Алгоритми та складність

II семестр

Лекція 10

Пошук найкоротших шляхів в графі

- Будемо розглядати алгоритми, які дозволяють ефективно розв'язувати задачі пошуку найкоротших шляхів.
- Задано зважений орієнтований граф G = (V, E) з дійсною ваговою функцією w: $E\rightarrow\boldsymbol{R}$.
- *Вага шляху* p = <$v_0, v_1, \ldots, v_k$> – сумарна вага ребер, що входять до нього:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

- *Вага найкоротшого шляху* з вершини u в вершину v:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\}, & \text{якщо існує шлях з u в v,} \\ \infty & \text{інакше.} \end{cases}$$

- Найкоротший шлях з вершини u в вершину v – будь-який шлях, що задовольняє умову w(p) = δ(u,v).

Пошук найкоротших шляхів в графі

- Ваги ребер графа можна інтерпретувати не лише як відстані.
- Вони також можуть представляти часові проміжки, вартості, штрафи, збитки чи іншу величину, яка лінійно накопичується в процесі руху вздовж ребер графа і котру треба мінімізувати.
- Алгоритм пошуку в ширину є алгоритмом пошуку найкоротшого шляху в незваженому графі (вага кожного ребра вважається одиничною).
- Розглянемо задачу про *найкоротші шляхи з однієї вершини*.
- Шлях починається із заданої *вершини-джерела* $s \in V$ і закінчується в кожній з вершин $v \in V$.
- Алгоритм, що розв’язує таку задачу, можна використати для розв’язання низки пов’язаних задач.

Пошук найкоротших шляхів в графі

- <u>Задача про найкоротші шляхи до однієї вершини</u>.

Треба знайти найкоротші шляхи до заданої *цільової* вершини t, які починаються із кожної з вершин v.

Зводиться до задачі про одну вихідну вершину зміною напряму ребер графа.

- <u>Задача про найкоротший шлях між парою вершин</u>

Треба знайти найкоротший шлях із заданої вершини u в вершину v.

Розв'язок елементарно отримується, якщо вже знайдені найкоротші шляхи з вершини u.

Найгірші оцінки відомих алгоритмів, що розв'язують часткову задачу, співпадають з найгіршими оцінками для кращих алгоритмів розв'язання загальнішої задачі.

Пошук найкоротших шляхів в графі

- <u>Задача про найкоротші шляхи між усіма вершинами</u>

Треба знайти найкоротші шляхи з кожної вершини u в кожну вершину v.

Можна розв'язати, шукаючи по черзі найкоротші шляхи з кожної з вершин. Але ефективніше використати спеціальні алгоритми.

- Ідея більшості алгоритмів пошуку найкоротших шляхів: найкоротший шлях між парою вершин містить інші найкоротші шляхи – властивість оптимальної структури.
- Нехай $p = \langle v_0, v_1, \dots, v_k \rangle$ – найкоротший шлях з вершини v_0 до v_k у зваженому орграфі та для довільних i та j ($0 \le i \le j \le k$) шлях $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$ є підшляхом p з вершини v_i до вершини v_j. Тоді p_{ij} – найкоротший шлях з v_i до v_j.

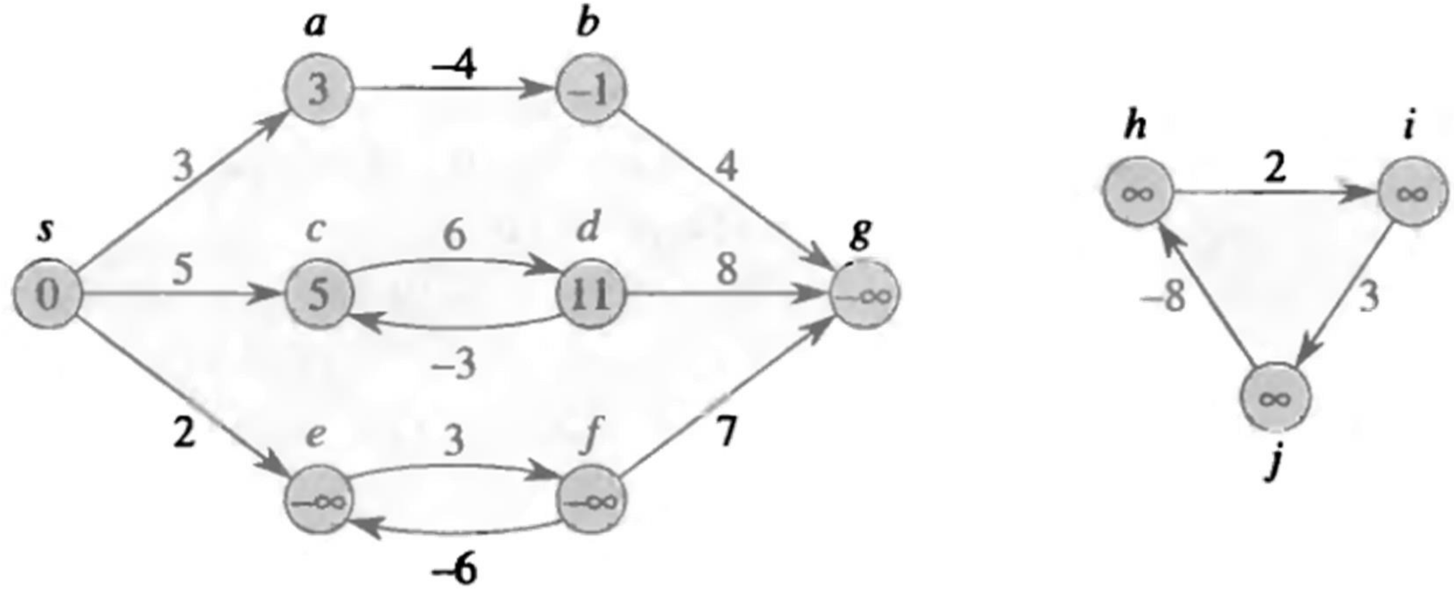
Пошук найкоротших шляхів в графі

Вплив ребер з від'ємною вагою.

- Деякі задачі дозволяють вазі ребер приймати від'ємні значення.
- Якщо граф не містить циклів з від'ємною вагою, досяжних з джерела, вага найкоротших шляхів може бути обчислена (і при цьому може бути від'ємною).
- За наявності досяжного від'ємного циклу визначити найкоротший шлях неможливо: завжди можна знайти ще коротший шлях, обійшовши цикл.
- Якщо на шляху з s до v є цикл з від'ємною вагою, покладемо $\delta(s,v) = -\infty$.
- Деякі алгоритми вимагають невід'ємної ваги ребер, інші допускають від'ємні ребра. Останні зазвичай дозволяють виявляти цикли від'ємної ваги.

