Починаємо розбір ось цієї формули: $W_k[i,j] = W_{k-1}[i,j] \vee (W_{k-1}[i,k] \& W_{k-1}[k,j])$.

1) Якщо $W_{k-1}[i,k] = 0$, тобто вершина k з вершини i не є «досяжною». То $W_k[i,j] = W_{k-1}[i,j] \vee (W_{k-1}[i,k] \& W_{k-1}[k,j])$ перетворюється у $W_k[i,j] = W_{k-1}[i,j]$. Тобто для заданого *i*, для якого $W_{k-1}[i,k] = 0$ і для всіх j (це стовпчик) ми просто стовпчик матриці переписали у нову матрицю. Так?

2) Якщо $W_k[i,k] = 1$, то там формула буде такою: $W_k[i,j] = W_{k-1}[i,j] \vee (W_{k-1}[i,k] \& W_k[k,j])$, – тобто це буде елемент з попередньої матриці або (ось це $W_k[i,k] = 1$, тому його можна опустити… 1*на що завгодно=що завгодно): $W_k[i,j] = W_{k-1}[i,j] \vee W_k[k,j]$. І, якщо це розглядати як досяжність, то ми одержимо ось тут: $W_k[i,j]$, – що елемент j буде досяжний з *i*; або, якщо він був досяжний раніше, або, якщо він досяжний через вершину $W_k[k,j]$. Тобто тут: $W_k[i,k] = 1$ – маємо маршрут з *i* до k, а тут: $W_k[k,j]$ – маємо або не маємо маршруту з k до j.

Чому за цим алгоритмом швидше?

Скільки потрібно виконати операцій, щоб виконати піднесення матриці до степеня? Множення 2 квадратних матриць… скільки операцій?

Ви повинні для обчислення одного елементу перемножити рядок на стовпчик – це буде n-множників.

Скільки у вас елементів? n^2. Тому обчислення добутку двох матриць – це буде n^3-операцій.

У вас матриць від нульової – до $(n-1)$-ої. Кожна матриця може обчислюватися шляхом множення від попереднього. Але їх n, тому загальна кількість операцій, які ви виконаєте буде n^4.

А тут (за алгоритмом) – n^3. У вас потрійний *for*. Кожний *for* робить n-операцій. Тобто $n \cdot n \cdot n$-операцій за *for*.

Чи є швидший алгоритм? Поки-що ні. Алгоритм Уоршелла прискорюється за рахунок реалізації. Тобто це все рівно бітові операції – їх можна пачками – регістрами виконувати… тобто там лінійне прискорення.

Так ось.

Теорема. Матриця W_k – це буде матриця досяжності графу G, якщо обмежити всі проміжні вершини на шляхах лише вершинами від першої – до *k*-тої. І не чіпаючи всі інші вершини. (*ми обмежуємо всі проміжні вершини на маршрутах лише вершинами з множини* W_1, W_2, …, W_k*; і, відповідно, коли ми дійдемо до матриці* W_n*, то там будуть всі можливі проміжні вершини, тобто це будуть всі можливі маршрути, які ми розглядаємо. Тому це буде матриця досяжності.*)

Так ось. Доведення цієї теореми.

Матриця W_0 – це фактично матриця суміжності: $W_0 = A_G$ – і вона описує нам наявність маршрутів без проміжних вершин, тобто: початок, кінець, 1 ребро (жодних проміжних вершин).

Тобто я можу сказати, що це: $W_0 = A_G$ – є досяжність без проміжних вершин.

Матриця W_1 *(знову дивіться на 5-ий рядок алгоритму…* $k = 1$*: і ми перевіряємо чи можемо ми з вершини* v_i *дійти до вершини* v_1 *(*$W_k[i,j]$*)*; а потім – чи можемо ми з вершини v_1 *дійти до вершини* v_j. *Тобто чи можуть бути вершини* v_i *та* v_j *чи просто поєднані ребром (*$W_{k-1}[i,j]$*), чи поєднані ланцюгом через вершину* v_1 *(*$W_k[k,j]$*).*

Так? Тоді давайте мені якусь емоційну реакцію… ☺).

Тобто матриця W_1 – розглядає всі маршрути між парами вершин, які проходять через вершину v_1 та всі попередньо розглянуті вершини. Але попередньо розглянутих вершин у нас жодної. Тому це буде просто досяжність через v_1.

