Esercizio 1 (22.2-8, [1]). Sia $G = (V, E)$ un grafo non diretto, allora si definisce il *diametro* di G, $diam(G) = \max_{u,v \in V} d(u, v)$, il massimo della distanza tra due qualsiasi vertici di G. Fornire un algoritmo in pseudo-codice che restituisca il diametro di un grafo G, nel caso in cui G sia un albero. E' possibile ottenere una soluzione con tempo di esecuzione $\mathcal{O}(|V|)$?

```
MaxDist (G:grafo, u:nodo){
    Dist[n]={-1,-1..-1}
    Dist[u]=0
    Q:queue
    Q.push(u)
    do{
        x=Q.pop()
        For each(y~x){
            if(Dist[y]==-1){
                Dist[y]=Dist[x]+1
                Q.push(y)
            }
        }
    }while(Q≠0)
    m=0
    w:nodo
    for(int i=0; i<n; i++){
        if(Dist[i]>m){
            m=Dist[i]
            w=i
        }
    }
    return (w, m)
}
```

```
Diametro(G:grafo){
    u=nodo di G a caso
    (x, a)=MaxDist(G, u)
    (y, b)=MaxDist(G, x)
    return b
}
```

Esercizio 2 (I. Salvo). Fornire un algoritmo in pseudo-codice che, dato un grafo non diretto $G = (V, E)$ e due nodi u e v, restituisce tutti i nodi che hanno la stessa distanza da u e v in tempo $\mathcal{O}(|V| + |E|)$.

```
Dist_comuni(G:grafo, u:nodo, v:nodo){
    DistU=BFS(G,u)
    DistV=BFS(G,v)
    O:set
    for(i=0,1...,n-1){
        if(DistU[i]∧DistV[i]){ O.add(i) }
    }
    return O
}
```

Esercizio 3 (I. Salvo)**.** Rimuovere un arco da un grafo (diretto o non diretto) modifica le distanze tra alcune coppie di vertici e ne lascia invariate alcune. Fornire un algortimo in pseudo-codice che dato un grafo $G = (V, E)$ diretto, un vertice $s \in V$, un arco $(u, v) \in E$ e un vettore dei padri *Parent*, relativo ad una BFS effettuata su G a partire da s determina se la rimozione di (u, v) da G modifica le distanze da s. E' possibile ottenere una soluzione con tempo di esecuzione $\mathcal{O}(|V|)$?

Se l'arco rimosso NON fa parte dell'arborescenza, non ci sono modifiche. altrimenti puo' variare.

```
ModificaDist( G:grafo, s:nodo, (u,v): arco, P:array){
    iF( P[v]≠u){ return False}
    Dist=BFS(G,s)
    For(i=0...,n-1){
        iF( (i,v)∈E(G)){
            iF( Dist[i]==Dist[u]) {return False }
        }
    }
    return True
```

NON so se funziona :/

Esercizio 4. Nel gioco degli scacchi il *cavallo* si muove nella seguente maniera: due caselle in orizzontale e una in verticale, o viceversa. Fornire un algoritmo in pseudo-codice che, data una scacchiera MxN, la posizione di partenza e quella di arrivo, calcola il numero minimo di turni necessari ad un cavallo di passare da una posizione all'altra in maniera che il tempo di esecuzione sia pari a $\mathcal{O}(M + N)$.

Che modifiche sono necessarie se invece del cavallo si usa uno degli altri pezzi? Oppure se sono presenti degli altri pezzi statici ma non mangiabili?