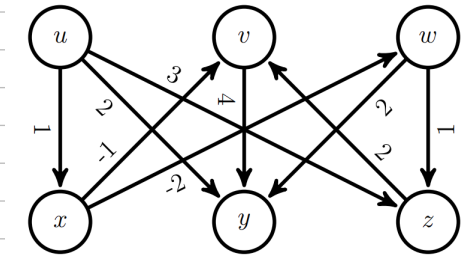
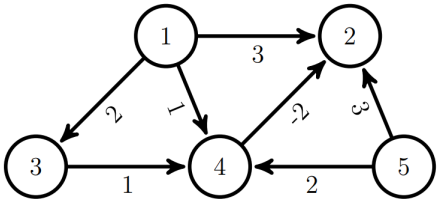
Esercizio 1. Il problema ALL-PAIRS SHORTEST PATH consiste nel ottenere il peso di un cammino minimo per ogni coppia di vertici di un grafo, cioè dato un grafo diretto $G = (V, E)$, la richiesta è quella di ottenere per ogni due vertici $u, v \in V$, il cammino di peso minimo da u a v. Risolvere questo problema fornendo la matrice dei cammini e dei padri per ognuno dei seguenti grafi:





- GRAFO 1

	u	v	w	x	y	z
u	0	0	-1	1	2	0
v	∞	∞	∞	∞	4	∞
w	∞	3	0	∞	2	1
x	∞	-1	-2	0	0	-1
y	∞	∞	∞	∞	0	∞
z	∞	2	∞	∞	6	0

- GRAFO 2

	1	2	3	4	5
1	0	-1	2	1	∞
2	∞	0	∞	∞	∞
3	∞	-1	0	1	∞
4	∞	-2	∞	0	∞
5	∞	0	∞	2	0

Esercizio 2 (25.2-6, [1]). Modificare lo pseudo-codice dell'algoritmo di Floyd-Warshall per individuare, se esiste, un ciclo di peso negativo nel grafo in esame. Inoltre, fornire lo pseudo-codice di un algoritmo che, se esiste, ritorna la lunghezza minima di un ciclo di peso negativo con un costo computazionale di $\mathcal{O}(|V|^4)$.

Il seguente algoritmo trova il ciclo di peso negativo col minor numero di archi.

```
LeastLongNegCycle( G:grafo con n nodi) {
    S:Stack
    inS[n]={0,0...,0} // indica quali nodi sono nello stack
    Vis[n]={0,0...,0}
    out = ∞
    For (u ∈ V(G) | Vis[u]≠1) {
        S.push(u)
        inS[u]:1
        Vis[u]=1
        while(S≠∅) {
            v:S.top()
            if(∃ w∈v.adj | Vis[w]==0){
                Vis[w]=1
                S.push(w)
                inS[w]=1
            }else{
                For(w∈v.adj){
                    if(inS[w]==1){ // ciclo trovato
                        out=min(out, NegCycle(G,w,S))
                    }
                }
                inS[S.pop()]=0

            }
        }
    }
    if(out==∞){return NULL}
    return out
}

NegCycle(G:grafo, u:nodo, S:stack){
    A=S.copy()
    start=u
    l=0
    p=0
    do{
        K=A.pop()
        p+=ω(K,u)
        l++
        u=K
    }while(start≠K)
    if(p≥0){return ∞}
    return l
}
```

Esercizio 3 (25.2-8, [1]). Dato un grafo diretto $G = (V, E)$, la chiusura transitiva di G è un grafo $G^* = (V, E^*)$ tale che $(u, v) \in E^*$ se e solo se esiste un cammino da u a v in G. Fornire un algoritmo per calcolare la chiusura transitiva di un grafo diretto $G = (V, E)$ in maniera che il tempo di esecuzione sia $\mathcal{O}(|V| \cdot |E|)$.

```
ChiusuraTransitiva(G:grafo){
    G*:grafo
    V(G*):V(G)
    E(G*):E(G)
    For(u∈V(G)){
        Vis[n]:{0,0...,0}
        DFS_t(G,u,Vis,u,G*)
    }
    return G*
}
```

```
DFS_t(G:grafo, u:nodo, Vis:array, s:nodo, G*:grafo){
    iF(s≠u){
        E(G*):E(G*)U{(s,u)}
    }
    Vis[u]:1
    For(w∈u.adj){
        iF(Vis[w]≠1){ DFS_t(G, w, Vis, s, G*) }
    }
}
```

Esercizio 4 (M. Lauria). Si considera una griglia $n \times n$ con $n > 0$. Un cammino su questa griglia deve partire dalla cella di coordinate $(0, 0)$ in alto a sinistra e deve arrivare alla posizione di coordinate $(n-1, n-1)$ in basso a destra. E' possibile muoversi solo su celle adiacenti, andando di un passo verso il basso o di un passo verso destra. Inoltre, sono vietati i cammini che toccano le celle di coordinate (i, j) con $i > j$, cioè non è permesso andare sotto la diagonale che va da $(0, 0)$ a $(n-1, n-1)$. Fornire in pseudo-codice un algoritmo che calcoli il numero di cammini validi con un tempo di esecuzione $\mathcal{O}(n^2)$.

```
CamminiSopraDiagonale(n:intero){
    M[nxn]:matrice
    For(i=0,...n-1){M[i,0]=1}
    For(i=1,...n-1){M[0,i]=0}
    For(i=1,...n-1){
        For(j=1...,n-1){
            iF(i>j){M[i,j]=0}
            else{M[i,j]=M[i-1,j]+M[i,j-1]
        }
    }
    return M[n-1,n-1]
}
```