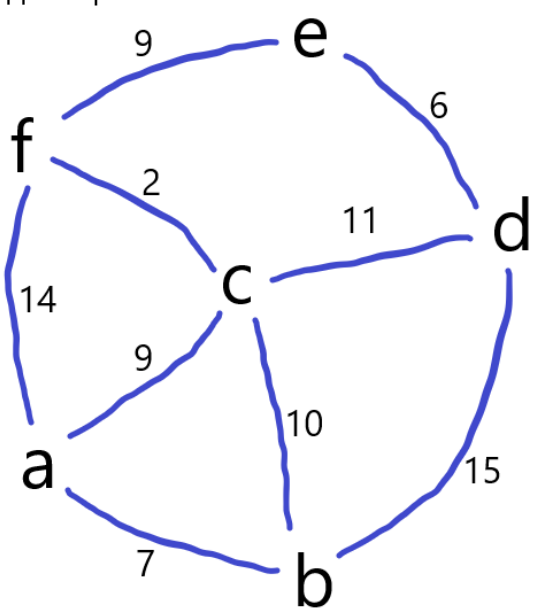
Зад. 1

Дадени са свързан, тегловен и ориентиран граф G=(V, E) и връх v∈V. Да се намерят теглата на най-кратките пътища от v до всеки друг връх u∈V.

а) ако няма отрицателни тегла (бонус за поддръжка на пътя)

б) ако има отрицателни тегла (бонус за поддръжка на пътя и проверка за отрицателен цикъл)

а) Дийкстра



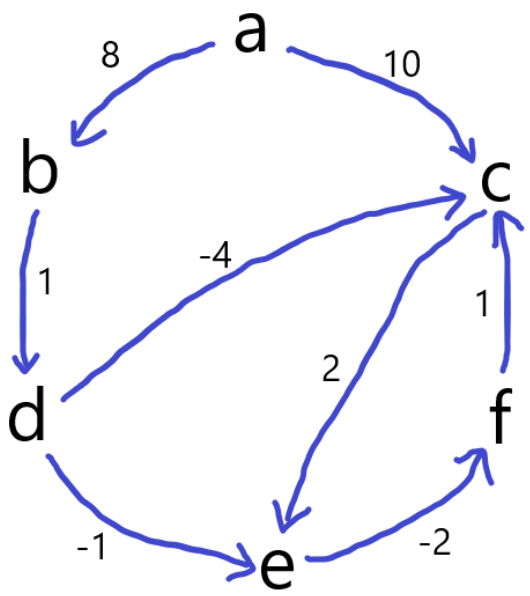

```
dijkstra(G=(V, E), stV∈V):
        n←|V|
        visited[1..n]←[FALSE, …, FALSE]
        dist[1..n]←[∞, …, ∞]
        parent[1..n]←[undef, …, undef]
        pq←PriorityQueue.init() //min pq of 3-tuple (vertex, weight, fromVertex), compared by the
2nd coordinate
        pq.push((stV, 0, stV))
        while pq.isEmpty()=FALSE do
                (u, w, fromV)←pq.pop()
                if visited[u]=TRUE then
                        continue
                visited[u]←TRUE
                dist[u]←w
                parent[u]←fromV
                for each (u, v, w')∈E do
                        if visited[v]=FALSE then
                                pq.push((v, dist[u]+w'))
        return (dist, parent)
```

Сложност:

