Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

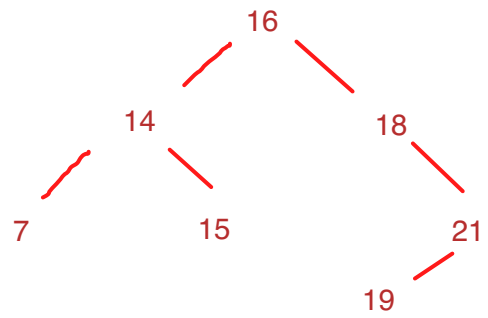
Compitino del 16 gennaio 2017

TEMPO DISPONIBILE: 1 ora e 40 minuti

Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

1. Considerate un albero binario di ricerca ottenuto inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri in un albero inizialmente vuoto: `16 18 21 14 7 19 15`

(a) Disegnate l'albero



(b) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in ampiezza

16 14 18 7 15 21 19

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine anticipato

16 14 7 15 18 21 19

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine simmetrico

7 14 15 16 18 19 21

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine posticipato

7 15 14 19 21 18 16

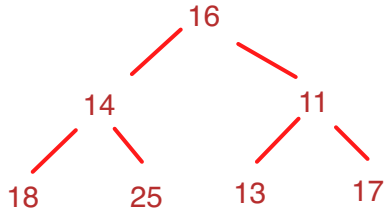
(f) L'albero ottenuto è un albero AVL? No

(g) Se avere risposto SI, nel riquadro a destra scrivete la definizione di albero AVL; altrimenti scrivete il valore contenuto in un nodo per il quale la condizione di bilanciamento degli alberi AVL non sia verificata.

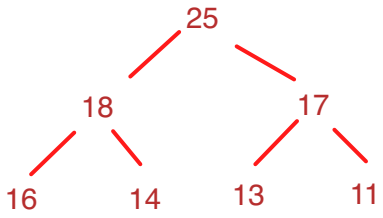
18

2. Considerate l'albero binario rappresentato dal vettore posizionale contenente la seguente sequenza di numeri: `16 14 11 18 25 13 17`

(a) Disegnate l'albero



(b) Disegnate l'albero che si ottiene trasformando l'albero ottenuto al punto (a) in uno heap (prima fase di *Heapsort*)



(c) Scrivete la sequenza di valori contenuti nel vettore posizionale che rappresenta l'albero ottenuto al punto (b)

25 18 17 16 14 13 11

3. Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

x	$f(x)$
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7
j	7
k	7
l	8
m	9

x	$f(x)$
n	10
o	10
p	11
q	12
r	12
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

