Част от задачите тук ще ги обсъдим другия път.

```
Dijkstra(G, w, start):
for all v in V do:
    dist[v] ← ∞
    π[v] ← nil
dist[start] ← 0
H ← makeQueue(V) //използва
//dist ст-тите 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)
```

```
Bellman-Ford(G, w, s): O(n·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 |V| - 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. цикл
    return <d, π>

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, ω)

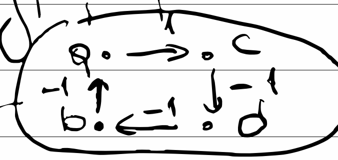
d[1, ..., n][1, ..., n] : distances

π[1, ..., n][1, ..., n] : parents

init(G, ω, d, π)

използв. наредба (линейна) и/ц 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)



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⟩

init(G, ω, d, π)
    for v ∈ V do
        for u ∈ V do

$$d[v, u] \leftarrow \begin{cases} 0, & v = u \\ \omega(v, u), & \langle u, v \rangle \in E \\ \infty, & \text{else} \end{cases}$$

d[i, k] + d[k, j]

i d[i][j] j

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

Секция задачки задачки

1. Имаме граф $G = \langle V, E \rangle$.
Най-лекото ребро съдържа ли се в някое MST?

Да, с която и да е стойност MST съдържащо го и като цяло, ако е уникално теглото на най-лекото ребро, всяко MST го съдържа, но ако има повече от едно ребро с това тегло, не винаги ще се съдържа.

2. Имаме покриващо дърво T на G, $w(\langle u, v \rangle) \in E$ за $u, v \in V$.
Дали, ако заменим w с ф-я w', т.ч. $w'(\langle u, v \rangle) \leq w(\langle u, v \rangle) + 1$
за $\langle u, v \rangle \in E$

MST T ще продължи да бъде MST?

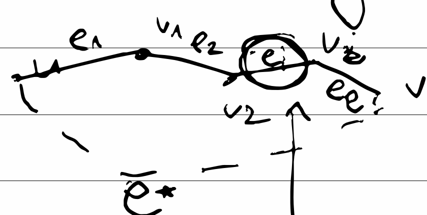
Да ще продължи, т.к. теглата на всички покриващи дървета се увеличава с $|V|-1 = n-1$.

③ Имаме граф и намаляваме $w(\langle u,v\rangle)-k$ за някое ребро $\langle u,v\rangle \in E$. Как ще се ъпдейтне MST-то T?

- Ако $\langle u,v\rangle \in T$, то само сме го направили още добро.
- Ако $\langle u,v\rangle \notin T$, то:

- взимаме най-тежкото ребро в пътя от u до v от T

Нека $e^* = \langle u,v\rangle$

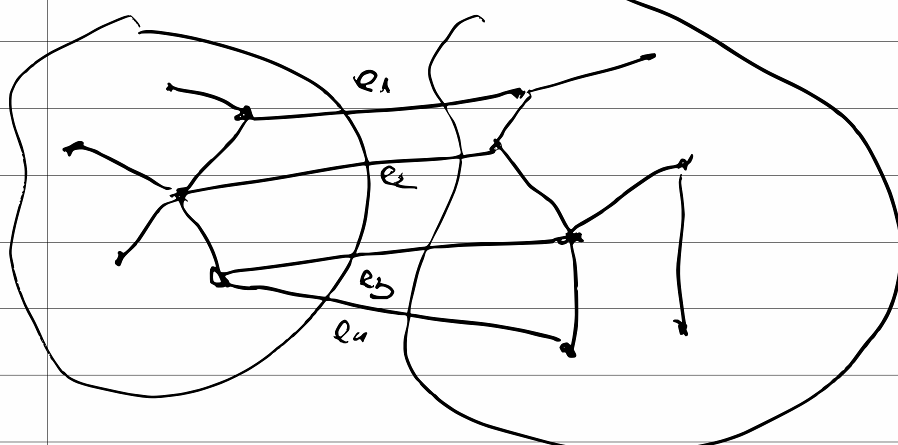


нека е най-тежкото и го белязаме с e_{max}

- Ако $e_{max} > w(e^*)-k$, то махаме e_{max} от T и добавяме e^*
- Иначе нищо.

$O(n)$.

④ Същото като ⑤, но проверим +ж.
Ако реброто не участва в T, то е окей.