O(|E|*log(|V|)) - binary heap

O(|E|+|V|*log(|V|)) - fibonacci heap

б) Белман-Форд

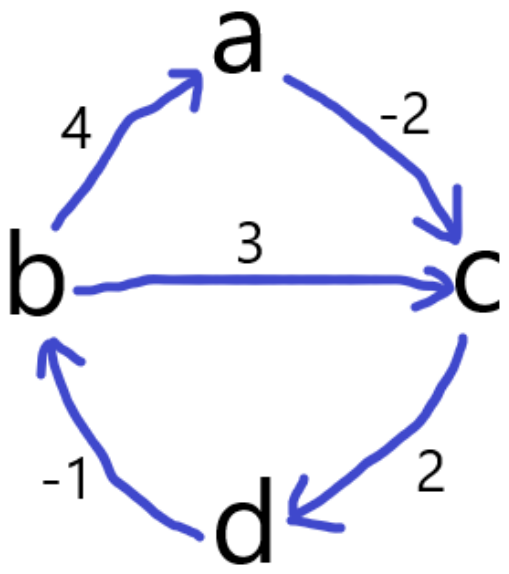



```
bellman-ford(G=(V, E), stV∈V):
        n←|V|
        dist[1..n]←[∞, …, ∞]
        pred[1..n]←[undef, …, undef]
        dist[stV]←0
        pred[stV]←stV
        for i←1 to n-1
                for each (u, v, w)∈E do
                        if dist[u]+w<dist[v] then
                                dist[v]←dist[u]+w
                                pred[v]←u
        return (dist, pred)
```

За проверката за отрицателен цикъл просто пускаме още 1 итерация на for-а.. ако имаме подобрение за някое ребро, то има отрицателене цикъл.

Сложност:
O(|E|*|V|)

Зад. 2
Даден е свързан, тегловен, ориентиран граф G=(V, E) без отрицателни цикли. Да се намери минимален път от всеки до всеки връх.

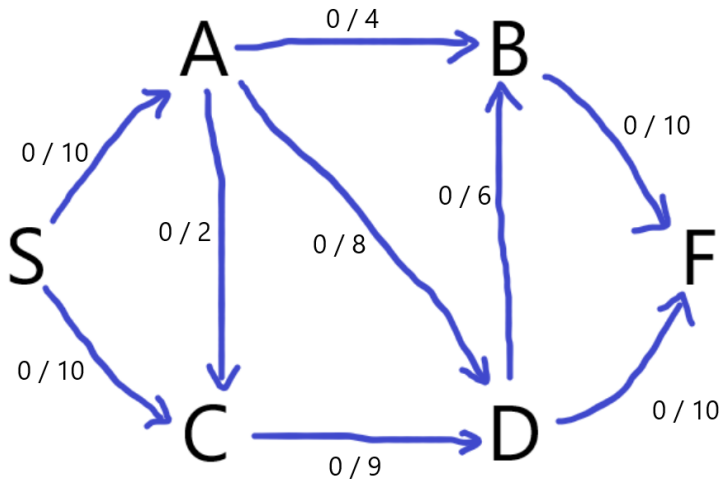



```
floyd-warshal(G=(V, E)):
        n←|V|
        dist[1..n][1..n]←[[∞, …, ∞], …, [∞, …, ∞]]
        for each (u, v, w)∈E do
                dist[u][v]←w
        //for i←1 to n
        //        dist[i][i]←0
        for cap←1 to n
                for u←1 to n
                        for v←1 to n
                                dist[u][v]←min(dist[u][v], dist[u][cap]+dist[cap][v])
        retrun dist
```