Пошук найкоротших шляхів в графі

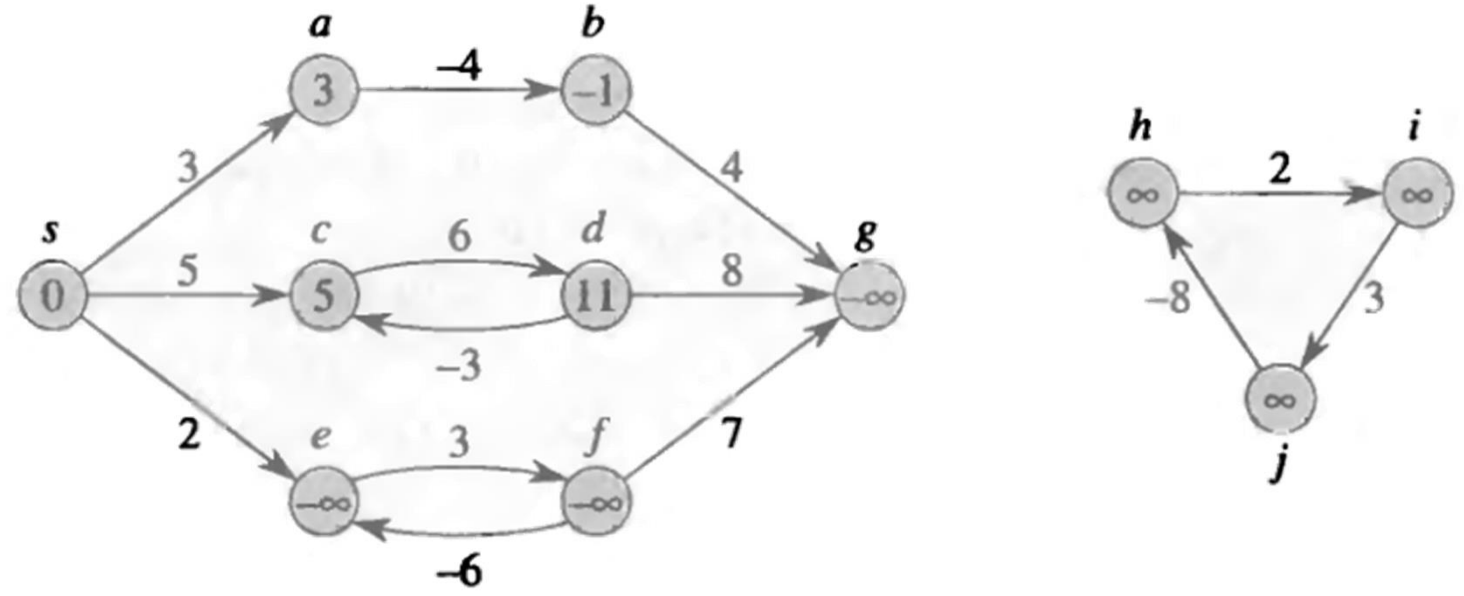
Вплив ребер з від'ємною вагою. В кожній вершині вказана вага найкоротшого шляху до неї з джерела s.



- Вершини h, i, j недосяжні з джерела, тому $\delta(s,i) = \delta(s,j) = \delta(s,h) = \infty$ незважаючи на те, що вони утворюють цикл з від'ємною вагою.

Пошук найкоротших шляхів в графі

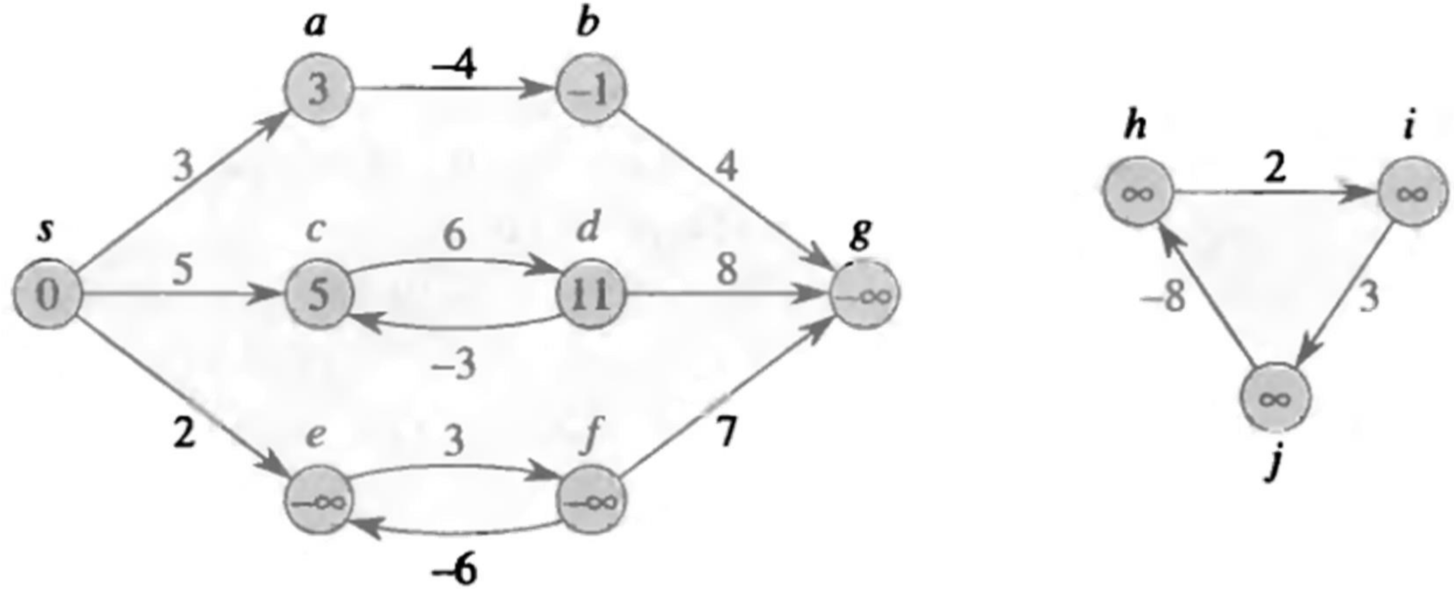
Вплив ребер з від'ємною вагою. В кожній вершині вказана вага найкоротшого шляху до неї з джерела s.



- δ(s,b) = 3+ (–4) = –1 – від'ємний шлях.
- δ(s,c) = 5 – при цьому потрапити до вершини c можна через нескінченну кількість шляхів (за рахунок циклу <c,d,c>, але вага його додатна).

Пошук найкоротших шляхів в графі

Вплив ребер з від'ємною вагою. В кожній вершині вказана вага найкоротшого шляху до неї з джерела s.



- δ(s,e) = –∞ – до вершини e існує нескінченна кількість шляхів через цикл <e,f,e>, і вага його від'ємна.
- δ(s,f) = –∞ – аналогічно.
- δ(s,g) = –∞ – вершина досяжна з f.

Пошук найкоротших шляхів в графі

Вплив циклів.

- Найкоротший шлях не може містити цикл з від’ємною вагою.
- Найкоротший шлях не може містити цикл з додатною вагою (видалення циклу зі шляху лише зменшить його вагу).
- Якщо в шляху є цикл з нульовою вагою, його можна вилучити без впливу на вагу.
- Отже, можна вважати, що будуть знаходитися найкоротші шляхи без циклів.
- В довільний ациклічний шлях графа $G = (V, E)$ входить не більше $|V|$ вершин та $|V|–1$ ребер, тому обмежимося розглядом найкоротших шляхів з не більш ніж $|V|–1$ ребра.

Пошук найкоротших шляхів в графі

Представлення найкоротших шляхів з джерела s.

