Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

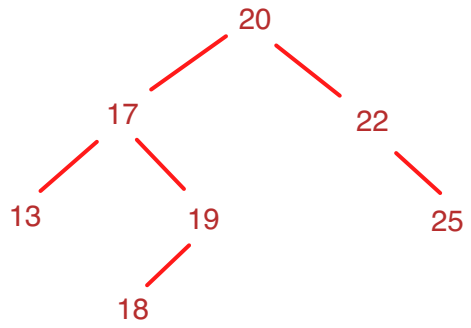
Prova scritta del 3 febbraio 2017

TEMPO DISPONIBILE: 2 ore

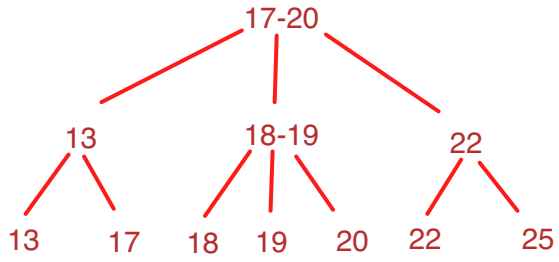
Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere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

1. Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `20 22 13 17 19 25 18`

(a) Disegnate l'albero AVL



(b) Disegnate l'albero 2-3



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

20 17 13 19 18 22 25

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

13 17 18 19 20 22 25

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

13 18 19 17 25 22 20

(f) L'albero AVL ottenuto al punto (a) è perfettamente bilanciato?

No

(g) Se avere risposto SI, nel riquadro sottostante scrivete la definizione di albero perfettamente bilanciato; altrimenti scrivete il valore contenuto in un nodo per il quale la condizione di bilanciamento perfetto non sia verificata.

20

2. Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`423 451 516 491 333 243 216 126 321`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

243 321 126 216 333 423 491 516 451

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

516 491 423 451 333 243 216 126 321

(c) Supponete di ordinare la sequenza mediante l'algoritmo `radixSort`. Indicate il contenuto dell'array dopo le prime due iterazioni del ciclo principale dell'algoritmo.

516 216 321 423 126 333 243 451 491

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

Sia h la funzione che trasforma ogni parola k sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando f al primo carattere di k e g la funzione che trasforma ogni parola nel più piccolo numero primo maggiore o uguale della lunghezza di k. Ad esempio $h(\texttt{gatto}) = 6$, $g(\texttt{gatto}) = 5$, $h(\texttt{granchio}) = 6$, $g(\texttt{granchio}) = 11$.
Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
leone cavallo gazzella pantera lepre
giaguaro pitone granchio
```

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

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

0	
1	giaguaro
2	cavallo
3	
4	
5	
6	gazzella
7	
8	leone
9	pitone
10	
11	pantera
12	granchio
13	lepre
14	
15	

4. L'eccentrico sindaco di una città di cui non si può riportare il nome, vuole creare un servizio centrale di aromaterapia a beneficio degli abitanti. Il progetto prevede di diffondere in ogni abitazione della città delle sostanze aromatiche, distribuite tramite un sistema di tubazioni collocate sotto tutte le strade, ricalcando esattamente la mappa stradale della città.

- Le tubazioni *primarie*, tutte connesse tra loro, scorreranno sotto alcuni tratti di strada in modo che ciascun incrocio sia raggiunto, nel sottosuolo, da almeno una di esse.
- Le tubazioni *secondarie* saranno collocate sotto i tratti di strada non serviti da tubazione primarie. Ogni tubazione secondaria sarà collegata alle tubazioni primarie che si trovano sotto i due incroci che delimitano il tratto di strada da essa servito.

Al fine di ridurre i costi di realizzazione del progetto, è necessario trovare una soluzione che minimizzi la lunghezza totale delle tubazioni primarie.

(a) Spiegate come il problema possa essere descritto e formalizzato in termini di grafi.

(b) Progettate un algoritmo che determini l'insieme delle strade che devono essere attraversate da tubazioni primarie e l'insieme delle strade che devono essere attraversate da tubazioni secondarie: descrivete l'algoritmo a parole e poi, ad alto livello, in pseudocodice. *È opportuno ispirarsi a uno degli algoritmi presentati nel corso.*

(c) Discutete una possibile rappresentazione del grafo utilizzato e una corrispondente implementazione dell'algoritmo, fornendo una stima dei tempi di calcolo.

Note:

- Un tratto di strada è una parte di strada delimitata da due incroci e senza incroci all'interno. La fine di una strada a fondo chiuso è considerata un incrocio.
- Il problema richiede di minimizzare la lunghezza totale delle tubazioni primarie. Nella realtà (ad esempio reti per distribuzione acqua o gas) questi problemi devono essere risolti considerando altri parametri che influiscono, ad esempio, sulla scelta dei diametri delle tubazioni utilizzate nei vari tratti di strada.

Traccia di una possibile soluzione

(a) La mappa della città può essere rappresentata mediante un grafo G=(V,E) non orientato nel quale l'insieme dei vertici V rappresenta l'insieme degli incroci, l'insieme degli archi rappresenta i tratti di strada. Una funzione peso w associa ad ogni arco la lunghezza del corrispondente tratto di strada.
In base ai vincoli del problema, l'insieme delle tubazioni primarie sarà un albero ricoprente di peso minimo di G, mentre l'insieme delle tubazioni secondarie è l'insieme di tutti gli archi rimanenti.

(b) L'insieme delle tubazioni primarie P può essere ottenuto mediante l'algoritmo di Kruskal oppure mediate l'algoritmo di Prim.
L'algoritmo di Kruskal opera come segue:
- ordina l'insieme degli archi (tratti di strada) per lunghezza non decrescente
- analizza gli archi nell'ordine ottenuto aggiungendolo a P (inizialmente vuoto) un arco se non forma cicli con quelli già in P
Per ottenere anche l'insieme delle tubazioni secondarie S, quando un arco non viene aggiunto a P lo si aggiunge a S (anch'esso all'inizio sarà vuoto).

```
ALGORITMO Kruskal (grafo pesato G=<V,E,w>) -> (insieme archi, insieme archi)
  ordina E in base al peso (non decrescente)
  P <- insieme vuoto //tubazioni primarie
  S <- insieme vuoto //tubazioni secondarie
  FOR EACH arco (u,v) in E, in ordine di costo DO
    IF u e v non sono connessi da un cammino di archi in P THEN
      aggiungi (u, v) a P
    ELSE aggiungi (u, v) a S
  RETURN (P, S)
```

(c) Si può rappresentare il grafo utilizzando una lista d'archi che può essere copiata in un array e ordinata. (Si può procedere in maniera analoga con altra rappresentazioni.)
L'algoritmo può essere implementato utilizzando la tecnica union-find...

Attenzione a non confondere il problema di determinare il minimo albero di ricoprimento, con il problema di determinare i cammini minimi in un grafo o con ik probl ma del commesso viaggiatore!