 6 oggetti di peso 4 (peso compl.=24) e valore 8 (valore totale=48)																						6	7	1	da peso 0		a peso 5 : soluzioni[] NON viene modificato					
2	4	8																										1	0	18	6	0	VERA	7	=0+7	7	da 6
3	10	23		54 = 2 oggetti di peso 10 e valore 23 e 1 di peso 4 e valore 8 (valore=23+23+8=54 e peso=10+10+4=24)																								1	5		11	0	VERA	7	=0+7	7	a 11
																									M			1	6		12	0	VERA	14	=7+7	14	da 12
indice j >>>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			1	11		17	0	VERA	14	=7+7	14	a 17
↓ i ↓	contenuto del vettore soluzioni[] alla fine della iterazione i-esima																											1	12		18	0	VERA	21	=14+7	21	da 18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			1	18		24	0	VERA	28	=21+7	28	da 24
1	0	0	0	0	0	0	7	7	7	7	7	7	14	14	14	14	14	14	21	21	21	21	21	21	28												
																										4	8	2	da peso 0		a peso 3 : soluzioni[] NON viene modificato					
2	0	0	0	0	8	8	8	8	16	16	16	16	24	24	24	24	32	32	32	32	40	40	40	40	48			2	0	20	4	0	VERA	8	=0+8	8	da 4
																												2	3		7	7	VERA	8	=0+8	8	a 7
3	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	39	39	46	46	47	47	54			2	4		8	7	VERA	16	=8+8	16	da 8 a 11
																												2	8		12	14	VERA	24	=16+8	24	da 12 a 15
	contenuto del vettore soluzioni[] per l'oggetto 3 alla fine delle iterazioni j da 3.0 a 3.10 , successiva a i=2 e j=20 (2.20)																											2	12		16	14	VERA	32	=24+8	32	da 16 a 19
	j = 0										j + p[i] = 10, s[10]= 16 è minore di 23 = 0+23 = s[j]+v[i] quindi si modifica s[10] cioè s[j+p[i]]																	2	16		20	21	VERA	40	=32+8	40	da 20 a 23
2.20	0	0	0	0	8	8	8	8	16	16	~~16~~	16	24	24	24	24	32	32	32	32	40	40	40	40	48			2	20		24	28	VERA	48	=40+8	48	da 24
3.0	0	0	0	0	8	8	8	8	16	16	23	~~16~~	24	24	24	24	32	32	32	32	40	40	40	40	48												
3.1	0	0	0	0	8	8	8	8	16	16	23	23	24	24	24	24	32	32	32	32	40	40	40	40	48												
3.2	0	0	0	0	8	8	8	8	16	16	23	23	24	24	24	24	32	32	32	32	40	40	40	40	48	10	23	3	da peso 0		a peso 9 : soluzioni[] NON viene modificato					
3.3	0	0	0	0	8	8	8	8	16	16	23	23	24	24	~~24~~	24	32	32	32	32	40	40	40	40	48			3	0	14	10	16	VERA	23	=0+23	23	da 10 a 11
3.4	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	~~24~~	32	32	32	32	40	40	40	40	48			3	2		12	24	falsa	23	=0+23	falsa	NO modif.
3.5	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	32	32	40	40	40	40	48			3	4		14	24	VERA	31	=8+23	31	
3.6	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	32	32	40	40	40	40	48			3	6		16	32	falsa	31	=8+23	falsa	NO modif.
3.7	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	32	32	40	40	40	40	48			3	8		18	32	VERA	39	=16+23	39	
3.8	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	39	32	40	40	40	40	48			3	10		20	40	VERA	46	=23+23	46	
3.9	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	39	39	40	40	40	40	48			3	14		24	48	VERA	54	=31+23	54	
3.10	0	0	0	0	8	8	8	8	16	16	23	23	24	24	31	31	32	32	39	39	46	40	40	40	48												
indice j >>>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			i	j	M - p[i]	j+pi	s[j+p[i]]	<	s(j)+v(i)		

Problema dello zaino 0-1

prof.ssaGrandillo

Ogni oggetto può NON essere inserito nello zaino (**0**) oppure essere inserito una sola volta (**1**, cioè senza ripetizioni).