Сложност:
$O(|V|^3)$

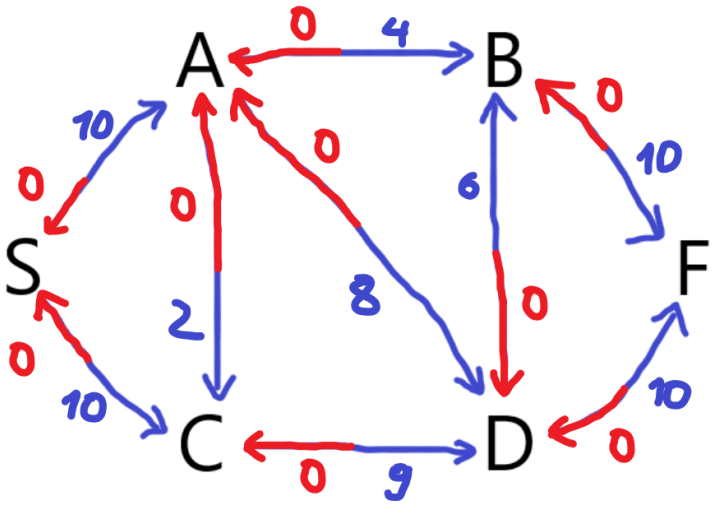
Зад. 3

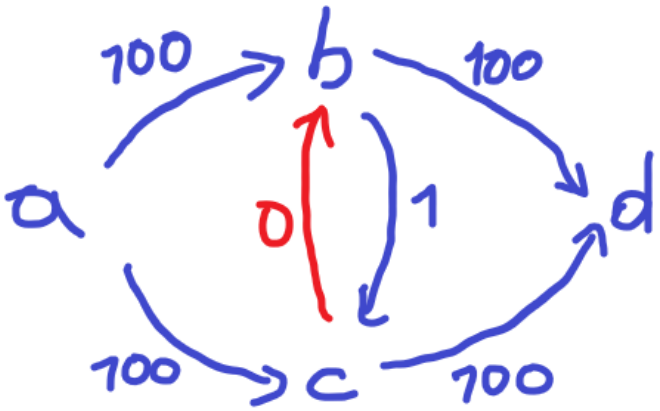
Даден е свързан, тегловен и ориентиран граф, съдържащ точно един източник и точно един сифон. Да се намери максималния поток от източника до сифона, като теглата на ребрата указват максималния капацитет на потока, който може да мине през тях.




```
dfs(G=(V, E), n, currV, fnV, parent[1..n], visited[1..n]):
        if visited[currC]=TRUE then
                return FALSE
        visited[currV]←TRUE
        for each (currV, u, w)∈E do //E is adj. matrix
                if w>0 and visited[u]=FALSE then
                        parent[u]←currV
                        if u=fnV or dfs(G, n, u, fnV, parent, visited) then
                                return TRUE
        return FALSE

ford-fulkerson(G=(V, E), stV, fnV):
        n←|V|
        E'←expandEdges(E)
//∀ <u, v, w>∈E[<u, v, w>∈E' & (∄ w' <v, u, w'>∈E ⇒ <v, u, 0>∈E')]
        paths←List.Init() // list of couples: (path, weight), where path is list of verticies and weight is
the path weight
        totWeight←0
        parent[1..n]←[undef, …, undef]
        visited[1..n]←[FALSE, …, FALSE]
        while dfs(G'=(V, E'), n, stV, fnV, parent, visited)=TRUE do
                path←constructPath(fnV, parent, stV)
                minW←getMinWeight(path, E')
                for each (u, v)∈path do
                        reduceW((u,v), minW)
                        increaseW((v, u), minW)
                paths.push_back((path, minW))
                totWeight←totWeight+minW
                parent[1..n]←[undef, …, undef]
                visited[1..n]←[FALSE, …, FALSE]
        return (totWeight, paths, E')
```





Метода, използвайки dfs си няма име и е със сложност O(f*|E|), където f е отговора на задачата

Метода, използвайки bfs се казва алг. на Edmons-Karp и е със сложност $O(|V| * |E|^2)$

Метода, използвайки bfs и dfs. *И*деята е bfs да сложи level-и на vertex-ите спрямо stV и после пускаме dfs-та докато можем, като dfs-то може да пусне рекурсивно друго dfs САМО с по-високо ниво. Когато не можем да открием повече пътичща с dfs, пускаме отново bfs за level-ване. В момента в който bfs не успее да стигне от stV до fnV, то алг. приключва работата. Сложността е $O(|V|^2 |E|)$ и дори $O\left(\sqrt{|V|} \left|E\right|\right)$ при двуделни графи.

Приемаме на доверие, че е така..