Ако участва, то го махаме и получаваме две компоненти на силна свързност (не забравяйте, че сме в неориентиран граф) и сега м/у тези две компоненти избираме най-леко ребро



$\Theta(n^2)$

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

Можем да модифицираме Dijkstra

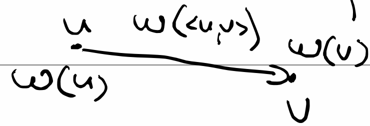
```
ModifiedDijkstra(G, w, start):
  for all v in V do:
    dist[v] ← ∞
    paths[v] ← 0
  dist[start] ← 0
  paths[start] ← 1
  H ← makeQueue(V)
  while H ≠ ∅ do
    u ← deleteMin(H)
    for all edges <u,v> ∈ E do
      if dist[v] > dist[u] + w(<u,v>)
        dist[v] ← dist[u] + w(<u,v>)
        paths[v] ← paths[u]
        decreaseKey(H, v)
      else if
        dist[v] = dist[u] + w(<u,v>)
          paths[v] ← paths[v] +
                     paths[u]
```

original
modified

⑥ С имя тежести на ребрата и на върховете. Как да намерим най-късите пътища?

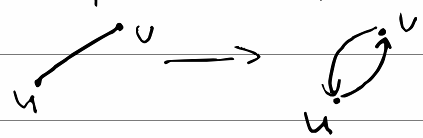
Тви начин:

$$w'(\langle u,v\rangle) \leq \underbrace{w(\langle u,v\rangle)}_{\text{ребро}} + \underbrace{w(v)}_{\text{връх}}$$



Вече с новото w' прилагаме алгоритми за най-къс път.

* Ако имаме неориентиран граф, го правим ориентиран и след



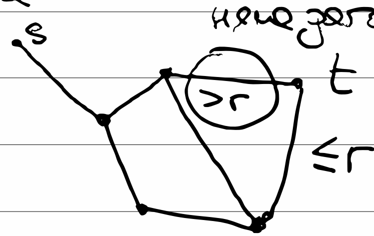
това ще приложим модификация.

⑦ Възможно ли е MST и SPT (shortest paths tree) да нямат общо ребро?

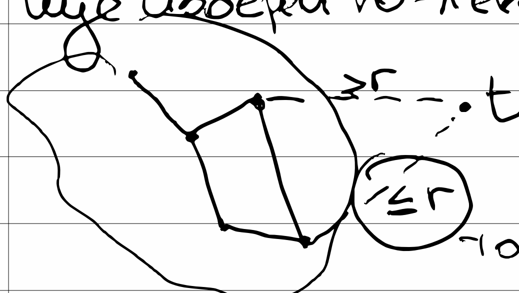
Не, най-лекото ребро излизащо от start ще участва и в двете.

(8) Нека между два върха ∃ път, в който най-тежкото ребро е ограничено от константа c. Дали има MST, т.ч. всички ребра да са ≤ от r.

Реално при алгоритми за MST всеки път, когато правим избор кое ребро да добавим ние имаме това



Всеки път между s и t независимо откъде минем ще има ребро тежко поне r. Ако знаем, че всички ребра са тежки най-много r, то в пътя с тежест на ребро ≤r ще избере по-лекото:

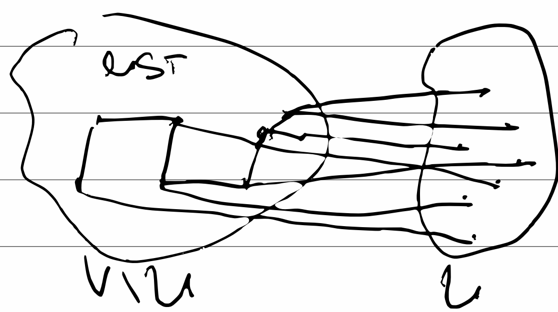


(9) Искаме най-лекото MST за конструиране, т.е. $U \subsetneq V$ мн-во от върхове да станат листа в това MST.

Идея:

(1) Строим MST от $V \setminus U$.

(2) Добавяме най-леките ребра от $V \setminus U$ към U и готово



(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. Търси се най-надежден път от един възел s до друг възел t.

Идея:

Върховете са комуникационните възли. Ребрата ще са каналите помежду им. Ориентиран е графа. Теглата на ребрата са надеждността на каналите. Тогава надеждността на пътя е произведението от надеждностите на ребрата.

Въпросът е, че като умножаваме много числа в горе $(0;1)$ получаваме изключително малки числа.

Можем следното да направим:

- Логаритмуваме теглата

  $\lg p$ за $p \in (0,1)$ е отрицателно
- Умножаваме по (-1) и получаваме положителни числа т.е.

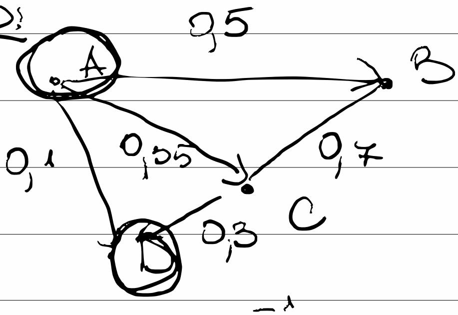
$-\lg p > 0$.