Si riporta l'Algoritmo in **Programmazione Dinamica** descritto durante il *Corso di Preparazione per la selezione territoriale OII – LUISS 2021 (5^ lezione).*

http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/informatica/all-a/problemazaino_pubblicato.pdf

Dato lo zaino di capacità **W** e gli **N** oggetti di peso **w[i]** e valore **v[i]**, viene utilizzata la matrice **M [i] [j]** dove ogni elemento rappresenta il massimo valore che può essere ottenuto con i **primi i oggetti** (i <= N) in uno **zaino di capacità j** <= W); a fine elaborazione, considerati tutti gli N oggetti, il maggior valore per il sacco di capienza W si troverà in posizione **M [N] [W].**

Per ogni oggetto **i**, nell'elemento **M [i] [j]** :

- si trascrive il valore della riga precedente [i-1] di posizione **j** , per ogni capacità **j** minore del peso **w[i]** dell'oggetto **i**
- per capacità **j** uguali o maggiori del peso **w[i]** si trascrive invece il **massimo** tra:
  - il valore della riga precedente [i-1] in posizione **j**
  - il valore della riga prec. in posizione **j - w [i]** al quale viene aggiunto **v [i]** il valore dell'oggetto **i**

```
#include <iostream>
using namespace std;
#include <fstream>
  int N, W;
  int w[101];              //peso
  int v [101];             //valore
  int M[101][101];         // matrice
int max(int a, int b) {    // oppure  return (a < b) ? b : a;
  if (a < b)       return b;
  else             return a;  }
int knapsack_bottom_up(){
   for (int i = 1;   i <= N;  i++) {
    for (int j = 1; j <= W; j++)
     if ( w [ i ] > j )
         M [ i ] [ j ] =          M [i-1] [ j ];
     else
         M [ i ] [ j ] = max (    M [i-1] [ j ] ,
                                  v [ i ]  + M [i-1] [ j  - w [ i ] ] ) ;
   }
   return M [N] [W];
}
int main(){
 ifstream in ("input.txt");
 ofstream out("output.txt");
   in >> N >> W;

   for (int i=1; i<=N; i++){
      in >> w[i] >> v[i];     //lettura peso e valore oggetto i
      M[i][0] = 0;            //inizializzazione colonna 0
   }

   for (int i=0; i<=W; i++)
      M[0][i] = 0;            //inizializzazione riga 0

   out << knapsack_bottom_up();
   return 0;
}
```

i / N	pesi	valori
4		
1	26	30
2	6	7
3	4	8
4	10	23

M = 24

soluzione: valore **38** =23+8+7 di peso **20** =10+4+6

indice j >>>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	M=24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	0	0	0	0	8	8	8	8	8	8	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
4	0	0	0	0	8	8	8	8	8	8	23	23	23	23	31	31	31	31	31	31	38	38	38	38	38
indice j >>>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

w[i]	v[i]	i	j	w[i]>j	M[i-1][j]	j-w[i]	M[i-1][j-w[i]] +v[i]	MAX	
26	30	1	1-24	VERA	imposta tutti a ZERO da riga i-1=1-1= 0				
6	7	2	1-5	VERA	viene trascritto M [i-1] [j] = M[1][j] = 0				
		2	6	FALSA	0	0	0	7	7	
		2	7	FALSA	0	1	0	7	7	
		2	24	fino a	0	18	0	7	7	
4	8	3	1-3	VERA	viene trascritto M [i-1] [j] = M[2][j] = 0				
		3	4	FALSA	0	0	0	8	8	
		3	9	fino a	0	5	0	8	8	
		3	10	FALSA	7	6	7	15	15	=7+8
		3	24	fino a	7	20	7	15	15	
10	23	4	10	FALSA	15	0	0	23	23	=0+23
		4	11	FALSA	15	1	0	23	23	
		4	14	FALSA	15	4	8	31	31	=8+23
		4	20	FALSA	15	10	15	38	38	=15+23
		4	24	FALSA	15	14	15	38	38	