**Esercizio 1.** Dare lo sudo codice di un algoritmo O(n) che accetta in input il vettore dei padri P[ ] di un albero e vertice x, y, z e restituisce il minimo antenato comune dei vertici x, y, z.

Prop: Se $LCA(u,v)=x$, allora $LCA(u,v,\omega)=LCA(x,\omega)$

```
LCA(u:nodo, v:nodo, P:array){
    AU[n]={0,0,...0}
    do{                                   //O(n)
        AU[u]=1
        u=P[u]
    while(u≠P[u])
    while(true){                          //O(n)
        if(AU[v]==1){ return v }
        v=P[v]
    }
}

LCA_glob(ω:nodo, u:nodo, v:nodo, P:array){
    x=LCA(u,v,P)
    return LCA(x,ω,P)
}
```

**Esercizio 2.** Sia A[1,...,n] un array di n interi con n dispari. Sia A ordinato e ogni valore occorre esattamente due volte tranne uno. Dare il pseudocodice per un algoritmo O(log n) che prenda in input l'array A e trova il valore presenta una sola volta. Per esempio, dato l'input: A = [2, 2, 5, 7, 7, 9, 10, 10], l'algoritmo restituisce "7". Dato l'input: A = [1, 1, 4, 4, 6, 6, 8, 8, 9], l'algoritmo restituisce "9".

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

**Esercizio 3.** Dare lo sudo codice di un algoritmo $O(n^2m)$ che prende in input un grafo diretto aciclico con ogni vertice colorato o rosso o blu e due vertici x e y e restituisce "YES" in output se esiste un cammino da x a y con lo stesso numero di vertici blu che rosso. Se un tal cammino non esiste, l'algoritmo restituisce "NO" in output.

```
DFS_col(G:grafo, u:nodo, y:nodo, r:int, b:int){
    if(Sol=='YES'){return}
    if(colore(u)=='ROSSO'){ r++ }
    else{ b++ }
    if(u==y){
        if(r==b){ Sol='YES' } // Sol e' globale
        return
    }
    For each w ∈ u.adj_out{
        DFS_col(G,w,y,r,b)
    }
}
```

```
Es3(G:grafo, x:nodo, y:nodo){
    Sol='NO'
    DFS_col(G,x,y,0,0)
    return Sol
}
```

**Esercizio 4.** Dare lo sudo-codice di un algoritmo che prende in input un grafo non-diretto e connesso con ogni vertici colorato o rosso o blu e restituisce in output un albero di copertura con un numero minimo di archi con termine dello stesso colore (quindi il numero minimo di archi colorati rosso-rosso oppure blu-blu).

```
ES4 (G: grafo, col: E(G) → {r, b}) {
    A1 = {(x,y) ∈ E(G) | col(x) ≠ col(y)}
    A2 = {(x,y) ∈ E(G) | col(x) = col(y)}
    T: albero
    For each e ∈ A1 {
        iF(T ∪ {e} non ha cicli) {
            T = T ∪ {e}
        }
    }
    For each e ∈ A2 {
        iF(T ∪ {e} non ha cicli) {
            T = T ∪ {e}
        }
    }
    return T
}
```