- В заданому графі G = (V, E) до кожної вершини v додамо атрибут її попередника ($v.\pi$) та використаємо підграф передування $G_\pi = (V_\pi, E_\pi)$, визначений як в алгоритмі пошуку в ширину:

$$V_\pi=\{v\in V:\ v.\pi \neq \text{NIL}\}\cup\{s\},$$

$$E_\pi=\{(v.\pi,v)\in E:\ v\in V_\pi - \{s\}\}.$$

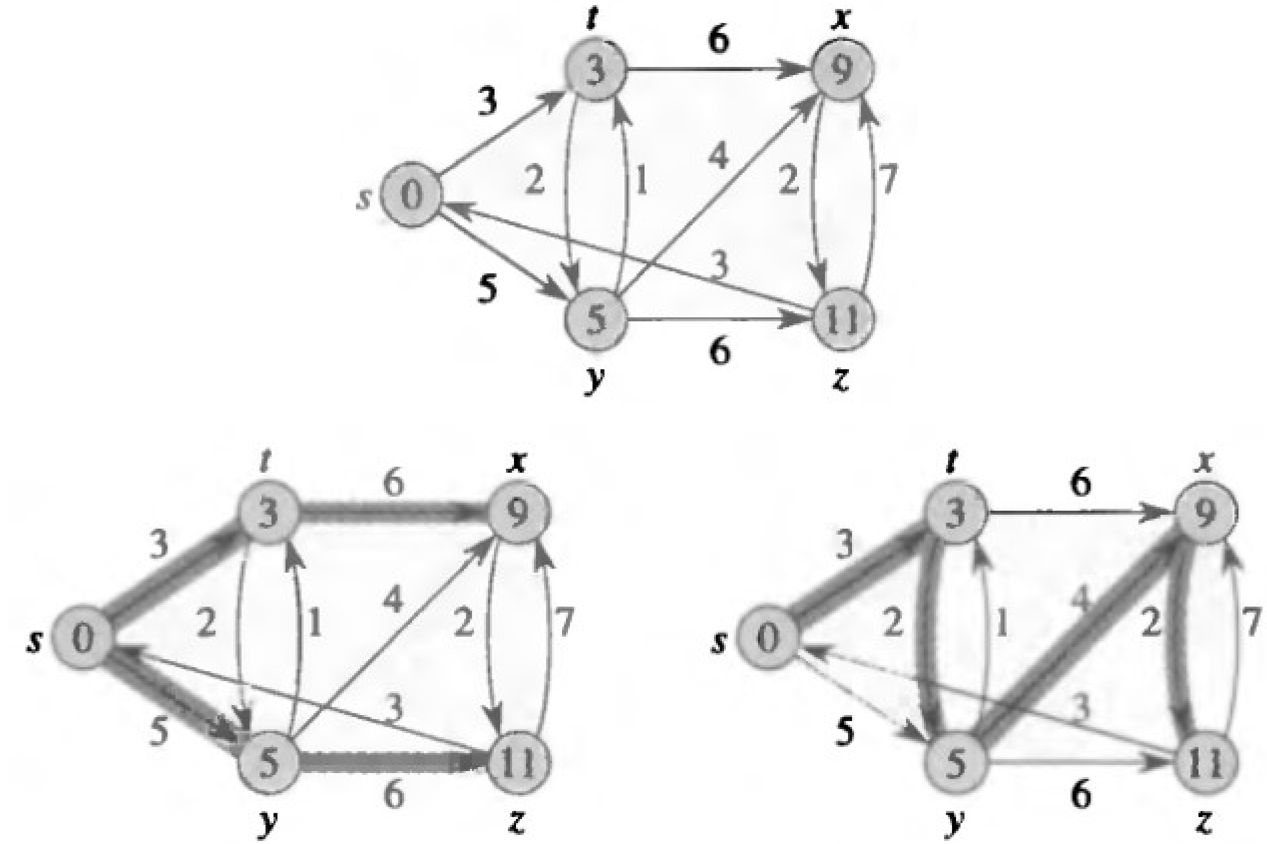
- Тоді в результаті роботи алгоритму в G_π отримаємо дерево найкоротших шляхів: на відміну від дерева пошуку в ширину, воно містить найкоротші шляхи з джерела, що визначаються не кількістю ребер, а їх вагою.

- Для заданої вершини v ($v.\pi\neq$NIL) можна вивести найкоротший шлях з s до v використавши процедуру PRINT-PATH(G,s,v).

Пошук найкоротших шляхів в графі

- Нехай $G = (V, E)$ – зважений орієнтований граф з дійсною ваговою функцією w. Нехай він не містить циклів з від'ємною вагою, досяжних з джерела $s \in V$, так що найкоротші шляхи цілком визначені.
- Тоді *дерево найкоротших шляхів* з коренем в s є підграфом $G^* = (V^*, E^*)$, де $V^* \subseteq V$ і $E^* \subseteq E$ визначені так:
  - V^* є множиною вершин, досяжних з джерела s графа G;
  - граф G^* утворює кореневе дерево з коренем s;
  - для всіх $v \in V^*$ простий шлях з вершини s до вершини v визначається однозначно в графі G^* та є найкоротшим шляхом з s до v в графі G.
- Найкоротші шляхи і дерева найкоротших шляхів є не обов'язково єдиними.

Пошук найкоротших шляхів в графі



• Зважений орграф та два дерева найкоротших шляхів з коренем в джерелі s.

• У вершинах вказані ваги найкоротшого шляху до них з джерела s.

Пошук найкоротших шляхів в графі

Метод релаксації (ослаблення).

- Для кожної вершини підтримується атрибут v.d, що представляє верхню границю ваги найкоротшого шляху з джерела s в v – *оцінка найкоротшого шляху*.
- Ініціалізація атрибутів вершин виглядає так:

INITIALIZE-SINGLE-SOURCE(G, s)

for каждой вершины $v \in G.V$
    $v.d = \infty$
    $v.\pi = \text{NIL}$
$s.d = 0$

- В результаті для всіх v$\in$V буде виконуватися v.π = NIL, s.d = 0 та v.d = ∞ для v$\in$V–{s}.

Пошук найкоротших шляхів в графі

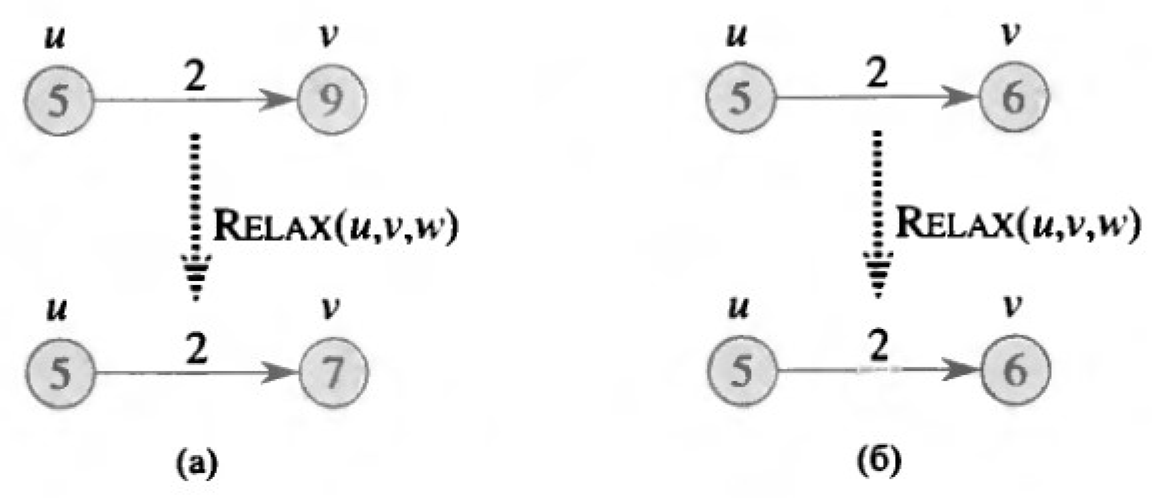
Метод релаксації (ослаблення).

- Процес *ослаблення* (*релаксації*) ребра (u,v) полягає в перевірці, чи не можна покращити поточний знайдений найкоротший шлях до вершини v, проходячи через вершину u.
- Ослаблення може зменшити оцінку v.d та оновити v.π.
- Процедура виконується за константний час:

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

- Термін «ослаблення» пов’язаний з ослабленням обмеження v.d $\leq$ u.d + w(u,v).

Пошук найкоротших шляхів в графі



- Приклад ослаблення ребра (u,v) вагою w(u,v)=2.
- У вершинах – оцінки їх найкоротших шляхів.

(а) 9 > 5 + 2: значення v.d зменшується до 7.

(б) 6 $\leq$ 5 + 2: ослаблення не змінює v.d.

Пошук найкоротших шляхів в графі

Метод релаксації (ослаблення).

