



UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dispense del Corso di Laboratorio di
Fondamenti di Informatica II e Lab

Esercitazione 06: Algoritmi Greedy

Ultimo aggiornamento: 10/04/2019

Problema dello Zaino (*Knapsack Problem*)

Dato un insieme di n oggetti ciascuno dotato di un peso $w[i]$ e di un valore $v[i]$ e dato un peso massimo p , si vuole determinare il sottoinsieme di oggetti tali che:

- la somma dei pesi degli oggetti selezionati sia minore o uguale al peso massimo:

$$\sum_{i=1}^n x[i] * w[i] \leq p$$

dove $x[i]$ indica se l'oggetto i -esimo è stato o meno inserito nello zaino.

- il valore degli oggetti selezionati sia massimizzato secondo l'euristica greedy che utilizza come criterio di appetibilità il *valore specifico* degli oggetti definito come $v[i]/w[i]$.

Algoritmi Greedy: Zaino Discreto 1/4

- Esercizio 1 (Zaino Discreto):

Nel file `zainodiscreto.c` si implementi la definizione della funzione `ZainoDiscreto`:

```
oggetto *ZainoDiscreto(oggetto *o, int n, int p, int *cs)
```

La funzione prende in input un vettore di oggetti `o`, la sua dimensione `n`, il peso massimo caricabile nello zaino `p` e un puntatore ad una variabile di tipo `int` (`cs`) in cui si dovrà inserire il numero di oggetti caricati nello zaino. La funzione deve scegliere quali oggetti, tra quelli disponibili, caricare nello zaino. Si utilizzi a tale scopo la funzione di costo descritta nella slide precedente. La funzione deve ritornare un vettore di oggetti allocato dinamicamente nell'heap e contenente gli oggetti scelti.

Algoritmi Greedy: Zaino Discreto 2/4

- Un oggetto deve essere caricato per intero o non caricato affatto;
- Un oggetto può essere caricato una sola volta;
- Non si possono fare assunzioni sull'ordinamento del vettore di input o;
- Il tipo `oggetto` è definito come segue:

```
typedef struct {
    int p;
    int v;
}oggetto;
```

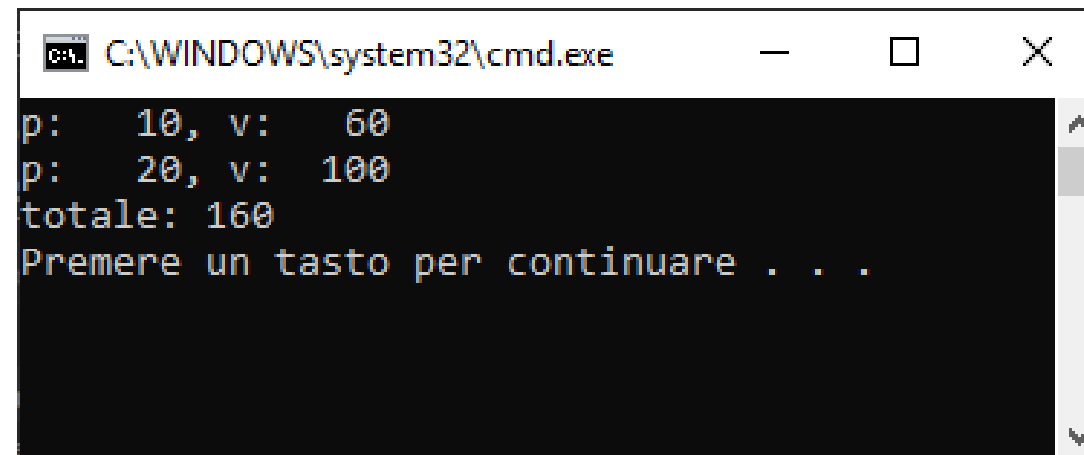
dove p indica il peso e v il valore dell'oggetto.

