```
Dijkstra (G, w, start):
for all v in V do:                    } init
    dist[v] ← ∞
    π[v] ← nil
dist[start] ← 0
H ← makeQueue(V) //използва
// dist ст-тиете на върховете
// като ключове
while H ≠ ∅ do
    u ← deleteMin(H)
    for all edges <u,v> ∈ E do
        if dist[v] > dist[u] + w(u,v)
            dist[v] ← dist[u] + w(u,v)
            π[v] ← u
            decreaseKey(H, v)
```

start

Bellman-Ford(G, w, s): $O(n \cdot m)$

d[1...n]: distances, nums

π[1...n]: parents array

for i ← 1 to n do:

π[i] ← Nil , d[i] ← ∞

d[s] = 0

for i ← 1 to n-1 do:

for each edge ⟨u,v⟩ ∈ E

relax(u, v, w, d, π)

for each edge ⟨u,v⟩ ∈ E

if d[v] > d[u] + w(u,v)

return false // отриц. цикъл

return ⟨d, π⟩

$O(n \cdot m)$

relax(u, v, w, d, π)

if d[v] > d[u] + w(u,v)

d[v] ← d[u] + w(u,v)

π[v] ← u

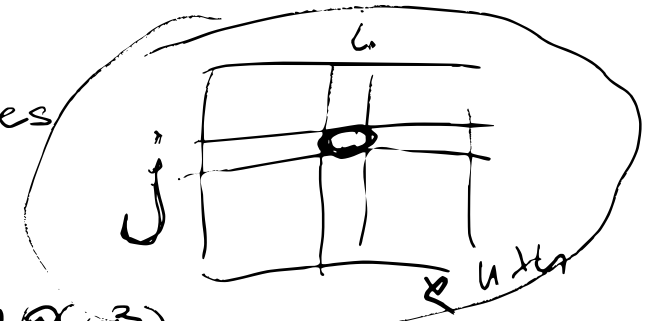
Floyd-Warshall(G, w):

d[1...n][1...n] : distances
π[1...n][1...n] : parents
init(G, w, d, π)

използв. черепа (линейно) и/у номерацията на върховете

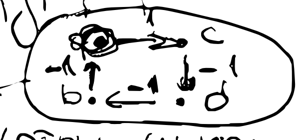
```
for k ← 1 to n do:
    for i ← 1 to n do
        for j ← 1 to n do
            relax(i, j, k, π, d)
```

} $\Theta(n^3)$



dist на k до k трябва да е 0

```
for k ← 1 to n
    if d[k, k] < 0
        return false   //отриц. цикъл
return <d, π>
```



```
relax(i, j, k, π, d)
    if d[i][j] > d[i][k] + d[k][j]
        d[i][j] ← d[i][k] + d[k][j]
        π[i][j] ← k   междинен връх
```

```
init(G, w, d, π)
    for v ∈ V do
        for u ∈ V do
            d[v, u] ← { 0, v = u ; w(v, u), <u, v> ∈ E ; ∞, else }
            π[u][v] ← nil
```

(3) Искаме най-леката MST за конструиране, т.е. (U) ≤ Q бр-во от върхове да стоят листа в тази MST.

(I) MST V\U

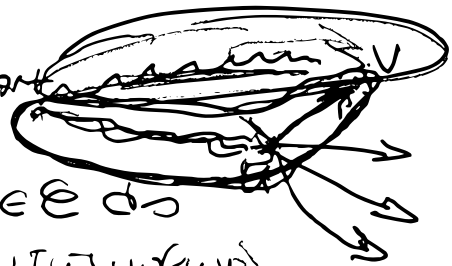
(II) Добавяме най-леките ребра м/у V\U и U.

⑤ Как да изброим всички най-къси пътища в граф м/у два върха като теглата са положителни на ребрата?

Дадени са $u, v \in V$ и знаем, че

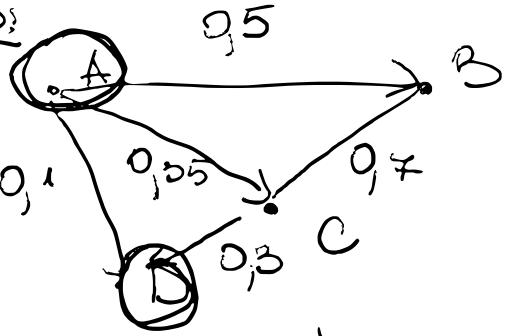
$\omega : V \times V \to \mathbb{R}^+$

```
Dijkstra(G, w, start):                    нека start=u
for all v in V do:
    dist[v] ← ∞
    numPaths[v] ← 0                       } init
dist[start] ← 0, numPaths[start] ← 1
H ← makeQueue(V) //използва
// dist с-тиците на върховете
// като ключове
while H ≠ ∅ do
    u ← deleteMin(H)
    for all edges <u,v> ∈ E do
        if dist[v] > dist[u]+w(<u,v>)
            dist[v] ← dist[u]+w(<u,v>)
            numPaths[v] ← numPaths[u]
            decreaseKey(H,v)
        else if dist[v] = dist[u]+w(<u,v>)
            numPaths[v] ← numPaths[v] +
                          numPaths[u]
```



(10) Възли в компютърна мрежа предават съобщения. Всеки канал има определена надеждност $p \in (0,1)$
p - вероятност съобщението да бъде предадено по канала без грешка.
Търсим дали най-надеждният път от един възел s до друг възел t.