От свойства на логаритмите, то дължината на пътя ще бъде сбора от теглата на ребрата.

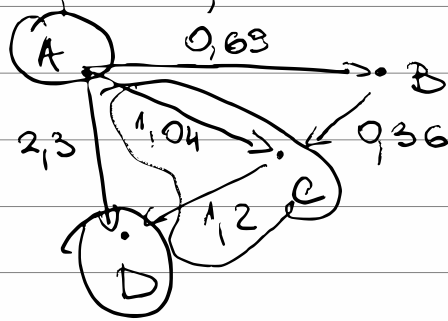
Заради минуса търсим най-късия.

Т.е. Dijkstra $\rightarrow O(m + n\lg n)$

пример:



$$\ln 0{,}1 \approx -2{,}3 \xrightarrow{-1} 2{,}3$$
$$\ln 0{,}3 \approx -1{,}2 \rightarrow 1{,}2$$
$$\ln 0{,}35 \approx -1{,}04 \rightarrow 1{,}04$$
$$\ln 0{,}5 \approx -0{,}69 \rightarrow 0{,}69$$
$$\ln 0{,}7 \approx -0{,}36 \rightarrow 0{,}36$$



$$A \xrightarrow{1{,}04} C \xrightarrow{1{,}2} D$$
$$A \xrightarrow{-1{,}04} C \xrightarrow{-1{,}2} D$$
$$A \xrightarrow{0{,}35} C \xrightarrow{0{,}3} D$$

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

най-добра надеждност.

(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.

Върховете са градовете, а ребрата са пътищата. Теглата на ребрата са разстоянията м/у градовете.

Може да добавим нов връх start и да го свържем с всеки град, в който има полиц. участък с ребро с тегло 0. След това пускаме Dijkstra от него и търсим в края връх с макс. число.

(ДИ) (12) Имаме неориентиран свързан граф $G = \langle V, E \rangle$, $w: E \to \{a, b\}$, $0 < a < b$.

Нека r е теглото на някое MST на G.
Как да намерим това r с
линеен алгоритъм, който не е
нужно да намира MST.

Нека разгледаме $G' = \langle V, E' \rangle$
подграф, т.е. $E' \subseteq E$ и за
$E' = E \setminus \{e \mid w(e) = a\}$.
G' може да е свързан и тогава
това ST G' е всъщност ST за G.
Не сме сигурни дали е MST
G'. Знаем със сигурност, че
$r \geq (n-1).a$ със сигурност.
Ако G' не е свързан, то със
сигурност имаме няколко компоненти
на свързаност (поне 2). Нека бележим
броят им с $t \geq 2$ и с $V_1, \dots, V_t$ са
броят върхове във всяка една.
Тогава във всяка компонента
може да има излишни ребра.
От i-тото компонента на свързаност
взимаме $(V_i - 1)$ ребра с тегло a за
(прим. dfs пускане) $i \in \{1, \dots, t\}$, т.е.

запазваме свързаността.
100% още $(t-1)$ тежки ребра ни трябват, тип b трябва да добавим, за да стане свързано дървото.
Всяко едно от ребрата свързва точно две компоненти на свързаност.
Тогава:

$$r = \sum_{i=1}^{t} (v_i - 1).a + (t-1).b =$$

$$= a.\Big(\underbrace{\sum_{i=1}^{t} v_i}_{n} - t\Big) + (t-1).b =$$

$$= a.(n-t) + (t-1).b$$

(13) Създайте ефикасен алгоритъм, който при даден граф G и ребро от него e намира дали в графа има цикъл, съдържащ e.
Махаме реброто e и пускаме bfs, за да видим дали върха v е все още достижим от u.
Ако да, значи e участва в цикъл. Иначе не.

(14) Даден е ориент. граф $G = \langle V, E \rangle$ с $|V| = n$ върха и ребра представени чрез матрица на съседство. Повечето алгоритми, работещи с матрици на съседство, вървят за време от $\Omega(n^2)$, но има и някои изключения. Универсален канал наричаме връх, в който влизат ребра от всички останали върхове, но не излиза нито едно ребро т.е. с входяща степен $n-1$ и изходяща степен 0. Съставете алгоритъм, който намира дали в дадения граф има универсален канал за време $O(n)$.

Нека $A[n][n]$ е матрицата на съседство. Ако u е универсален канал, то всички върхове сочат с ребро към него (колоната на u е пълна с 1-ци с изключение на $A[u][u]$) и той не сочи към

или никого (резултат но и еднозначен с $O(n)$)
Има най-много един универсален
канал в графа (помислете защо).

