Informatica 3 – prova intermedia del 3 Luglio 2002

Nome _____ **(stampatello)**

Cognome _____ **(stampatello)**

Matr _____

---

spazio per il docente

Punteggi

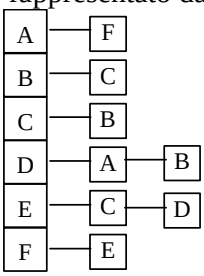
1) ___/2

2) ___/4

3) ___/6

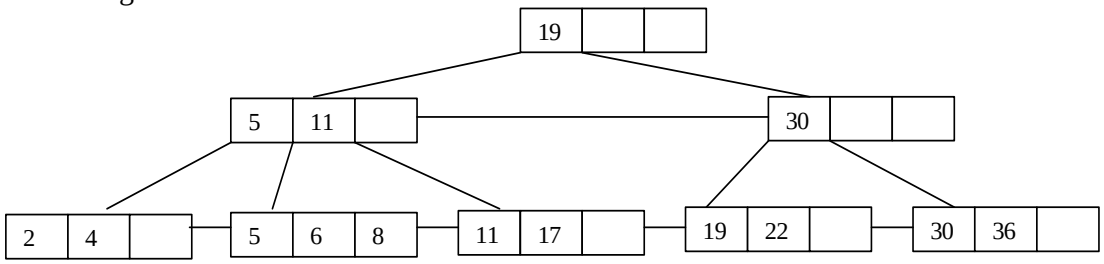
4) ___/8

Esercizio 1 (2 punti)
Considerando il grafo orientato (*directed*) rappresentato dalla seguente lista delle adiacenze,



1. disegnare il grafo;
2. dire se esso contiene cicli, indicando quali in caso affermativo;
3. fornire la rappresentazione del grafo come matrice di adiacenza;
4. considerando il grafo che costituisce la versione non orientata del grafo dato, dire se esso è connesso.

Esercizio 2 (4 punti)
Dato il seguente albero:



a) Di che tipo e ordine è?

b) Si indichi una sequenza di inserimenti che, partendo dall'albero vuoto, porta all'albero indicato, mostrando gli alberi intermedi ottenuti a seguito delle operazioni di *split*.

Esercizio 3 (6 punti)
Nel gioco del SuGiu un giocatore deve indovinare un numero, compreso in un intervallo prefissato, attraverso una serie di tentativi. A ogni tentativo il giocatore indica un valore e un interlocutore che conosce il numero da indovinare dà una delle seguenti tre risposte: SI se il numero è stato indovinato, SU se il numero da indovinare dal giocatore è *superiore* al numero indicato, GIU se è *inferiore*. Tratteggiare l'implementazione di un programma (e.g., un metodo statico Java) che, ricevendo come parametri in ingresso gli estremi inferiore e superiore dell'intervallo di possibili valori per il numero da indovinare, riesca a trovarlo con un numero di tentativi minimo (si assuma che il programma possa fare la parte del giocatore, scrivendo sull'output standard i suoi tentativi e leggendo le indicazioni dell'interlocutore dall'input standard).

static void suGiu (int inf, int sup)

Valutare asintoticamente la complessità di calcolo di tale programma (numero di tentativi massimo che il programma fa nel caso pessimo prima di indovinare il numero) come funzione dell'ampiezza dell'intervallo di valori possibili per il numero da indovinare.

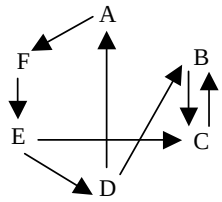
Esercizio 4 (8 punti)

Per un dato albero binario *T*, si definisce Lunghezza dei Cammini Interni, indicata con LCI(*T*), la somma delle lunghezze dei cammini che vanno dalla radice ai nodi interni dell'albero; si definisce Lunghezza dei Cammini Esterni, indicata con LCE(*T*), la somma delle lunghezze dei cammini che vanno dalla radice alle foglie dell'albero.

1. Indicare la relazione esistente, in un albero binario *T* pieno e con *n* nodi interni, tra LCE(*T*), LCI(*T*) ed *n*.
2. Dimostrare tale relazione (suggerimento: operare per induzione).

Soluzioni

Es1
Soluzione
1.



2. Il grafo contiene i cicli AFEDA e BCB

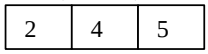
3.