Algoritmi Greedy: Zaino Discreto 3/4

Nel `main()`, chiamare la funzione `ZainoDiscreto` e mostrare su standard output la soluzione, ovvero la sequenza di oggetti da caricare e il valore totale degli oggetti caricati come nell'esempio seguente:

$o = \{ \{.p = 20, .v = 100\}, \{.p = 10, .v = 60\}, \{.p = 30, .v = 120\} \}$ e $p = 50$

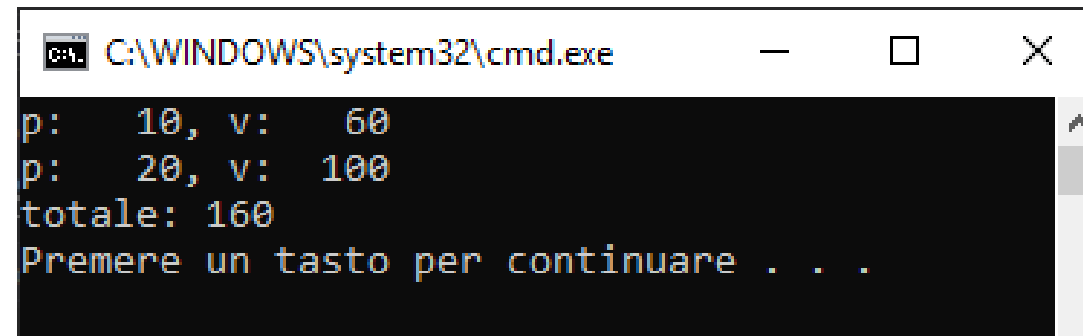
l'output dovrà essere



```
C:\WINDOWS\system32\cmd.exe
p: 10, v: 60
p: 20, v: 100
totale: 160
Premere un tasto per continuare . . .
```

Algoritmi Greedy: Zaino Discreto 4/4

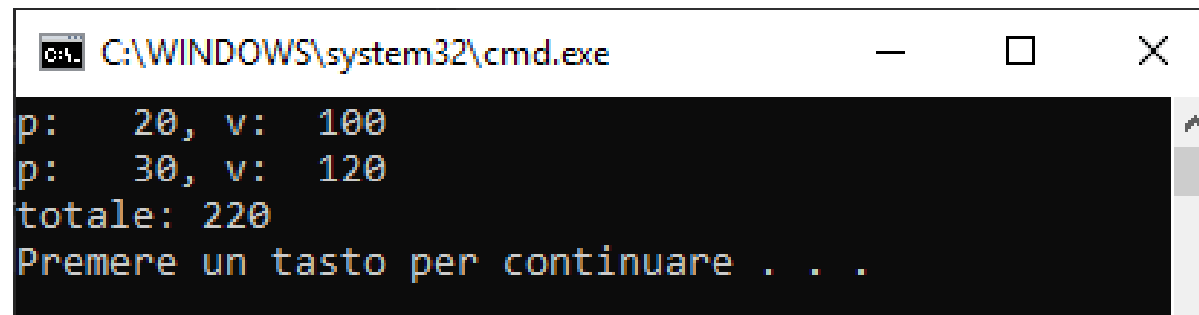
Si noti che utilizzando la funzione di costo definita nelle slide precedenti l'algoritmo greedy trova la soluzione localmente ottima:



```

C:\WINDOWS\system32\cmd.exe
p: 10, v: 60
p: 20, v: 100
totale: 160
Premere un tasto per continuare . . .
    
```

che non corrisponde però a quella globalmente ottima:



```

C:\WINDOWS\system32\cmd.exe
p: 20, v: 100
p: 30, v: 120
totale: 220
Premere un tasto per continuare . . .
    
```

Algoritmi Greedy: Zaino Continuo 1/4

- Esercizio 1 (Zaino Continuo):

Nel file `zainocontinuo.c` si implementi la definizione della funzione `ZainoContinuo`:

```
oggetto *ZainoContinuo(oggetto *o, int n, int p, int *cs)
```

