✓ **Esercizio 1.** G un grafo non-diretto e connesso. Dati due vertici v_1 e v_2, la distanza da v_1 a v_2, scritto $dist(v_1, v_2)$, è la lunghezza minima di un cammino da v_1 a v_2. Invece, dati due sottoinsiemi di vertici X e Y, la distanza da X a Y è

$$\min dist(x, y) \text{ con } \{x \in X, y \in Y\}.$$

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

Dare lo pseudocodice di un algoritmo $O(|V| + |E|)$ per trovare dist(X,Y) dati G, X,Y in input.

```
BFS_set(G:grafo, X:set, Y:set){
    n=|V(G)|
    Dist[n]={-1,-1,...,-1}
    Q:queue
    For each x ∈ X{
        Q.enque(x)
        Dist[x]=0
    }
    while(Q≠∅){
        v=Q.dequeue()
        For each w∈v.adj_out(){
            if(Dist[w]==-1){
                Q.enqueue(w)
                Dist[w]=Dist[v]+1
            }
        }
    }
    m=∞
    For each y∈Y{
        m=min(m,Dist[y])
    }
    return m
}
```

Esercizio 2. Dare lo pseudocodice per risolve il seguente problema: dati n oggetti $x_1, x_2, \dots, x_n$ ognuno di un costo c_i e un valore v_i, $1 \le i \le n$, e un vincolo A, trovare il sottoinsieme di $x_1, x_2, \dots, x_n$ di costo minimo con valore totale al minimo A.

$T[K, \alpha]$ = costo minimo dei primi K oggetti con somma almeno α

$$T[1, \alpha] = \begin{cases} \infty \text{ se } v_1 < \alpha \\ c_1 \text{ altrimenti} \end{cases}$$

```
ES2({x1...xn}, {c1...cn}, {v1...vn}, A:int){
    T[n+1 x A]: matrice
    For(i=1...,α){
        T[1,i] = ∞
        iF(i ≤ v1){T[1,i] = c1}
    }
    For(i=2...n){
        For(J=1...,A){
            m = T[i-1,J]
            iF(vi<J){m=min(m, T[i-1, J-vi]+ci)}
            iF(J ≤ vi){m=min(m, ci)}
        }
    }
    Sol: List
    J=A
    For(i=n,n-1...,0){
        iF(T[i,J] ≠ T[i-1,J]){
            Sol.add(xi)
            J-=vi
        }
    }
    return Sol
}
```

valore	1	4	4	2	3
costo	1	3	2	1	2
1	1	1	1	1	1
2	∞	3	2	1	1
3	∞	3	2	2	2
4	∞	3	2	2	2
5	∞	4	3	3	3

} oggetti

$T[K-1, \alpha]$

$T[K-1, \alpha - v_K] + c_K$

c_K se $v_K \ge \alpha$

Esercizio 3. Dato un grafo diretto G con pesi p sugli archi, il peso di un ciclo è la somma dei pesi degli archi. Dare lo pseudocodice di un algoritmo che prenda in input un grafo diretto con pesi e trovi il ciclo di peso minimo. Se il grafo non ha cicli, l'algoritmo dovrebbe restituire "INFINITY". L'algoritmo dovrebbe avere complessità O(m(n+m)logn).

```
DFS(G:grafo, u:nodo, t:array, T:array, Vis:array, c:int){
    c++
    Vis[u]=1
    t[u]=c
    For each w~u {
        if(Vis[w]==0){ DFS(G, w, t, T, Vis, c) }
    }
    T[u]=c
}

MinCycle(G:grafo){
    t[n]={0,...0}
    T[n]={0...0}
    Vis[n]={0...0}
    c=0
    For each u∈V(G){
        if(Vis[u]==0){ DFS(G, u, t, T, Vis, c) }
    }
    m=∞
    For each ((x,y)∈E(G)){                          //O(m)
        if( [t[x], T[x]] ⊆ [t[y], T[y]] ){
            Dist=Dijkstra(G, y)
            m=min(m, Dist[x]+w(x,y))                //O((m+n)·log(n))
        }
    }
    return m
}
```

Esercizio 4 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 $x \notin A$. 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).

```
Es4(A:array){
    n=A.length()
    if(A[n-1]==n){return n+1}
    if(A[0]≠1){return 1}
    i=⌊n/2⌋
    in=⌊n/4⌋
    while(true){
        if(A[i-1]==i ∧ A[i]≠i+1){ return A[i] }
        if(A[i]==i+1){ i=i+in }
        else{ i=in }
        in=in/2
    }
}
```