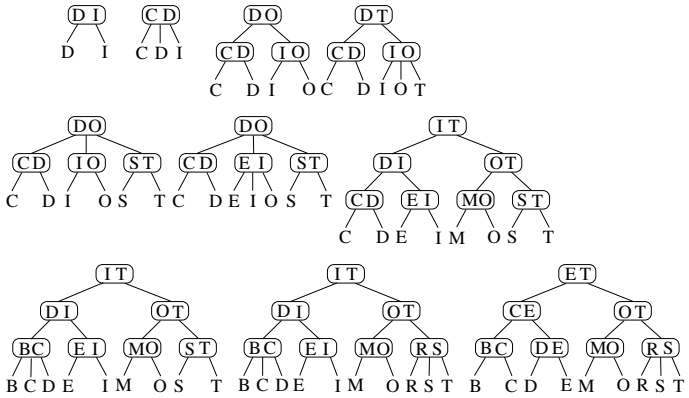
Algoritmi e Strutture Dati
10 settembre 2018

Nome:	.....
Cognome:	.....
Matricola:	.....

1. Partite da un albero 2-3 con i soli valori **D** e **I**, ed eseguite 8 inserimenti in sequenza, relativi ai caratteri **C,O,T,S,E,M,B,R** (disegnate l'albero risultante al termine di ciascun inserimento). In ultimo, cancellate la **I** e mostrate l'albero finale.
Sol.



2. Come è possibile rappresentare un albero con radice ordinato tramite un albero binario? Dato un albero con radice ordinato di grado k e altezza h, qual è l'altezza massima dell'albero binario che lo rappresenta? Motivate la risposta.

Sol. La corrispondenza tra albero con radice ordinato e albero binario si ottiene facendo in modo che, per ogni nodo v dell'albero binario, il figlio sinistro e il figlio destro di v corrispondano al primo figlio di v (nell'albero ordinato) e al più piccolo tra i fratelli maggiori di v (quello immediatamente a destra). Se il grado dell'albero è k, l'ultimo dei nodi del primo livello si troverà al massimo al livello k dell'albero binario. Similmente, l'ultimo dei nodi di livello 2 (nell'albero ordinato), si troverà al massimo al livello $2k$ dell'albero binario. In generale, L'altezza dell'albero binario sarà al massimo hk.

3. In quali casi gli algoritmi di ordinamento per inserzione (Insertionsort) e a bolle (Bubblesort - versione adattiva) hanno prestazioni lineari (ovvero complessità $O(n)$)?

Sol. Si veda il rapporto tra numero di inversioni (in una sequenza) e numero di confronti eseguiti (slide del corso). In particolare, l'insertionsort ha prestazioni lineari se il numero totale di inversioni della sequenza di ingresso è lineare, mentre il bubblesort adattivo ha prestazioni lineari se il numero di inversioni associato a ciascun elemento della sequenza da ordinare è limitato da una costante C.

4. Illustrate il funzionamento delle operazioni di inserimento e cancellazione in una tabella hash dinamica con concatenazioni separate.

Sol. Si vedano i lucidi del corso e il paragrafo 9.3 delle dispense.

5. Considerate l'equazione di ricorrenza

$$C(i,j) = 2C(i,j-1) - C(i-1,j)$$

insieme alle condizioni iniziali

$$C(i,0) = i \quad C(0,j) = j^2$$

e scrivete una funzione che riceve in ingresso due interi m, n (entrambi non negativi) e calcola $C(m, n)$ in maniera efficiente (tramite programmazione dinamica).

Sol. Ispirandoci alla programmazione dinamica, procediamo utilizzando una tabella `T` in cui memorizzare i risultati, ovvero `T[i][j]`$=C(i, j)$. La dimensione della tabella in funzione dell'input (i due interi m, n) è di immediato calcolo: servono $m + 1$ righe e $n + 1$ colonne. I valori relativi a j dipendono da quelli relativi a $j - 1$, i quali dipendono da quelli relativi a $j - 2$, e così via (il calcolo di $C(a, j - 1)$ avviene prima del calcolo di $C(b, j)$). Similmente, il calcolo di $C(i - 1, a)$ deve avvenire prima del calcolo di $C(i, b)$. Possiamo quindi sviluppare una funzione formata da due cicli for annidati, il primo (quello esterno) per j da 1 a n, il secondo (quello interno) per i da 1 a m. Prima di tali cicli, ricordiamo di inizializzare la tabella con le condizioni iniziali.

```
int C(int m,int n)
{int i,j;
for(i=0;i<=m;i++) T[i][0]=i;
for(j=0;i<=n;j++) T[0][j]=j*j;
for(j=1;i<=n;j++)
 for(i=1;i<=m;j++) T[i][j]=2T[i][j-1]-T[i-1][j];
 return T[m][n];
}
```

La funzione ha complessità $\Theta(m \cdot n)$ (sia in tempo, sia in spazio).