Матриця W_2. Знов таки: ми розглядаємо досяжність через вершину v_2. Тобто чи існує маршрут від початку до v_2 і від v_2 – до кінця. А також через всі попередні. Але попередні – це вершина v_1, досяжність якої ми вже знаємо, бо вже розглянули. *Так?*

І, відповідно, матриця W_3 – це досяжність через v_3 та всі попередньо розглянуті, а саме: v_2 та v_1.

Матриця W_4 – це досяжність через v_4 та всі попередньо розглянуті, а саме: v_3, v_2 та v_1.

І на кожному кроці із ростом k (п. 2 алгоритму) у мене це все (п. 5 алгоритму) змінюється, змінюється, змінюється… я розглядаю все більше, більше, більше досяжності. Тому ставлю **…**

Кінець доведення.

Так ось… чому це швидше за піднесення матриці у степінь? Чому це ідеологічно швидше? Цей алгоритм побудований на так званому принципі «динамічного програмування». *Хто чув це словосполучення? Дехто чув. Так?*

Значить, що таке «динамічне програмування»?

У вас є задача. Для обчислення цієї задачі потрібно обчислити певну кількість підзадач. Для обчислення цих під задач потрібно обчислити ще певну кількість підзадач – і ви можете опускатися на найнижчий рівень – на задачі, які не поділяються, які потрібно розв'язувати загалом. Потім ви з цих задач будуєте відповідно розв'язок вашої великої задачі. Але під час цієї декомпозиції у вас можуть виникати однакові типи задач. І ви їх будете розв'язувати, хоча ви їх уже розв'язали. Ви будете ще… і ще … і ще розв'язувати. Так ось, динамічне програмування (тут «програмування» – це не те, що ви на компіляторах пишете, це «програмування» в сенсі – процедура, обчислення) полягає в тому, що ми усі під задачі, які у нас виникають, відсортовуємо таким чином, щоб для розв'язання поточної задачі нам потрібні були розв'язки лише попередніх задач. Це буде таке собі «топологічне сортування часткового порядку»: ви вишиковуєте ваші задачі в рядок – і починаєте розв'язувати від першої – до останньої; і ви кожну під задачу будете розв'язувати один раз і ніколи не будете повертатися, тобто у вас усе, що необхідне для обчислення уже буде на поточний момент.

Ось так, коли ви, наприклад, обчислите якийсь там степінь A^4: ми її обчислювали як $A^3 \cdot A$ - досяжність від якоїсь вершини, довжини 3, а потім ще окремо ребро (4-те) – до кінцевої вершини. Так ось: у вас в середині A^3 уже міститься «куча» уже повторюваної інформації. А потім, коли ви її множите на А, ця інформація ще дублюється.

Значить Уоршелл видумав, як ось це дублювання прибрати. Ось саме цим шляхом, коли ми не розглядаємо усі можливі маршрути між i-ою та j-ою вершинами. Ми розглядаємо спочатку маршрути, які не містять проміжних вершин, потім – які містять 1 проміжну вершину: і це саме вершина v_1. Потім маршрути, які проходять через вершину v_2. Відповідно усі маршрути у нас структуруються за наявністю проміжних вершин. Кожен маршрут ви будете розглядати лише 1 раз на кожному кроці. І для того, щоб побудувати маршрути, які проходять через v_2 та v_1 - вам потрібно маршрути, які проходять через v_1 і все. Ви їх розглянули на попередньому кроці. І так на кожному кроці ви додаєте 1 вершину і розглядаєте можливі маршрути. Додаєте ще вершину – розглядаєте ще маршрути… наприкінці виявляється, що ви розглянули усі можливі маршрути.

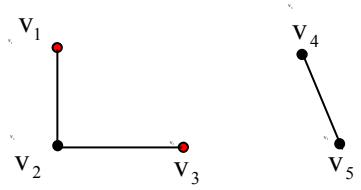
Зрозуміло?

Алгоритм Уоршелла узагальнюється. Він розв'язує з десяток різних задач. Якщо поміняти 5-ий рядок на якийсь інший.

Існують інші алгоритми на основі динамічного програмування.

А поки що приклад.

Візьмемо ось такий граф (рис. 1)



Граф маленький, тому його матрицю досяжності можна одразу намалювати. Це дуже добре, адже ми одразу знаємо, що ми повинні одержати наприкінці.
Треба побудувати матрицю суміжності. У нас виходить 5×5 -матриця.

$$W_0 = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{*}_{*}$$

у нас вершина v_1 поєднана лише з v_2. Тому 0,1,0,0,0.

Аналогічно інші вершини. Ось така у нас матриця суміжності W_0.

