Esercizio 1. Per un certo grafo non diretto connesso e pesato G sono stati appena calcolati il minimo albero di copertura T_1, e l'albero dei cammini minimi T_2 dal nodo 1 agli altri nodi di G. Si consideri ora il grafo G' ottenuto da G incrementando il costo di ciascun arco di G di una stessa quantità. Dimostrare o confutare ciascuna delle seguenti affermazioni:

a) T_1 è un albero di copertura minimo anche per il grafo G'.
b) T_2 è un albero dei cammini minimi con sorgente il nodo 1 anche per il grafo G'.

2) Vero, si basti pensare all'algoritmo di Kruskal, gli archi sono ordinati secondo il loro peso, aumentando di una stessa quantità ogni arco, l'ordinamento rimane il medesimo $\omega(e) > \omega(k) \Rightarrow \omega(e)+\delta > \omega(k)+\delta$ quindi Kruskal restituirà lo stesso MST.

b) Falso, si noti come nel seguente esempio, il cammino minimo da 1 ad x cambia.

Esercizio 2. Sia A[1,...,n] un array di n interi con n dispari. Sia A ordinato e ogni valore occorre o uno o due volte. Dare il pseudocodice di un algoritmo O(log n) che prenda in input l'array A e trova un valore presenta una sola volta. Per esempio, dato il input: A = [1, 2, 2, 5, 7, 9, 9, 10, 10], l'algoritmo puo' restituire "7" o alternativamente "1".

```
Es2(A:array){
    n=A.length()
    i=⌊n/2⌋
    if(n>1 ∧ A[1]≠A[0]){return A[0]}
    if(A[n-1]≠A[n-2]){return A[n-1]}
    if(A[i]≠A[i-1]){
        if(A[i]≠A[i+1]){return A[i]}
        if(i%2==0){return Es2(A[i:n-1])}
        return Es2(A[0:i-1])
    }
    if(i%2==0){return Es2(A[0:i])}
    return Es2(A[i+1,n-1])
}
```

Esercizio 3. Un foglie di un albero è un vertice con un solo vicino. Dare lo pseudocodice per un algoritmo che prende in input un albero di n vertici in formato del vettore dei padri e restituisce un elenco di tutti le foglie con complessità O(n).

```
Foglie(P:array){
    n=P.length()
    F[n]:array
    For(i=0,1...,n-1){
        F[i]=1
    }
    For(i=0,1...,n-1){
        F[P[i]]=0
    }
    L:list
    For(i=0,1...,n-1){
        if(F[i]==1){L.add(i)}
    }
    return L
}
```

Esercizio 4. Una pedina è posizionata sulla casella (1, 1) in alto a sinistra di una scacchiera n×n e mediante una sequenza di mosse tra caselle adiacenti deve raggiungere la casella (n, n) in basso a destra. Una pedina posizionata sulla generica casella (i, j) ha al più due mosse possibili: spostarsi verso il basso nella casella (i + 1, j+1), posto che i < n, j < n, spostarsi verso destra nella casella (i, j + 1), posto che j < n, o spostarsi verso destra nella casella (i + 1, j), posto che i < n. La sequenza di caselle toccate determina un cammino. Ogni casella della scacchiera ha un colore c(i, j) che è o rosso o verde. Descrivere un algoritmo che in tempo $O(n^2)$ calcola il numero di cammini che passano solamente per caselle rosse e vanno da (1, 1) a (n, n).

```
Caselle( n: intero , c: (i,j) → {r,v}) {
    T[n×n]: matrice
    T[0,0]=1
    For(i=0,1...,n-1){
        T[i,0]=0
        T[0,i]=0
        iF(C(i,0)==r){
            T[i,0]=T[i-1,0]
        }
        iF(C(0,i)==r){
            T[0,i]=T[0,i-1]
        }
    }
    For(i=1...,n-1){
        For(j=1...,n-1){
            T[i,j]=0
            iF(C(i,j)==r){
                T[i,j]+=T[i,j-1]
                T[i,j]+=T[i-1,j]
                T[i,j]+=T[i-1,j-1]
            }
        }
    }
    return T[n-1,n-1]
}
```