La funzione prende in input un vettore di oggetti `o`, la sua dimensione `n`, il peso massimo caricabile nello zaino `p` e un puntatore ad una variabile di tipo `int` (`cs`) in cui si dovrà inserire il numero di oggetti caricati nello zaino. La funzione deve scegliere quali oggetti, tra quelli disponibili, caricare nello zaino. Si utilizzi a tale scopo la funzione di costo descritta nella slide precedente. La funzione deve ritornare un vettore di oggetti allocato dinamicamente nell'heap e contenente gli oggetti scelti.

Algoritmi Greedy: Zaino Continuo 2/4

- In questo caso possiamo caricare anche solo una parte dell'oggetto;
- Un oggetto può essere caricato una sola volta;
- Non si possono fare assunzioni sull'ordinamento del vettore di input o;
- Il tipo `oggetto` è definito come segue:

```
typedef struct {
    int p;
    int v;
    double x;
}oggetto;
```

dove p indica il peso e v il valore di un oggetto. x è un valore $]0,1]$ che indica in che quantità l'oggetto è stato caricato nello zaino.

Algoritmi Greedy: Zaino Continuo 3/4

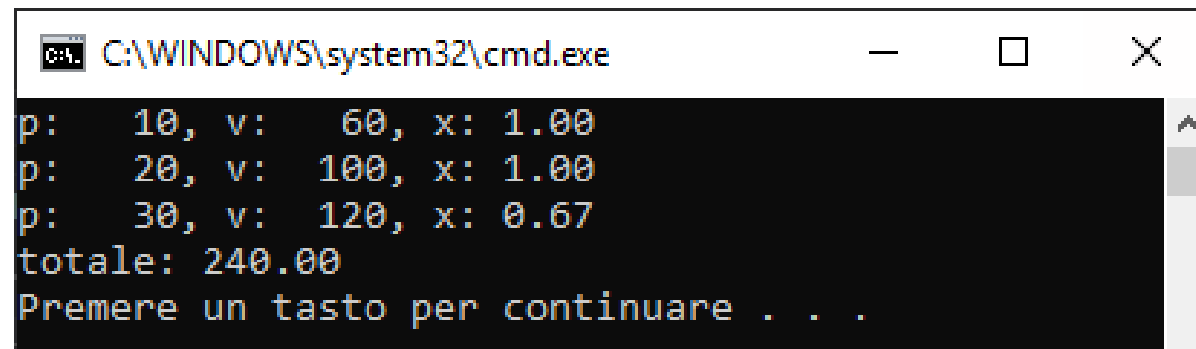
- Si noti che il campo x della `struct oggetto` non ha alcun significato per gli oggetti del vettore di input o , ma servirà soltanto per il vettore di oggetti effettivamente caricati nello zaino (vettore di output).
- Si è scelto di introdurre comunque x nella `struct oggetto` per semplificare l'implementazione.
- Si noti inoltre che nel vettore di output non devono essere inseriti gli oggetti che non sono caricati, ovvero non devono esserci oggetti per cui $x = 0.0$

Algoritmi Greedy: Zaino Continuo 4/4

Nel `main()`, chiamare la funzione `ZainoContinuo` e mostrare su standard output la soluzione, ovvero la sequenza di oggetti da caricare e la corrispondente quantità e il valore totale caricato, come nell'esempio seguente:

$o = \{ \{.p = 20, .v = 100\}, \{.p = 10, .v = 60\}, \{.p = 30, .v = 120\} \}$ e $p = 50$

l'output dovrà essere



```
C:\WINDOWS\system32\cmd.exe
p:  10, v:  60, x: 1.00
p:  20, v: 100, x: 1.00
p:  30, v: 120, x: 0.67
totale: 240.00
Premere un tasto per continuare . . .
```

In questo caso la soluzione greedy corrisponde a quella ottima.