Далі починаємо обчислення.

Для того, щоб обчислити матрицю W_1 треба взяти 1-ий рядок і 1-ий стовпчик цієї матриці.

Пробігтися по 1-му рядку – подивитися: **якщо у нас нуль (0)**, то ми цей стовпчик просто переписуємо у нову матрицю W_1 (це випадок **1) Якщо** $W_{k-1}[i,k]=0$).

Якщо у нас один (1), то треба до цього стовпчика (другого) заводити новий стовпчик – додати за логічним «або» («v») – і результат записати у нову матрицю.

Тому матриця W_1: перший елемент першого рядка 0, тому 1-ий стовпчик просто переписуємо. 2-ий елемент 1-го рядка 1, тому ми ці (1-ий та 2-ий) два стовпчики потрібно логічно додати і записати те, що вийде. Так? Далі – 0, 0, 0, – тому ці (3,4,5-ий) стовпчики просто переписуються. *Зрозуміло*?

$$W_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{*}_{*}$$

Для обчислення матриці W_2 потрібно в W_1 взяти 2-ий рядок та 2-ий стовпчик. Пробігтися по 2-му рядку. Якщо 1 – проводити, записати; якщо 0 – просто переписати.

Дивимося: тут 1, тому потрібно заводити ці (1,2-ий) 2-ва стовпчики. Далі 1 (якщо заводити стовпчик сам із собою, то ви одержите те ж саме, що і було). Далі знову 1, тому знову заводимо 2-ва стовпчики (2,3-ій). Далі 0, 0, – тому просто переписуємо. *Зрозуміло*?

$$W_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{*}_{*}{}^{*}_{*} = W_3 .$$

Далі, щоб обчислити матрицю W_3 потрібно в W_2 взяти 3-ій рядок і 3-ій стовпчик. Пробігтися по 3-ому рядку: заводити і записати. *Процедура вже зрозуміла, здається. Так?*

Але хочу зауважити: 1, 2, 3-ій елементи «1», тому 1,2,3-ій стовпчики однакові. Далі стоять «0», тому у 4,5-му стовпчиках нічого не зміниться. Тому W_3 виглядає так само, як і W_2. *Зрозуміло?*

Тепер переходимо до матриці W_4. Перші 4-ри стовпчики просто переписуємо, бо там стоять «0». В 5-ий елемент 4-го рядка «1»

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

*

*

І, нарешті, остання – 5-та вершина: 5-ий рядок, 5-ий стовпчик. Перших 3-ри стовпчики переписуємо. 4-ий заводимо. 5-ий переписуємо без змін.

$$W_6 = \left[\begin{array}{ccc:cc} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right] = D_G$$ – Ось ця матриця буде матрицею досяжності графу G.

І ми бачимо, що вона має блокову структуру. Тобто у мене є сукупність вершин, які досяжні в нас в одне; іншу сукупність вершин, які досяжні в інше, а між ними вони не досяжні. Що це в нас є? класи еквівалентності за відношенням досяжності. Ось це (більший квадрат з «1») в нас один клас еквівалентності, ось це (меншийквадрат з «1») – інший клас еквівалентності. Наступного разу ми дізнаємось, що звуться вони «компонентами зв'язності».

Зараз занотуйте.

Для довільного графу завжди можна впорядкувати вершини таким чином, що його матриця досяжності буде мати блокову структуру. *(розказати про «блокові матриці»)* **Причому на діагоналі будуть стояти квадратні блоки з одних одиниць. Всі інші елементи будуть дорівнювати нулю.**

А взагалі: що таке «блокові матриці»? Зараз поясню… це просто…

Записали?

Що таке «блокова матриця»? Берете матрицю, а замість елементів вставляєте матриці.

Зокрема, ось ця матриця (D_G) має форму, наприклад:

$$D_G = \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix}$$, де матриця Y_1 – це буде 3×3-матриця з одних одиниць: $Y_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ – і вона буде квадратною. І матриця Y_2 це буде 2×2-матриця з одних одиниць: $Y_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ – і вона буде квадратною, а нулі – це «0». *Так?*

Значить, кожен блок описує окремий клас еквівалентності за відношенням досяжності.

Ну і, відповідно, чому мені потрібна «одинична діагональ»? Бо, якщо у вас є ізольована вершина, то це окремий клас еквівалентності. Якщо там не буде стояти «1», – там не буде блоку розміром 1×1 з одиничок.

Тема наступної лекції: Зв'язність графів.

Дякую за увагу. До зустрічі.