

Algoritmica

Appello 3/2/2009

Soluzione

Esercizio 1

[punti 10]

Modificare la procedura SelectionSort in modo tale che a ogni passo, invece di selezionare il minimo di n elementi, vengano selezionati il minimo e il massimo, che questi elementi assumano la loro posizione definitiva e la procedura venga poi iterata sui restanti $n-2$ elementi dell'array. Discutere la complessità della soluzione fornita.

```
SelectionSortMM (A, n):
  FOR (i=0; i <= n/2; i++) {
    Min = A[0]; imin = 0;
    Max = A[0]; imax = 0;
    ultimo = n-i;
    FOR (j=i+1; j < ultimo; j++) {
      IF (A[j] < Min) {
        Min = A[j];
        imin = j; }
      ELSE IF (A[j] > Max) {
        Max = A[j];
        imax = j;}
    }
    A[imin] = A[i]
    A[i] = Min;
    A[imax] = A[ultimo-1]
    A[ultimo-1] = Max;}
}
```

Il ciclo più interno viene eseguito su $n-1$ elementi, poi su $n-3$, poi su $n-5$ ecc. e il tempo all'interno del ciclo è costante. La complessità totale si ottiene notando che la somma dei numeri $1, 3, 5, \dots, n-3, n-1$ è sempre $O(n^2)$.

Esercizio 2

[punti 10]

Sono dati due Heap di minimo, ciascuno di n elementi memorizzati rispettivamente in due array a, b . Dare il codice di una procedura che selezioni il minimo corrente delle due strutture e lo memorizzi in un nuovo array c , per ottenere l'array ordinato

Confrontare la complessità ottenuta con quella che si otterrebbe applicando HeapSort ai due Heap, e facendo poi il Merge dei due array ordinati.

```
HeapMerge (a, b, n):
  \\ definisci c di lunghezza 2n\\
  i=0;
  WHILE (!isEmpty(a)&&!isEmpty(b)){
    IF (a[0] < b[0])
      c[i]= Dequeue (a);
    ELSE c[i]= Dequeue (b);
    i++;}
  WHILE (!isEmpty(a){
    c[i]= Dequeue (a);
```

```

    i++;}
    WHILE (!isEmpty(b){
        c[i]= Dequeue (b);
        i++;}

```

La procedura Dequeue costa $O(\log n)$ e viene invocata $2n$ volte, quindi in totale $2n \cdot O(\log n) = O(n \log n)$.

In alternativa si poteva applicare 2 volte HeapSort ai due heap e poi fare il merge, per un totale di $2 \cdot O(n \log n) + O(2n)$ che dà lo stesso risultato in ordine di grandezza ma ha una complessità maggiore.

Esercizio 3

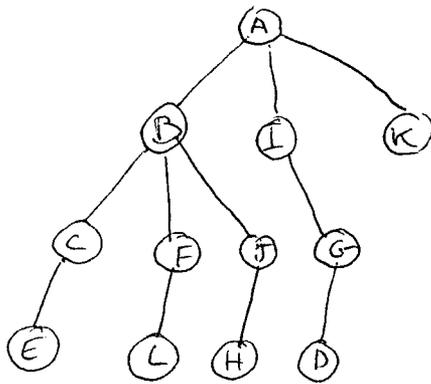
[punti 10]

Per il grafo seguente, memorizzato con liste di adiacenza ordinate alfabeticamente

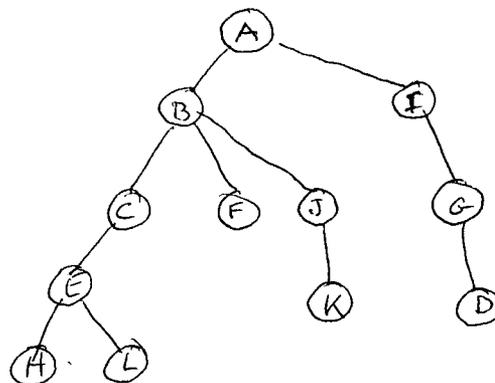
1. Indicare l'ordine con cui i vertici sono scoperti dalle visite BFS e DFS del grafo partendo dal vertice A.
2. Disegnare gli alberi BFS e DFS ottenuti con le visite
3. Classificare ogni arco con la classificazione indotta dalla visita DFS

1. BFS: A, B, I, K, C, F, J, G, E, L, H, D.
 DFS: A, B, C, E, H, L, F, J, k, I, G, D.

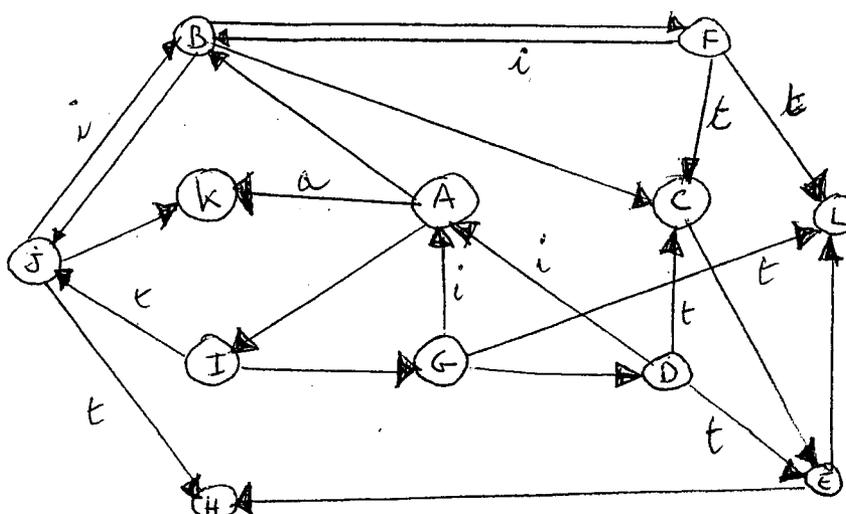
2. Albero BFS



Albero DFS



3. Classificazione



a: avanti
 i: indietro
 t: trasversale