- Суть ряду алгоритмів пошуку найкоротших шляхів полягає у виконанні ослаблення ребер після попередньої ініціалізації.
- Алгоритми відрізняються кількістю проведених релаксацій, а також порядком ребер, над якими виконується ця дія.
- Ослаблення – єдина операція, що може змінювати оцінки найкоротших шляхів та попередників вершин.

Пошук найкоротших шляхів в графі

Властивості найкоротших шляхів та ослаблення.

- Нерівність трикутника. Для кожного ребра $(u,v) \in E$ виконується $\delta(s,v) \le \delta(s,u) + w(u,v)$
- Властивість верхньої границі. Для всіх вершин $v \in V$ завжди справджується $v.d \ge \delta(s,v)$, а після досягнення рівності значення v.d більше не змінюється.
- Властивість відсутності шляху. Якщо немає шляху з s до v, завжди виконується $v.d = \delta(s,v) = \infty$.
- Властивість збіжності. Якщо шлях $s \leadsto u \to v$ є найкоротшим в графі для деяких $u,v \in V$ та $u.d = \delta(s,u)$ в будь-який момент до ослаблення (u,v), то $v.d = \delta(s,v)$ буде виконуватися в будь-який момент після нього.

Пошук найкоротших шляхів в графі

Властивості найкоротших шляхів та ослаблення.

- Властивість ослаблення шляху. Якщо маємо найкоротший шлях $p = \langle v_0, v_1, \ldots, v_k \rangle$ з $s = v_0$ до v_k та ребра p ослаблюються в порядку (v_0,v_1), (v_1,v_2), …, (v_{k-1},v_k), то $v_k.d = \delta(s,v_k)$. Властивість виконується незалежно від інших етапів релаксації, навіть якщо вони чергуються з ослабленням ребер шляху p.
- Властивість підграфа передування. Якщо для всіх вершин $v \in V$ виконується $v.d = \delta(s,v)$, то підграф передування є деревом найкоротших шляхів з коренем у джерелі s.

- Надалі вважатимемо $a + \infty = \infty + a = \infty$ для $a \neq -\infty$ та $a + (-\infty) = (-\infty) + a = -\infty$ для $a \neq \infty$.

Найкоротші шляхи з однієї вершини

- Нехай граф зберігається у вигляді списків суміжності та разом з ребром зберігається його вага.
- *Алгоритм Беллмана-Форда* розв'язує задачу про пошук найкоротшого шляху з фіксованого джерела s в загальній постановці: допускаються від'ємні ваги ребер та цикли.
- Повертається логічне значення, що вказує на наявність в графі циклу від'ємної ваги, досяжного з джерела. Якщо такий цикл існує, повідомляється про неіснування розв'язку, інакше виводяться найкоротші шляхи та їх вага.
- Здійснюється серія ослаблень, внаслідок чого оцінка найкоротшого шляху v.d для кожної вершини $v \in V$ зменшується, поки не стане рівною $\delta(s,v)$.

Найкоротші шляхи з однієї вершини

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for каждого ребра (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for каждого ребра (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

- Після ініціалізації значень d та π відбувається |V|–1 проходів по ребрам графа, при кожному з яких здійснюється однократне ослаблення кожного з ребер (внутрішній цикл **for** в рядках 3–4).
- Після завершення циклу в рядках 2–4 для всіх вершин $v \in V$, досяжних з s, виконується $v.d = \delta(s,v)$ за умови відсутності в графі циклів з від’ємною вагою.

Найкоротші шляхи з однієї вершини

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for каждого ребра (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for каждого ребра (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

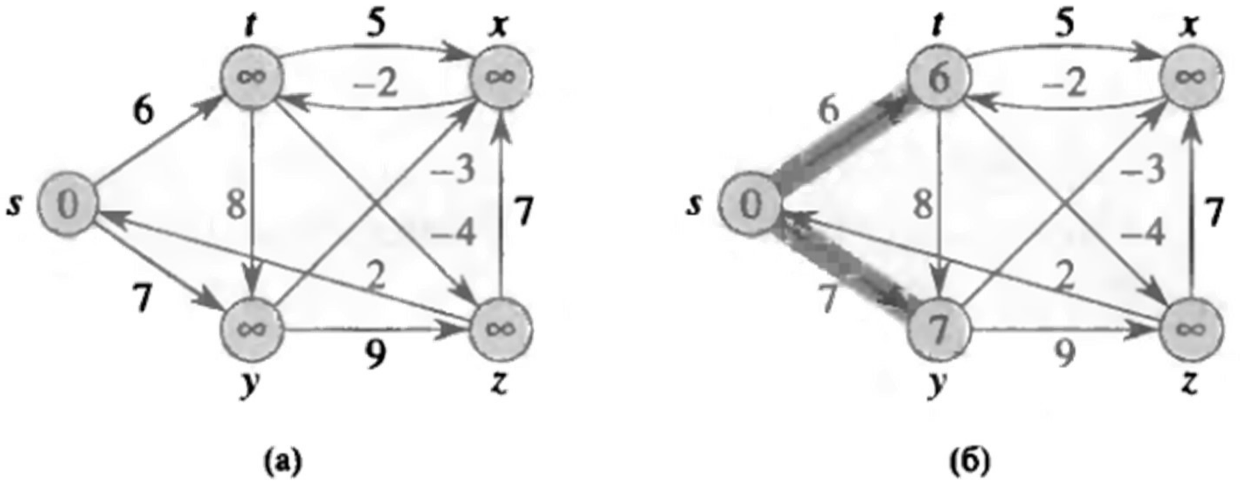
- Цикл в рядках 5–7 перевіряє наявність циклу з від’ємною вагою: якщо перевірка умови ослаблення виконується, значить вага шляху може бути ще зменшена, чого не може статися без від’ємного циклу.
- Варто зауважити, що шлях з вершини s до вершини $v \in V$ існує $\Leftrightarrow$ після застосування процедури BELLMAN-FORD справджується $v.d < \infty$.

Найкоротші шляхи з однієї вершини

- Час роботи алгоритму $O(V \cdot E)$:
  - – ініціалізація: $\Theta(V)$,
  - – подвійний цикл в рядках 2–4: $|V|-1$ проходів по $\Theta(E)$ ребрам,
  - – цикл в рядках 5–7: $O(E)$.
- Отже, якщо алгоритм BELLMAN-FORD застосовується до зваженого орієнтованого графа G з джерелом s та дійсною функцією ваги w, то він поверне TRUE, для всіх вершин $v \in V$ буде виконуватися $v.d = \delta(s,v)$ та підграф передування G_π буде деревом найкоротших шляхів з коренем в s за умови відсутності в графі циклів з від’ємною вагою, досяжних з s. Якщо граф міститиме досяжний з s цикл з від’ємною вагою, алгоритм поверне FALSE.