пример:



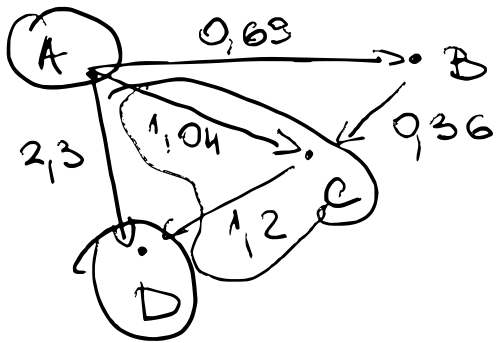
$\ln 0{,}1 \approx -2{,}3 \xrightarrow{\cdot(-1)} 2{,}3$

$\ln 0{,}3 \approx -1{,}2 \to 1{,}2$

$\ln 0{,}35 \approx -1{,}04 \to 1{,}04$

$\ln 0{,}5 \approx -0{,}63 \to 0{,}63$

$\ln 0{,}7 \approx -0{,}36 \to 0{,}36$



$A \xrightarrow{1,04} C \xrightarrow{1,2} D$

$A \xrightarrow{-1,04} C \xrightarrow{-1,2} D$

$A \xrightarrow{0,35} C \xrightarrow{0,3} D$

$p = 0{,}35 \cdot 0{,}3 = 0{,}105 = e^{\ln(0,3)+\ln(0,35)}$

наибольшая надежность.

(11) Имаме n града свързани с пътища. Дадени са дължините на пътищата. В някои градове има полицейски участъци. Престъпник иска да се скрие в град, който е най-отдалечен от полицейски участък т.е. разстоянието м/у него и най-близкия полицейски участък да е възможно най-голямо.

- Неор. граф.
- върхове: градовете
- ребрата: пътищата м/у градовете
- тегло: разстоянията на пътищата
  положителни

Още няколко Dijkstra

Вар 1: Нека всички ребра с поне. участващи са $\{v_1, \dots, v_t\}$ $t \leq n$

В init на Dijkstra при dist[s] ← 0

```
for u in U do:
    dist[u] ← 0
```

.....

В края след изп на Dijkstra избираме най-дългия път

Вар 2: Добавяме нов фиктивен връх start и от всеки връх, в който има поне. участващ, добавяме ребро от start към него с тегло 0.

ДИ (12) Нека неориентиран свързан граф
$G = \langle V, E \rangle$, $w: E \to [a, b]$, $0 < a < b$.
Нека T е произволно MST на G.
Как да намерим това T с
линеен алгоритъм, когато е
нужно да намерим MST.

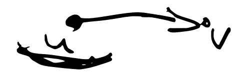
(13) Създайте ефикасен алгоритъм, който
при даден граф G и ребро e от него
е намира дали в графа има цикъл,
съдържащ e.

(14) Даден е ориент. граф $G=\langle V, E\rangle$ с $|V|=n$ върхо и ребра представени чрез матрица на съседство. Известно алгоритъм, работещ с матрицата на съседство, намира за време от $\Omega(n^2)$ дно или и някои изключения. Универсален канал наричаме връх, в който влизат ребра и всички останали върхове, но не излиза нито едно ребро т.е. с входяща степен $n-1$ и изходяща степен 0. Създайте алгоритъм, който по матрица на съседство граф намира универсален канал за време $O(n)$.

Observation: редът на унив. канал е пълен с 0-ли, а колоната му е пълна с 1-ци без примка към себе си

① • $S = V$

② • Взимаме $\{u, v\} \subseteq S$, т.ч. $u \neq v$.

- Ако $A[u][v] = 1$ 

  Тогава $S = S \setminus \{u\}$

- Ако $A[u][v] = 0$ 

  Тогава $S = S \setminus \{v\}$

$\Theta(n)$

Тогава $n-1$ пъти пускаме ②.

$S = \{t\}$

for $i \leftarrow 1$ to n do:

  if $A[t][i] = 1$

    return false

$O(n)$

if $A[i][t] = 0$ & $i \neq t$

return false

return true

$T(n) \simeq n + n \simeq n$

(15) Най-кус път и най-дълг път в DAG: $G = \langle V, E \rangle$ от v_0 до всички останали.

Зад (4) от Dina.pdf

Намиране диаметър на дърво.