Слайд 2

Алгоритм Дейкстри — це алгоритм на графах, що розроблений нідерландським вченим Едсгером Дейкстрою в 1959 році.

Слайд 3

В його основу закладений пошук найкоротшого шляху від однієї з вершин графа до всіх інших.

Алгоритм працює лише з графами без ребер відʼємного значення.

Алгоритм широко використовується в програмуванні, наприклад, в протоколах маршрутизації OSPF та IS-IS.

Слайд 4

Кожній вершині з множини вершин V необхідно зіставляємо мітку — мінімальну відому відстань від цієї вершини до початкової вершини а.

Алгоритм працює покроково — на кожному кроці він «відвідує» одну вершину й намагається зменшити індекс.

Алгоритм завершується після проходження всіх вершин.

Слайд 5

Індекс самої вершини а дорівнює 0, індекси інших вершин — нескінченність.

Це є відображенням того, що відстань від вершини а до інших вершин є невідомою.

На початковому кроці всі вершини графа є невідвіданими.

Слайд 6

Якщо всі вершини є відвіданими, то алгоритм завершує свою дію.

В іншому випадку, з невідвіданих вершин обирається вершина u, що має мінімальний індекс.

Ми розглядаємо різні маршрути, в яких u є передостаннім пунктом. Вершини, що сполучені ребрами з вершини u називаються сусідами цієї вершини. Для кожного сусіда вершини u, крім зазначених як відвідані, розглянемо нову довжину шляху, що дорівнює сумі значень поточної мітки u та довжини ребра, що з'єднує u з цим сусідом.

Якщо отримане значення довжини менше за значення індексу сусіда, то необхідно замінити індекс отриманим значенням.

Слайд 7

Необхідно знайти найкоротші маршрути з вершини 1 до всіх інших.

Слайд 8

Починаємо з вершини 1, її сусідами є вершини 2, 3 та 6. (малюнок 1)

Перший по черзі сусід вершини 1 — вершина 2, оскільки довжина шляху є мінімальною.

Довжина шляху у неї з вершини 1 дорівнює сумі значень індексу вершини 1 та довжини ребра, що йде з 1-ої у 2, тобто 0+7=7. І оскільки отримане значення 7 менше за нескінченність, то новий індекс 2-ої вершини дорівнює 7.(малюнок 2)

Аналогічну операцію пророблюємо з двома іншими сусідами 1-ої вершини — 3-ою та 6-ою.(малюнок 3).

Ну і оскільки вона вже розглянута, то її можна викреслити.

Слайд 9

Найближча з невідвіданих вершин — це вершина 2 з індексом 7, сусіди 1,3, 4.

Перший сусід вершини 2 — вершина 1. Проте оскільки вона вже відвідана, то з нею нічого не робимо. (малюнок 1)

Наступним сісідом є вершина 3 і якщо туди піти, то довжина шляху буде 7+10=17, проте з вершини 1 напряму довжина дорівнює 9, а тому не змінюємо.(малюнок 2)

Слайд 10

Наступним сусідом вершини 2 — вершина 4.

Тоді довжина шляху буде 7+15=22(малюнок 1) і оскільки 22 менше за нескінченність, то змінюємо.

Всі сусіди вершини 2 — розглянуті, заморожуємо стан.

Слайд 11

Тепер же найближча невідвідана вершина 3, бо має найменшу довжину шляху з вершини 1.

Її сусідами будуть вершини 1,2,4,6.

1 та 2 вже відвідані, тож не розглядаємо.

Найближчою нерозглянутою є вершина 6, тож 9+2=11. Проте попередній шлях дорівнював 14, якщо іти з вершини 1 і оскільки 11 менше за 14, то замінюємо.

Аналогічно з вершиною 4, бо 9+11=20 і це менше за 22, то замінюємо.

Слайд 12

Найближча невідвідана вершина 6, бо має найменшу довжину шляху з вершини 1.

Її сусідами будуть вершини 1,3,5, проте перші дві вже відвідані, тож розглянемо лише пʼяту і довжина такого шляху буде 11+9=20.

Слайд 13

Наступна вершина, що буде розглянута — 4, бо при рівних довжинах відбувається порядок.

Сусідами такої вершини будуть 2,3,5. Перші дві вже відвідані, тоє залишається лише пʼята. 20+6=26, що більше за 20, що вже є, тому пропускаємо.

Слайд 14

Останньою невідвіданою вершиною буде 5-та, її сусідами будуть відвідані вершини 6 та 4, а значить алгоритм завершується.

Алгоритм закінчує роботу, коли усі вершини відвідані.

Результат роботи алгоритму видно на останньому малюнку: найкоротший шлях від вершини 1 до 2-ї становить 7, до 3-ї - 9, до 4-ї - 20, до 5-ї - 20, до 6-ї - 11.

Якщо в якийсь момент усі невідвідані вершини позначені нескінченністю, то це означає, що до цих вершин не можна добратися (тобто граф незв'язний). Тоді алгоритм можна завершити достроково.

Тепер, коли ми розібралися з принципом його роботи, то пропоную перейти до реалізації.

Слайд 15

V — множина вершин графа

E — множина ребер графа

w[ij] — довжина ребра ij

а — вершина з якої шукаються довжини

U — множина відвіданих вершин

d[u] — по закінченню роботи алгоритма дорівнює довжині найкоротшого шляху з вершини а до u.

p[u] — по закінченню роботи алгоритму містить найкоротший шлях з а в u.

v — поточна вершина, що оброблюється.

Слайд 16

Присвоїмо d [a]←0, p [a]←0.

Для всіх u ∈ V відмінних від a присвоїмо d[u]←∞.

Поки ∃v∉U. Нехай v∉U — вершина з мінімальним d[v] занесемо v в U.

Для всіх u∉U таких, що vu∈E

Якщо d[u]>d[v]+w[vu], то

замінимо d[u]←d[v]+w[vu] та p[u]←(p[v],u)

Слайд 17

Реалізація на пітон

Слайд 18

Реалізація на С#