	A	B	C	D	E	F
A	0	0	0	0	0	1
B	0	0	1	0	0	0
C	0	1	0	0	0	0
D	1	1	0	0	0	0
E	0	0	1	1	0	0
F	0	0	0	0	1	0

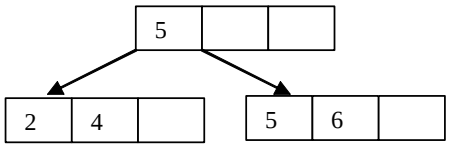
4. Si, è connesso.

Es 2
a) B+ di ordine 4

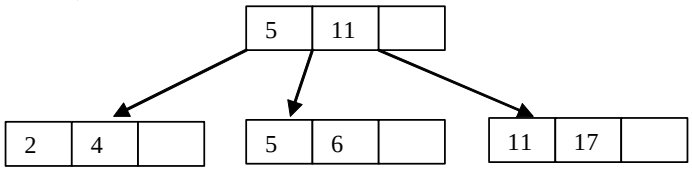
b)
ins 2, ins 4, ins 5:



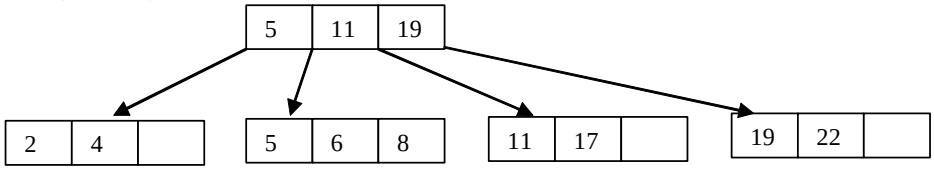
ins 6:



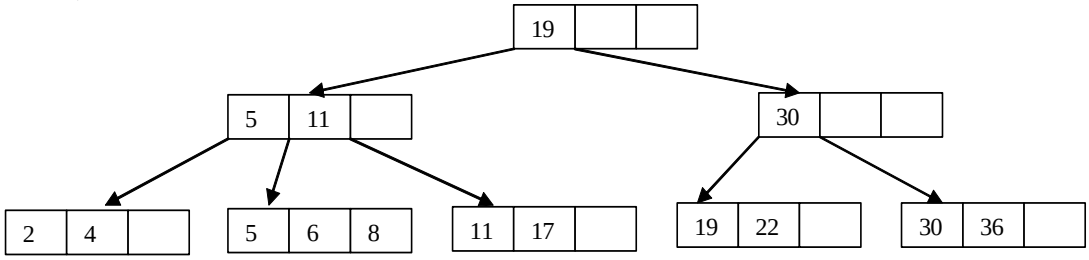
ins 11, ins 17:



ins 8, ins 19, ins 22:



ins 30, ins 36:



NB
la soluzione non è univoca: ad es. si può inserire 8 in una qualsiasi fase successiva a quella mostrata (ma non precedente), oppure permutare alcuni blocchi di inserimenti (come 2 e 4).

Es 3

Soluzione: la strategia migliore è di simulare una ricerca binaria, mantenendo un intervallo di valori all'interno del quale si assume che si trovi il numero da indovinare e utilizzando le indicazioni dell'interlocutore per dimezzare la dimensione di tale intervallo a ogni tentativo. La complessità asintotica di tale programma è logaritmica rispetto all'ampiezza dell'intervallo di valori ammissibili.

Bozza di programma:

```
static void suGiu (int inf, int sup) {
        int med;
        String risposta;

        do {
                med = (inf + sup) / 2;
                scrivi med;
                leggi risposta;
                if (risposta = GIU)
                        sup = med – 1;
                else if (risposta = SU)
                        inf = med + 1;
        } while (risposta != SI);
}
```

Es 4

1. In un albero *T* binario pieno LCE(T) = LCI(T) + 2n
2. Dimostrazione per induzione sul numero *n* dei nodi interni.

Caso Base: Un albero con la sola radice ha n=0, LCI=LCE=0.

Induzione: In un albero T pieno con k nodi interni, sia LCI(T)=h ed LCE(T)=h+2k. Otteniamo un albero T' pieno con k+1 nodi interni aggiungendo due figli a una delle foglie, che supponiamo (senza perdere in generalità) stia a profondità j. Nel nuovo albero un "cammino esterno" di lunghezza j è diventato un "cammino interno", e ci sono due nuovi "cammini esterni" di lunghezza j+1. Valgono allora le seguenti relazioni.

LCI(T') = LCI(T) + j = h + j

LCE(T') = LCE(T) – j + 2 * (j+1) =

= h + 2*k – j + 2*j + 2 =

= h + 2*k + j + 2 =

= h + j + 2*(k + 1) =

= LCI(T') + 2*(#nodi interni)