<u>Идея:</u>

• Нека м-вото S в началото да
съдържа всички потенциални
универсални канали т.е. $S = V$.
• Нека всеки път избираме
двойка различни върхове $\langle x, y \rangle$
проверяваме $A[x][y]$:
- Ако е 1, то махаме x от S
(понеже не трябва да има изходящи)
- Ако е 0, то махаме y от S
(понеже всички различни от него
го сочат).

Така след $(n-1)$ стъпки оставаме с
едно потенциално връхче t.
Проверяваме колоната и редовете
дали отговарят на условието.
Ако да -> има унив. канал, иначе
няма унив. канал този граф.

Сложност: $\underbrace{(n-1)}_{}$ + $\underbrace{2m}_{}$ ≃ n.

за извършване на потенциален цикъл в S

проверка дали реда и колоната на тотг. на условието

---

* Два класа графи автоматично изключват възможност да имат негативни цикли: DAGs и графи само с положителни тегла.

---

(15) Най-къс път и най-дълъг път в DAG: $G = \langle V, E \rangle$ от връх v до всички останали.

Можем да направим topoSort на G и след това да посещаваме върховете в реда на топологичното им сортировка и да изследваме ребрата излизащи от тях.

Важното тук е, че във всеки път в един DAG, върховете се появяват

във възходяща линейна наредба.
Така търсенето на път от
v до всички останали в DAG
става линейно спрямо |V| и |E|.
Модифицираме Bellman-Ford
като преди големия цикъл
правим topoSort след което в
него вземем върховете в реда
определен от topoSort и в
края може да изпуснем търсенето
за негативен цикъл (в DAG
няма цикли).
Най-дългият път от v до всички ще
намерим със същото.
но преди това умножаваме по
-1 всички тегла и тогава
пускаме горния алгоритъм.

Задачи от pdf на Nina.

★ Графика ! топологична наредба ⇒ той е и Хамилтонов път.
По същия начин можем да проверим дали DAG съдържа Хамилтонов път.

(30з) Ориентиран граф с n > 0 върха.
Какво трябва да има графа, за да има:
a) минимален брой топологични сортировки;
b) минимален строго положителен брой топологични сортировки;
c) максимален брой топологични сортировки.

a) → 0 е мин. брой и се достига, ако в графа има цикъл.
b) → 1 е мин. строго пол. брой и се достига, когато графа представлява Хамилтонов път (по ★).
c) → n! е макс. брой и се достига, когато в графа няма ребра.

Други задачи

① Може с dfs като го пуйкнем малко: когато стигнем до връх j приим. принтираме пътя от i до j и след това маркираме j като непосетен, за да генерираме всички прости пътища.

② • NP-complete е. Решаваме я примерно с генериране последователно на всички пермутации на върховете и проверяваме дали е Хамилтонов път: O(n.n!)

• За намиране на Хамилтонов цикъл с один. задачата вижте във файла на колегата.

③ a) Всеки прост път се определя от подмножеството от върховете, през които минава. Ако направим topoSort на тази DAG, тогава път минаващ през някакво подмножество от върхове ще минава през върховете в реда

дадени от топологическата сортировка.
Най-много пътища (прости) от
i до j в DAG може има 2^{n-2}.
заради i и j
always present

В св-с. на колелото има идея как
да се имплементира.
b) Отново NP-complete за произволен
граф, но за DAG има решение с
DP със сложност $O(|V|+|E|)$.
Можем да я разгледаме, когато
стигнем темата DP, ако не
сложим линк в св-с. на колелата
от идеята горе
(4) Диаметър на дърво - най-дълъг път
в дървото. Как за $O(|V|)$ да го
намерим? Първо наблюдение:
най-дългият ще бъде винаги
м/у два листа. Тогава от произволно
връх нека стартираме dfs и
запишем върха, който е най-отдалечен

от него в/и. И е листо и сега
нека от него стартираме dfs,
за да стигнем до най-отдалеченото
от него листо. Това е диаметъра
на дървото. Ще ви прикача pdf
с псевдокод и обяснения.
CLRS 22.2-8 е номера в int.GraphA4.pdf
Може и с bfs в int.GraphA5.pdf пак
за O(|V|) времева сложност

(5) В лекциите има такива алгоритми.
(6) и (7) от doc. на колегата def.
(8) Аз ще кажа за неориентиран
свързан мултиграф д-во.
(9) От лекции отговор и трябва в c) да
е G^T.
(10) Може да използваме алгоритъм
за намиране на Minimal feedback
arc set и, ако е само едно ребро
окей иначе няма такова ребро в графа.
Може да намерите повече инфо
в google. NP-hard е.

(11) В доса на колегата е описал подробно настоеките

(12) В записките на Скелета.

(13) Колегата е дал хубава идея за решение от записките на Скелета.