Найкоротші шляхи з однієї вершини

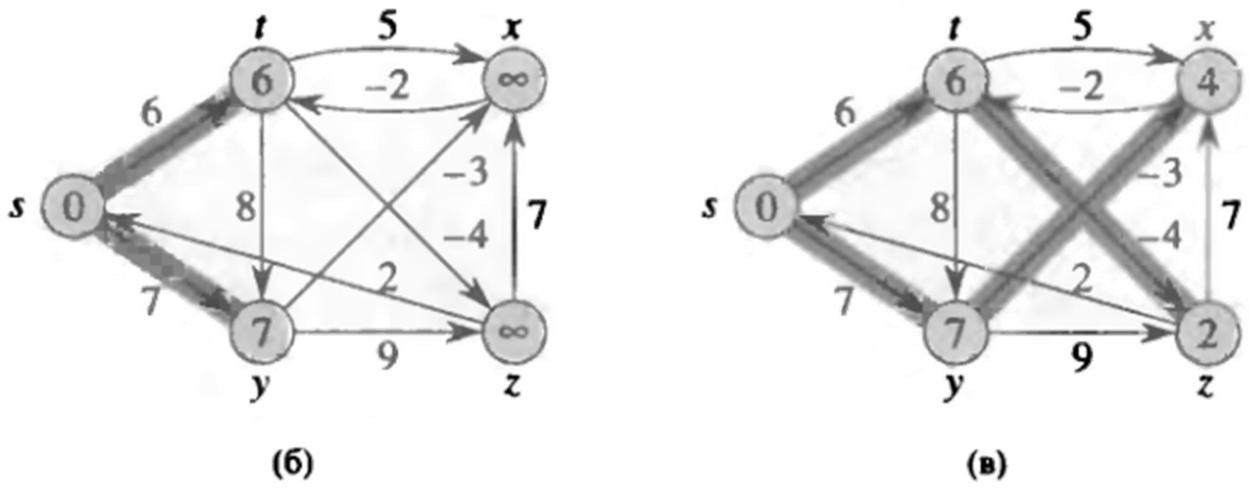
- Приклад роботи алгоритму Беллмана-Форда (1)



- Вершини містять значення d, затемнені ребра вказують попередників.
- Порядок проходу по ребрам при ослабленні: (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

Найкоротші шляхи з однієї вершини

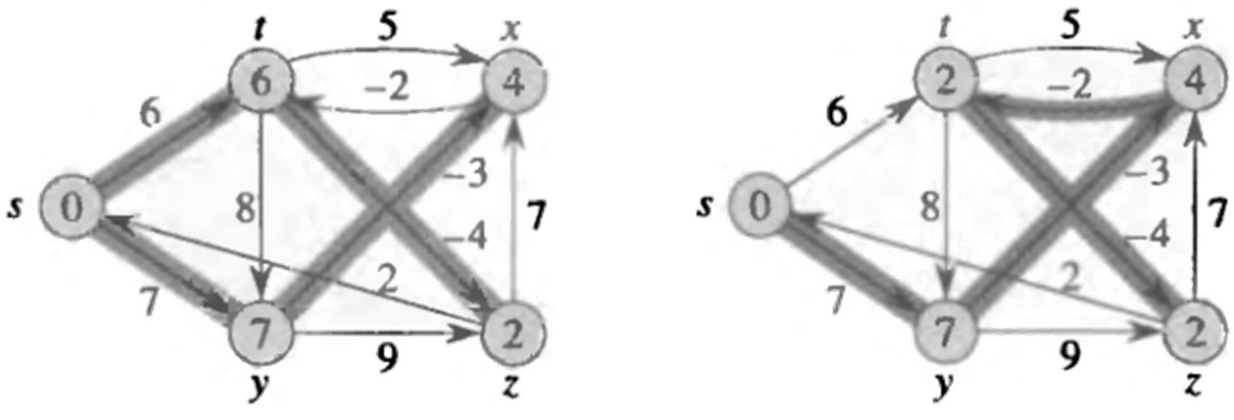
- Приклад роботи алгоритму Беллмана-Форда (2)



- Вершини містять значення d, затемнені ребра вказують попередників.
- Порядок проходу по ребрам при ослабленні: (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

Найкоротші шляхи з однієї вершини

- Приклад роботи алгоритму Беллмана-Форда (3)

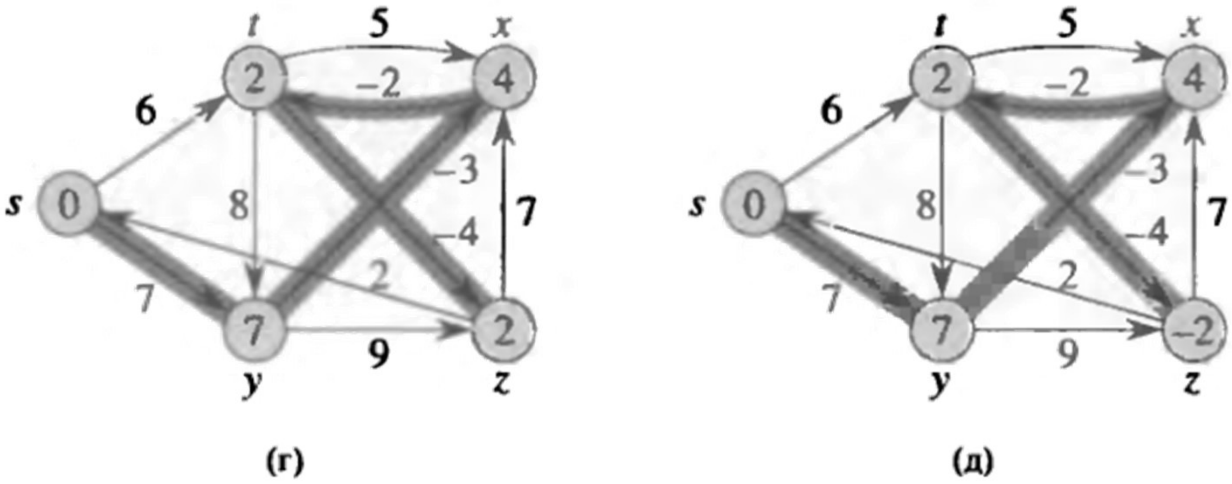


(в) (г)

- Вершини містять значення d, затемнені ребра вказують попередників.
- Порядок проходу по ребрам при ослабленні: (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

Найкоротші шляхи з однієї вершини

- Приклад роботи алгоритму Беллмана-Форда (4)



- Вершини містять значення d, затемнені ребра вказують попередників.
- Порядок проходу по ребрам при ослабленні: (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

Найкоротші шляхи з однієї вершини

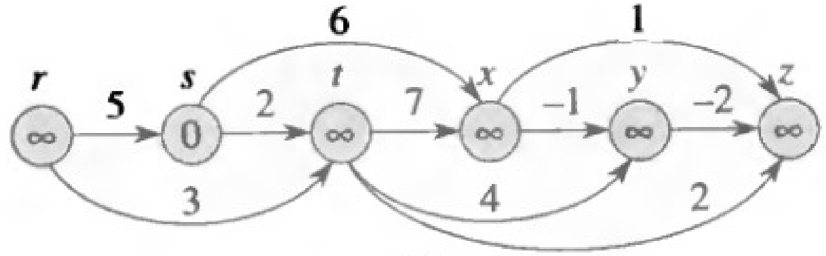
- Розглянемо випадок орієнтованих ациклічних графів.
- В таких графах найкоротші шляхи завжди цілком визначені.
- Спочатку виконується топологічне сортування графа.
- Якщо в графі існує шлях з вершини u в вершину v, то після топологічного сортування u передуватиме v.
- Вершини один раз переглядаються в топологічному порядку, при цьому виконується ослаблення всіх ребер, які виходять з поточної вершини.

Найкоротші шляхи з однієї вершини

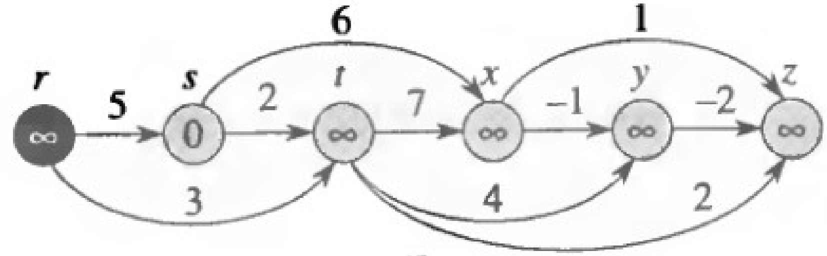
```
DAG-SHORTEST-PATHS(G, w, s)
Топологическая сортировка вершин графа G
INITIALIZE-SINGLE-SOURCE(G, s)
for каждой вершины u в порядке топологической сортировки
    for каждой вершины v ∈ G.Adj[u]
        RELAX(u, v, w)
```

- Час роботи топологічного сортування: $\Theta(V+E)$.
- Ініціалізація: $\Theta(V)$.
- Зовнішній цикл **for** переглядає всі вершини, внутрішній цикл загалом ослабляє кожне ребро по одному разу (груповий аналіз), тому час роботи кожної ітерації внутрішнього **for** константний.
- Загальний час роботи алгоритму $\Theta(V+E)$ – лінійна залежність від представлення графа списками суміжності.

