Esercizio 1. **(max 5)** G un grafo non-diretto e conesso. Dato due vertici v_1 e v_2, la distanza da v_1 a v_2, scritto dist(v_1, v_2), e il lunghezza minima di un camino da v_1 a v_2. Invece, dato due sottoinsieme di vertice X e Y, la distanza da X a Y e

$$\min_{\{x \in X,\, y \in Y\}} \text{dist}(x, y)$$

Noto nel caso in cui $X \cap Y \neq \emptyset$, diciamo che dist(X,Y) = 0.

```
BFS_set(G:grafo, X⊆V(G), Y⊆V(G)){
    AY[n]:{0,0,...,0}
    For each t ∈ Y { AY[t]:1 }
    Dist[n]={-1,-1...,-1}
    Q:queve
    For each x ∈ X{
        Dist[x]=0
        Q.push(x)
    }
    do{
        v=Q.pop()
        For each w ∈ v.adj_out {
            if(Dist[w]==-1){
                Dist[w]=Dist[v]+1
                Q.push(w)
            }
        }
    }while(Q≠∅)
    md=∞
    For(i=0,1...,n){
        if(Dist[i]<md ∧ AY[i]==1){    md=Dist[i] }
    }
    return md
}
```

Esercizio 2. (max 5) Dato un albero con radice r e vertice u, v, w, il minimo antenato in comune LCA(u,v,w) è il antenato in comune di u, v e w più recente, cioé il antenato in comune di u, v e w più lontano dal radice. Dare lo pseudocodice per un algoritmo che prende in input un albero come vettori dei padri **P** e vertice u,v,w e trova con complessità O(n) il LCA(u,v,w).

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

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

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

Esercizio 3. (max 5) Dato un array A[] ordinato di numeri reali (quindi A[1] ≤ A[2] ≤ ... ≤ A[n]), dare lo pseudocodice per un algoritmo per trovare un insieme di cardinalità minima di intervalli di lunghezza uno (quindi intervalli di tipo [x, x+1]) che contengono tutti gli elementi A[i]. Per esempio, dato A=[1.1, 2, 2.05, 3, 4], una soluzione sarebbe [1.1, 2.1], ~~[2,3]~~ [3,4]. La complessità dell'algoritmo dovrebbe essere O(n).

```
Min_unit_interval(A[0:n]: array di reali){
    SOL={ [A[0],A[0]+1] }
    for each i=1,2...,n-1 {
        if(A[i]> SOL.last()[1]){
            SOL.add( [A[i],A[i]+1)
        }
    }
    return SOL
}
```

Esercizio 4 (max 7) Dato un array di interi positivi A di lunghezza n con

- A ordinato,
- $A[i] \neq A[j]$ per indici i, j, $i \neq j$.

trovare il minimo intero x, $x > 0$ tale che non esiste un indice i con $A[i] = x$. Per esempio, dato A = [1, 2, 3, 5, 7, 8, 9] in input, l'algoritmo dovrebbe restituire "4" come output. L'algoritmo dovrebbe avere complessità $O(\log n)$.

```
Min_inexistent(A: array){
    n= A.length()
    if(A[n]==n){return n+1}
    if(A[1]≠1){return 1}
    i=⌈n/2⌉
    while(true){
        if(A[i]==i){
            if(A[i+1]≠i+1){ return i+1}
            else{i=i+⌈i/2⌉ }
        else{
            if(A[i-1]==i-1){ return i }
            else{ i=⌈i/2⌉}
        }
    }
}
```

Dare lo pseudocodice di un algoritmo che prenda in input un grafo diretto G, due vertici u,v, e un intero k e restituisca il numero di passeggiate distinte di lunghezza al massimo k da u a v.
L'algoritmo dovrebbe avere complessità $O(n \cdot m \cdot k)$

definisco $T[\alpha, x]$ = num di passeggiate lunghe α da u a x.

$$T[\alpha, x] = \sum_{\substack{(y,x) \\ \in E(G)}} T[\alpha-1, y]$$

e' chiaro che $T[1, x] = \begin{cases} 1 & \text{se } \exists\ (u,x) \\ 0 & \text{altrimenti} \end{cases}$

```
Passeggiate(k:int, G:grafo, u:nodo, v:nodo){
    n=|V(G)|
    M[k+1][n]:matrice
    For(i=0...,n-1){
        if(∃ (u, i)∈E(G)){ M[1][i]=1}
    }
    For(i=2...k){
        For(j=0...,n-1){
            sum=0
            For each x | ∃ (x,j)∈E(G){
                sum+=M[i-1, x]
            }
            M[i,j]= sum
        }
    }
    return Σ_{i=1}^{k} M[i][v]
}
```