UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

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

Silvia Cascianelli, Federico Bolelli

Esercitazione 05: Algoritmi Greedy

Ultimo aggiornamento: 29/04/2020

Algoritmi Greedy: Monete

- Esercizio 1 (Monete):

Nel file `monete.c` si implementi la definizione della funzione `Monete`:

```
extern int* Monete(int *t, int size, int b);
```

La funzione accetta come parametri un array `t` di valori che rappresentano i tagli di monete disponibili (ad esempio 50, 20, 10, 5, 2 e 1 centesimo/i), la sua dimensione `size`, e un budget `b` espresso in centesimi. La funzione deve trovare il numero minimo intero di monete necessarie per formulare il budget, allocare dinamicamente un array delle stesse dimensioni di `t` dove scrivere il quantitativo di monete di ogni tipo necessarie per raggiungere il budget .

Algoritmi Greedy: Monete

- La funzione deve quindi restituire l'array precedentemente allocato o `null` se il budget `b` è minore o uguale a 0.
- Si utilizzi a tale scopo un **algoritmo greedy** basato sull'opportuna funzione di costo.
- Si supponga di disporre di infinite monete per ogni taglio e che i tagli disponibili permettano di costruire il budget.
- Si assuma che l'array `t` preso in input dalla funzione sia ordinato in maniera decrescente.

Algoritmi Greedy: Monete

- Ad esempio, con

  `t = {50,20,10,5,2,1} e b = 126`

- L'output della funzione dovrà essere il vettore

  `{2,1,0,1,0,1}`

- In questo caso la soluzione *greedy* ci dice che il numero minimo di monete necessarie per costruire il budget è 5, due pezzi da 50c, uno da 20c, uno da 5c e uno da 1c.
- Si scriva un opportuno `main()` di prova per testare la funzione.
- In questo esempio la soluzione *greedy* corrisponde a quella ottima. Riuscite a trovare un esempio in cui la soluzione dell'algoritmo *greedy* proposto è peggiore di quella ottima?

Algoritmi Greedy: Gioielli

- Esercizio 2 (Gioielli):

Nel file `gioielli.c` si implementi la definizione della funzione `Gioielli`:

```
extern Gioiello* Gioielli(const char* filename, float b, int *ret_size);
```

dove `filename` è il nome di un file di testo e `b` rappresenta il budget. La procedura deve leggere da file i gioielli disponibili e selezionare quelli da comprare in modo da massimizzare il peso complessivo dei gioielli acquistati rispettando il budget. Un gioiello si deve comprare per intero, senza frazionamenti. Un gioiello si può acquistare una sola volta. Si utilizzi a tale scopo un algoritmo *greedy* con l'**opportuna funzione di costo**.

Algoritmi Greedy: Gioielli

- Nel file `gioielli.h` si definisca la struttura dati `Gioiello`:

```
typedef struct {
    int codice;
    float peso;
    float prezzo;
} Gioiello;
```

- dove `codice` identifica il gioiello, `peso` il suo peso in grammi e `prezzo` il suo prezzo di vendita in euro. Il file `filename` memorizza i dati per righe. Ogni riga del file contiene il codice, il peso e il prezzo di un singolo gioiello separati da spazi:

```
<codice><spazio><peso><spazio><prezzo><a capo>
```

- Si assuma che il file sia formato correttamente.

Algoritmi Greedy: Gioielli

- La funzione deve ritornare un vettore di `Gioiello` allocato dinamicamente e contenente i gioielli che devono essere acquistati per massimizzare il peso complessivo rispettando il budget o NULL se non è possibile aprire il file.
- Al termine della funzione, la dimensione del vettore allocato dinamicamente dovrà essere salvata nell'area di memoria puntata da `ret_size`.

- Ad esempio, dato un budget di 121 € e il file di gioielli seguente:

```
1 12 100
2 10 21
3 25 120
```

- Il vettore ritornato dovrà contenere i gioielli (non necessariamente in questo ordine):

```
1 12 100
2 10 21
```

- Si scriva un opportuno `main()` di prova per testare la procedura.
- Come si può notare, in questo caso la soluzione *greedy* non corrisponde all'ottimo assoluto.

Modalità di Consegna

- Per questa esercitazione dovrete consegnare tutti e due gli esercizi utilizzando un sistema di sottomissione online simile a quello che avete usato durante il corso di Fondamenti di Informatica I.
- **Collegatevi al sito https://aimagelab.ing.unimore.it/OLJ2/esami e fate il login utilizzando le vostre credenziali shibboleth di UNIMORE.**
- Selezionate *Esercitazione Greedy I,* aprite il link dell'esercizio di cui volete fare la sottomissione e incollate il codice nei box relativi ai rispettivi file. **Non dovete caricare il `main()`.**
- Assicuratevi tuttavia di scrivere il `main()` in Visual Studio per verificare se quello che avete fatto funziona prima di caricare la soluzione.

Modalità di Consegna

- Causa manutenzione, il sistema di sottomissione potrebbe avere qualche problema in questi giorni.
- Se doveste visualizzare un errore come quello che segue riprovate a collegarvi per eseguire la sottomissione più tardi:

Microsoft Access Database Engine error '80004005'

Unspecified error

/OLJ2/esami/inc_dbConnection.asp, line 81