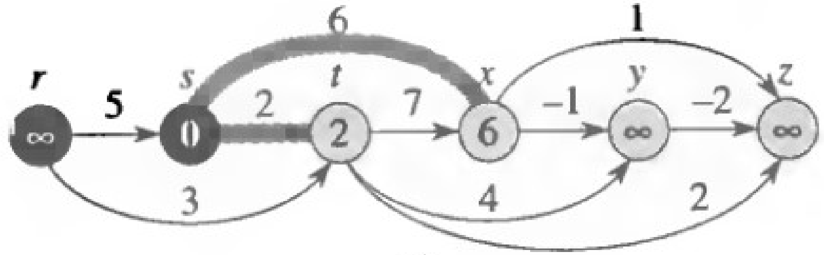
Найкоротші шляхи з однієї вершини



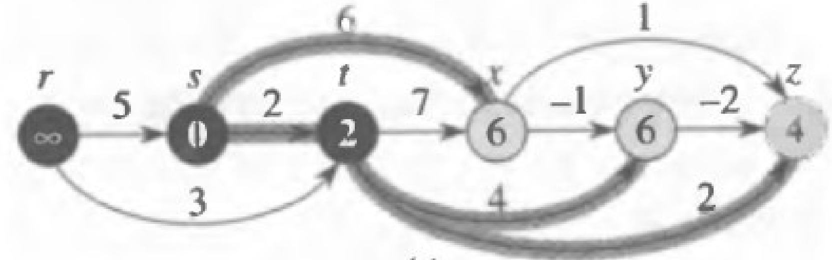
(а)



(б)



(в)

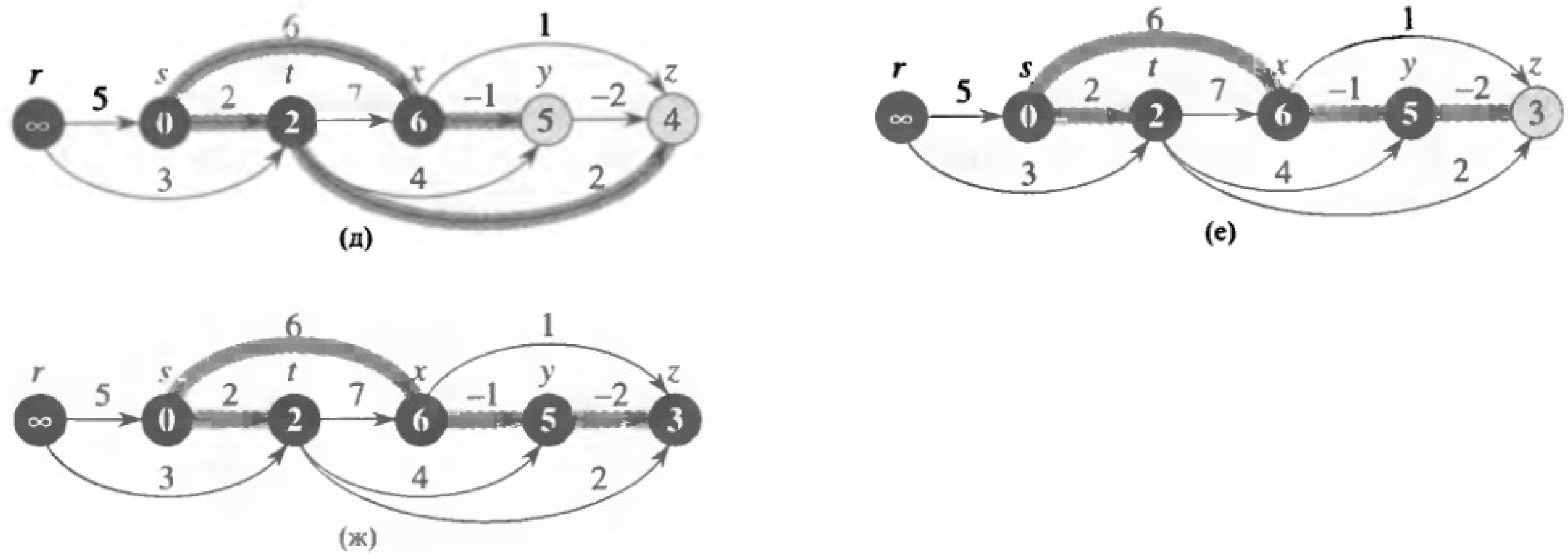


(г)

Приклад роботи алгоритму DAG-SHORTEST-PATH (1)

- Вершини топологічно відсортовані зліва направо, s – джерело.
- У вершинах вказані поточні значення атрибута d.
- Затемнені ребра – підграф передування.

Найкоротші шляхи з однієї вершини



(д) (е) (ж)

Приклад роботи алгоритму DAG-SHORTEST-PATH (2)

- Вершини топологічно відсортовані зліва направо, s – джерело.
- У вершинах вказані поточні значення атрибута d.
- Затемнені ребра – підграф передування.

Найкоротші шляхи з однієї вершини

- Якщо ребра ациклічного орграфа представляють завдання, їх ваги – час, необхідний на виконання завдання, то шлях в графі означає послідовність виконання завдань.
- Найдовший шлях відповідає найбільшому часу, необхідному для виконання упорядкованої послідовності завдань.
- Розглянутий алгоритм DAG-SHORTEST-PATH можна модифікувати для пошуку *критичного шляху* – найдовшого шляху в орієнтованому ациклічному графі.
- Перший спосіб. Знаки ваг ребер в графі замінюються на протилежні і виконується алгоритм DAG-SHORTEST-PATH.
- Другий спосіб. В алгоритм DAG-SHORTEST-PATH вносяться наступні зміни. При ініціалізації значення ∞ замінюються на $-\infty$, а в процедурі релаксації знак $>$ змінюється на $<$.

Найкоротші шляхи з однієї вершини

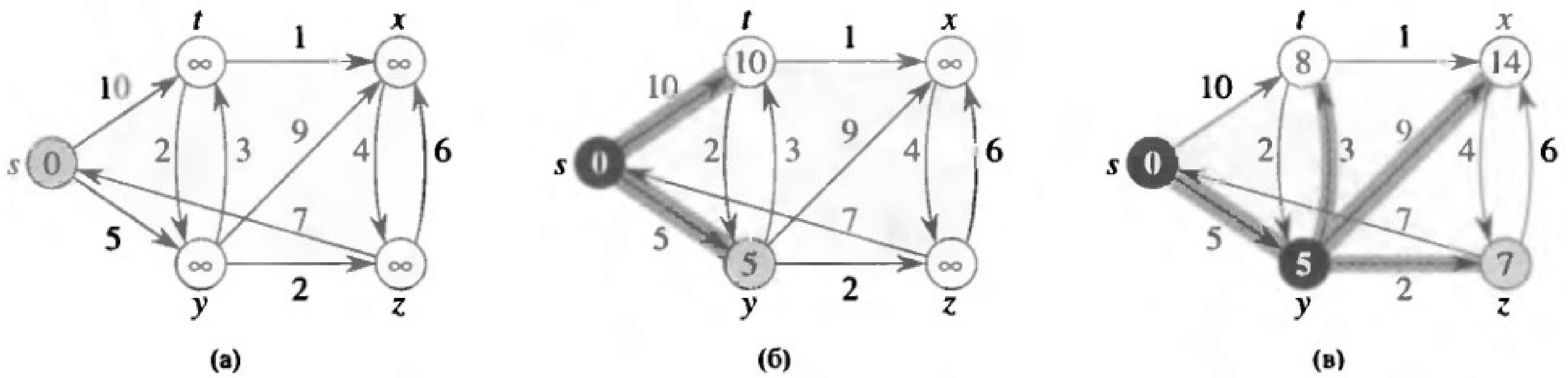
- *Алгоритм Дейкстри* розв'язує задачу про пошук найкоротшого шляху з фіксованого джерела s в орієнтованому графі у випадку невід'ємних ваг ребер.
- Для всіх ребер $(u,v) \in E$ будемо вважати $w(u,v) \geq 0$.
- Підтримується множина вершин S, для яких вже обчислені остаточні ваги найкоротших шляхів до них з джерела s.
- Алгоритм вибирає чергову вершину $u \in V - S$, якій відповідає мінімальна на цей момент оцінка найкоротшого шляху.
- Після додавання u до множини S проводиться ослаблення всіх ребер, які виходять з цієї вершини.
- Вершини організовані у неспадаючу чергу з пріоритетами Q, в ролі ключів виступають значення d.