Siano h e g la funzioni che trasformano ogni parola k sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ negli interi che si ottengono applicando f al primo carattere e all'ultimo carattere di k, rispettivamente. Ad esempio $h(\texttt{gatto}) = 6$, $g(\texttt{gatto}) = 10$. Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
leone gazzella giaguaro gorilla pantera
gufo bisonte coniglio
```

Come funzione hash utilizzate h. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot (1 + g(k)) \,) \bmod 16$$

0	bisonte
1	giaguaro
2	coniglio
3	
4	
5	
6	gazzella
7	gorilla
8	leone
9	
10	
11	pantera
12	gufo
13	
14	
15	

4. Si devono spedire delle scatole voluminose collocandole all'interno di alcuni bauli. Le scatole sono più piccole dei bauli, tuttavia il peso del contenuto di ciascun baule non può superare un limite fissato. Il numero di bauli disponibili supera il numero delle scatole. Tuttavia, poiché le spese di spedizione sono proporzionali al numero di bauli spediti, si vorrebbe ridurre al minimo il numero di bauli utilizzati. Progettate un algoritmo per la soluzione del problema mediante una strategia basata sull'uso della tecnica greedy.

Alcune semplificazioni: I bauli sono tutti uguali: hanno forma cubica, con lunghezza del lato interno L. Tutte le scatole hanno una base quadrata il cui lato è lungo poco meno di L. Pertanto vengono collocate all'interno dei bauli una sopra l'altra. Ad esempio, se $L = 50$, in un baule si potrebbero collocare 2 scatole di altezza 20 e una di altezza 5, purché il loro peso totale non superi il limite fissato, lasciando un po' di spazio vuoto.

Riassumendo:
Input: Scatole $c_1, c_2, \ldots, c_n$, con altezza h_i e peso p_i di ciascuna scatola; lunghezza L del lato dei bauli; peso totale ammissibile P all'interno di ciascun baule (tutti i dati sono numeri interi). Si ipotizzi che l'elenco delle scatole sia fornito in ordine non crescente di altezza, cioè $h_1 \geq h_2 \geq \ldots \geq h_n$. Inoltre $h_i \leq L$ e $p_i \leq P$, per $i = 1, \ldots, n$.
Output: Un elenco di bauli da utilizzare, con l'indicazione per ciascuno di essi delle scatole da inserire.
Vincoli: L'altezza totale delle scatole collocate in ciascun baule non può superare L, il peso totale delle scatole collocate in ciascun baule non può superare P.

Esempio: Date 6 scatole con i pesi e le altezze specificate nella tabella a destra, $L = 10$, $P = 80$, una possibile soluzione è:
`baule 1:` c_1 c_6 – `baule 2:` c_2 – `baule 3:` c_3 c_5 – `baule 4:` c_4

	c_1	c_2	c_3	c_4	c_5	c_6
h	8	8	5	4	3	2
p	20	60	40	20	20	30

Cosa si richiede:

(a) Descrivete *sinteticamente a parole* e poi ad alto livello in pseudocodice, un algoritmo che utilizzi la tecnica greedy per riempire ciascun baule.

(b) Ricavate una stima del tempo totale di calcolo in funzione del numero n di scatole.

(c) Indicate se l'algoritmo trova sempre una soluzione che utilizza il minimo numero di bauli necessario oppure no, motivando la risposta. In particolare, in caso di risposta negativa presentate un esempio in cui non viene trovato l'ottimo (indicate la soluzione ottenuta dall'algoritmo che avete scritto e una soluzione ottima).

Traccia della soluzione

Una possibile soluzione consiste nel procedere "per bauli", applicando l'algoritmo greedy per riempire ogni baule necessario:
- inizialmente si scandisce l'intera sequenza di scatole dalla prima all'ultima collocando nel primo baule una scatola se soddisfa i vincoli (la sua altezza il suo peso, aggiunti a quelli delle scatole già collocate, non superano L e P, rispettivamente).
- si ripete, utilizzando la sequenza di scatole rimaste, per riempire il secondo baule, e così via, sino ad esaurire tutte le scatole.

Una soluzione alternativa consiste nel procedere "per scatole": si considera ogni scatola cercando un baule, tra quelli già parzialmente riempiti, al quale possa essere aggiunta. Se non si trova nessun baule adatto, se ne utilizza uno nuovo. Dunque, la prima scatola viene collocata in un baule nuovo. La seconda viene collocata nel primo baule, se, insieme alla prima, soddisfa i vincoli; altrimenti viene collocata in un baule nuovo, e così via.

In entrambe le soluzioni si può scrivere l'algoritmo mediante due cicli innestati (esterno sui bauli, interno sulle scatole, nel primo caso; viceversa nel secondo).

Il tempo di esecuzione è quadratico in n, e il caso peggiore si raggiunge quando il numero di bauli necessario è pari al numero di scatole.

L'algoritmo non sempre trova l'ottimo. Esempio con 4 scatole di altezze e pesi in tabella, L=15, P=13:

	c_1	c_2	c_3	c_4
h	10	5	3	2
p	9	3	10	4

Soluzione ottima: $\{c_1, c_4\}$ $\{c_2, c_3\}$

Soluzione di entrambi gli algoritmi: $\{c_1\ c_2\}$ $\{c_3\}$ $\{c_4\}$