Найкоротші шляхи з однієї вершини

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for каждой вершины v ∈ G.Adj[u]
        RELAX(u, v, w)
        // С соответствующими вызовами DECREASE-KEY
```

- В алгоритмі Дейкстри з множини V–S завжди вибирається найлегша (найближча) вершина, отже, використовується жадібна стратегія.
- Можна показати, що після кожного додавання вершини u до множини S справджується u.d = δ(s,u).

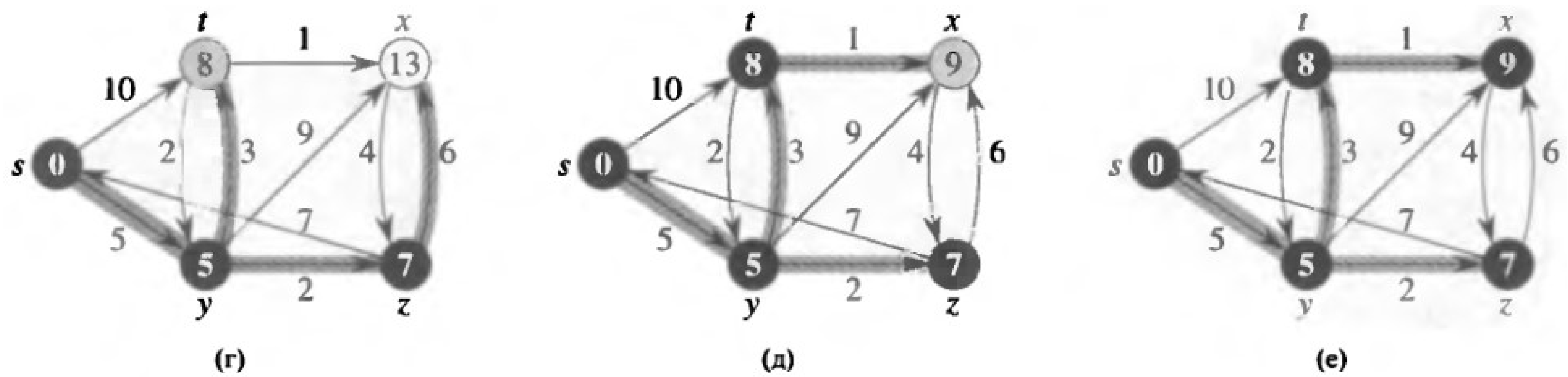
Найкоротші шляхи з однієї вершини



Робота алгоритму Дейкстри (1)

- Джерелом є s, у вершинах вказані оцінки найкоротших шляхів, затемнені ребра показують підграф передування.
- Чорні вершини належать до S, білі – до черги з пріоритетами Q=V–S.
- Сіра вершина має поточне мінімальне значення d.

Найкоротші шляхи з однієї вершини



Робота алгоритму Дейкстри (2)

- Джерелом є s, у вершинах вказані оцінки найкоротших шляхів, затемнені ребра показують підграф передування.
- Чорні вершини належать до S, білі – до черги з пріоритетами Q=V–S.
- Сіра вершина має поточне мінімальне значення d.

Найкоротші шляхи з однієї вершини

- Час роботи алгоритму Дейкстри залежить від реалізації неспадаючої черги з пріоритетами.
- Якщо використовується звичайний масив, час роботи $O(V^2)$.
- У випадку досить розрідженого графа (умова $E=o(V^2/\log V)$) можна використати бінарну піраміду. Тому повний час роботи $O((V+E) \log V)$, або ж якщо всі вершини досяжні з джерела, то $O(E \log V)$.
- Якщо використати піраміду Фібоначчі, можна досягти часу $O(V \log V + E)$.
- Розвиток пірамід Фібоначчі був зумовлений спостереженням на алгоритмом Дейкстри, що викликає DECREASE-KEY набагато частіше, ніж EXTRACT-MIN, тому постало питання пришвидшення амортизованого часу операції зменшення ключа.

Найкоротші шляхи з однієї вершини

- Можна знайти схожість алгоритму Дейкстри з пошуком в ширину та алгоритмом Прима.
- Множина S відповідає множині чорних вершин пошуку в ширину. Так само, як вершинам S співставляються остаточні ваги найкоротших шляхів, так і чорним вершинам при пошуку в ширину співставляються правильні відстані.
- Як і в алгоритмі Прима, за допомогою неспадаючої черги з пріоритетами знаходиться «найлегша» вершина поза межами заданої множини (множина S чи будоване кістякове дерево), ця вершина додається до множини з подальшим корегуванням та упорядкуванням ваг невикористаних вершин.

Найкоротші шляхи між усіма парами вершин

- Будемо вважати, що граф представлений у вигляді матриці суміжності $W = (w_{ij})$ розміром $n\times n$, що зберігає ваги орієнтованих ребер:

$$w_{ij} = \begin{cases} 0\ , & \text{если } i = j\ , \\ \text{вес дуги } (i,j)\ , & \text{если } i \neq j \text{ и } (i,j) \in E\ , \\ \infty\ , & \text{если } i \neq j \text{ и } (i,j) \notin E\ . \end{cases}$$

- Допускаються ребра від'ємної ваги, але не цикли з від'ємною вагою.
- На виході отримується матриця $D = (d_{ij})$ розміром $n\times n$, де d_{ij} містить вагу найкоротшого шляху між вершинами i та j. Тобто, в кінці роботи $d_{ij} = \delta(i,j)$.
- Шляхи зберігаються в матриці передування $\Pi = (\pi_{ij})$, де π_{ij}=NIL, якщо i=j або шлях з i до j відсутній, інакше π_{ij} – попередник вершини j в деякому найкоротшому шляху з вершини i.

Найкоротші шляхи між усіма парами вершин

- Визначимо для кожної вершини $i \in V$ підграф передування графа G для вершини-джерела i як граф $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$:

$$V_{\pi,i}=\{j \in V: \pi_{ij} \neq NIL\} \cup \{i\},$$

$$E_{\pi,i}=\{(\pi_{ij},j): j \in V_{\pi,i} - \{i\}\}.$$

- Якщо $G_{\pi,i}$ – дерево найкоротших шляхів, змінимо процедуру виводу шляхів, щоб вона виводила найкоротший шлях з вершини i до вершини j:

```
PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, j)
if i == j
    print i
elseif π_ij == NIL
    print "Пути из" i "в" j "не существует"
else PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, π_ij)
    print j
```

Найкоротші шляхи між усіма парами вершин

- *Алгоритм Флойда-Воршелла* реалізує підхід динамічного програмування.
- Вводиться поняття *проміжної вершини* простого шляху p = <$v_1, v_2, \ldots, v_n$>. Це – будь-яка вершина p, крім v_1 та v_n.
- Позначимо $d_{ij}^{(k)}$ – вага найкоротшого шляху з вершини i до j з проміжними вершинами із множини {1,2,…,k}:

$$d_{ij}^{(k)} = \begin{cases} w_{ij}, & \text{если } k = 0, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right), & \text{если } k \geq 1. \end{cases}$$

- У випадку k = 0 шлях з i до j не містить проміжних вершин та має не більше одного ребра: $d_{ij}^{(0)} = w_{ij}$.
- Матриця $D^{(n)} = (d_{ij}^{(n)})$ дасть остаточну відповідь для всіх пар вершин i та j: $d_{ij}^{(n)} = \delta(i,j)$.

Найкоротші шляхи між усіма парами вершин

FLOYD-WARSHALL(W)

1 $n = W.rows$
2 $D^{(0)} = W$
3 **for** $k = 1$ **to** n
4   Пусть $D^{(k)} = \left(d_{ij}^{(k)}\right)$ — новая матрица размером $n \times n$
5   **for** $i = 1$ **to** n
6   **for** $j = 1$ **to** n
7             $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
8 **return** $D^{(n)}$

- Алгоритм послідовно обчислює $d_{ij}^{(k)}$ в порядку зростання k. Час виконання рядка 7 константний, отже весь алгоритм виконується за $\Theta(n^3)$.
- Через простоту алгоритму і відсутність складних структур даних закладена в $\Theta(n^3)$ константа є малою, тому алгоритм Флойда-Воршелла має практичну цінність навіть для графів середнього розміру.

Найкоротші шляхи між усіма парами вершин

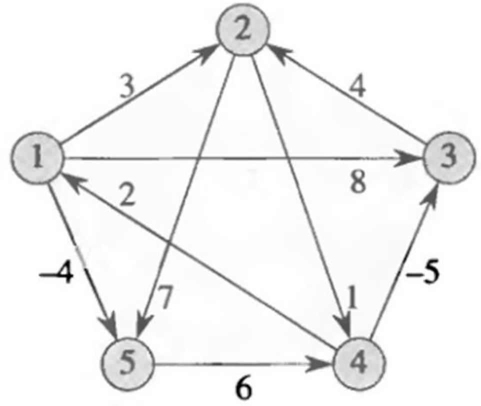
- Матрицю передування Π можна обчислювати паралельно з D як послідовність $\Pi^{(0)},\Pi^{(1)},\ldots,\Pi^{(n)}$, де $\Pi=\Pi^{(n)}$, а $\pi_{ij}^{(k)}$ – попередник вершини j на найкоротшому шляху з вершини і з проміжними вершинами із множини {1,2,…,k}.
- Рекурсивно визначимо $\pi_{ij}^{(k)}$. При k = 0 шлях з і до j не містить проміжних вершин, тому

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL}\ , & \text{если } i = j \text{ или } w_{ij} = \infty\ , \\ i\ , & \text{если } i \neq j \text{ и } w_{ij} < \infty\ . \end{cases}$$

- Для k $\geq$ 1 отримаємо:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)}\ , & \text{если } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\ , \\ \pi_{kj}^{(k-1)}\ , & \text{если } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\ . \end{cases}$$

Найкоротші шляхи між усіма парами вершин

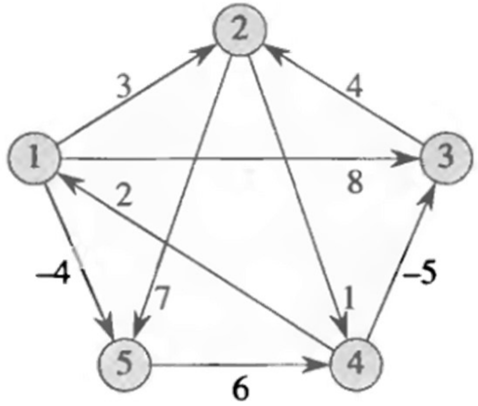


$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Послідовність матриць $D^{(k)}$ та $\Pi^{(k)}$, обчислених алгоритмом Флойда-Воршелла (1)

Найкоротші шляхи між усіма парами вершин

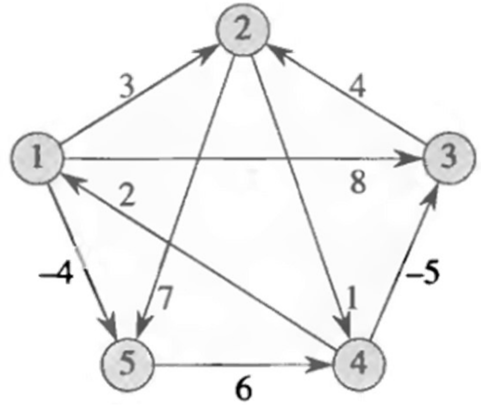


$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Послідовність матриць $D^{(k)}$ та $\Pi^{(k)}$, обчислених алгоритмом Флойда-Воршелла (2)

Найкоротші шляхи між усіма парами вершин



$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Послідовність матриць $D^{(k)}$ та $\Pi^{(k)}$, обчислених алгоритмом Флойда-Воршелла (3)

Найкоротші шляхи між усіма парами вершин

- Модифікацію алгоритму Флойда-Воршелла можна використати для пошуку транзитивного замикання графа.
- *Транзитивним замиканням* графа G = (V,E) є граф G′= (V,E′), де

  E′={(i,j): граф G містить шлях від вершини i до j}.
- Замість арифметичних операцій min та + використовуються логічні операції OR та AND відповідно.
- Обчислюються матриці $T^{(k)} = (t_{ij}^{(k)})$ за зростанням k, де $t_{ij}^{(k)} = t_{ij}^{(k-1)}$ OR ($t_{ik}^{(k-1)}$ AND $t_{kj}^{(k-1)}$).

Найкоротші шляхи між усіма парами вершин

TRANSITIVE-CLOSURE(G)

$n = |G.V|$
Пусть $T^{(0)} = \left(t_{ij}^{(0)}\right)$ — новая матрица размером $n \times n$
for $i = 1$ **to** n
    for $j = 1$ **to** n
        if $i == j$ или $(i, j) \in G.E$
            $t_{ij}^{(0)} = 1$
        else $t_{ij}^{(0)} = 0$
for $k = 1$ **to** n
    Пусть $T^{(k)} = \left(t_{ij}^{(k)}\right)$ — новая матрица размером $n \times n$
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$
return $T^{(n)}$

Найкоротші шляхи між усіма парами вершин

- Якщо розглядається розріджений граф, для пошуку найкоротших шляхів між усіма парами вершин можна використати *алгоритм Джонсона*.
- Алгоритм або повертає матрицю, що містить ваги найкоротших шляхів для всіх пар вершин, або повідомляє, що вхідний граф містить цикл від'ємної ваги.
- Граф представляється у вигляді списків суміжності.
- В якості підпрограм використовуються алгоритм Беллмана-Форда та алгоритм Дейкстри.
- Складність алгоритму Джонсона $O(V^2 \log V + V \cdot E)$ при реалізації черги з пріоритетами в алгоритмі Дейкстри через піраміди Фібоначчі.

ТЕМИ НА ТЕСТ

- Елементарні алгоритми на графах (Пошук в ширину, пошук в глибину, класифікація ребер. Топологічне сортування. Сильно зв'язні компоненти.)
- Елементи динамічного програмування
- Елементи жадібної стратегії
- Найкоротші шляхи з однієї вершини.
- Найкоротші шляхи між усіма парами вершин.
- Алгоритми побудови мінімальних кістякових дерев.