Кажемо, що вершина v_i досяжна з вершиною v_j, якщо в мене існує маршрут, який починається у v_j і закінчується – у v_i: $v_j \to .... \to v_i$. Тобто досяжність – це просто можливість почати в одній вершині і, проходячи ребрами графу, завершити шлях у іншій вершині.

Матриця досяжності графу G: D_G – це квадратна $n \times n$-матриця, елементи якої є нулями (0) та одиницями (1): $D_G = \left| d_{ij} \right|$. 0 – якщо вершини не досяжні; 1 – якщо вони досяжні.

Знову скористаюся дужками Айверсона (бо так коротше): $\deg = \left[v_j \textit{ досяжна з } v_i \right]$. *Що таке є «матриця досяжності» з алгебраїчної точки зору?* Оскільки матриця суміжність, то вона визначає певне бінарне відношення на множинах – відношення сусідства. І в матриці суміжності стоять 1 на парах з'єднаних вершин, 0 – на парах не з'єднаних вершин.

Відповідно «матриця досяжності» – це транзитивне замикання цього відношення.

Зауваження. Якщо у вас матриця суміжності (A_G) визначає відношення сусідства на вершинах, тоді матриця досяжності (D_G) є транзитивним замиканням цього відношення.

Чому це транзитивне замикання?

Якщо у вас є ребро з a в b і ребро з b в c, то за транзитивним замиканням у вас повинно бути ребро (a,c). Так? Ми не малюємо ребро, ми просто кажемо, що існує такий маршрут. Потім, якщо з c є маршрут у d, то існує маршрут з a в d. І так ми розгортаємо, розгортаємо, розгортаємо… поки можемо.

Для неорієнтованих графів у нас відношення сусідства є симетричним. Потім ми робимо транзитивне замикання – одержимо транзитивність.

Так ось. Якщо у вас – вже це псевдограф, де на кожній вершині є петля, або це граф, у якому немає ізольованих вершин, – то в нас ось це транзитивне замикання створює відношення еквівалентності. Воно розбиває вершину на підмножини, всередині якої кожна вершина досяжна з кожною іншою.

А коли ми розглядаємо графи з ізольованими вершинами, то там дуже незручно. Там є підмножина вершин, які досяжні і є ось ці окремі точки, які не є досяжними. Так іноді для того, щоб уніфікувати підхід просто за визначенням кажуть, що «кожна вершина досяжна сама із себе, шляхом довжини 0 (*ми стоїмо на місці – оп – ми досягли*)». І тоді D_G буде не просто транзитивним, а буде рефлексивно-транзитивним замиканням. Бо навіть, якщо в мене немає петлі з a в a *(a – ізольована вершина)*, то вона є досяжною. Там все одно з'являється 1 на діагоналі. Але це не завжди. Тому є пишу так:

(чому, якщо ізольована, то не буде еквівалентності? Тому, що якщо подивитися на це відношення, то воно не буде рефлексивним; воно буде симетричним, транзитивним, але воно не буде рефлексивним, тому що не має там пар (a,a)*;*

А чому ми не можемо зробити замикання (a,a)*? Можна, але це не буде ось ця операція… її (ось цю рефлексивність) потрібно внести у визначення цієї операції;*

А в псевдо графах обов'язково існують петлі для всіх вершин? Ні, але про це було сказано раніше. Так, що в псевдографі петля є на кожній вершині)

Тобто часто для уніфікації вважають, що кожна вершина є досяжною сама із себе, тоді D_G – це завжди буде відношення еквівалентності. Це відношення еквівалентності ми його називаємо «відношення досяжності».

Приклади

Є у вас такий граф (рис. 1).

Тоді виділені вершини є досяжними одна з одною, тому що Існує маршрут між ними.

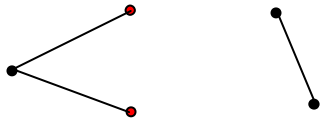
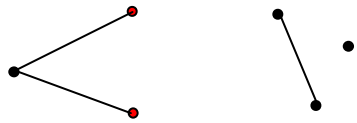


Рис. 1

Ці дві вершини (чорна вершина навпроти червоної вершини) не є досяжними, тому що не існує маршруту між ними.

Якщо у мене ще є ізольована вершина (рис. 2), то вона може вважатися досяжною сама із себе, а може не вважатися – в залежності від контексту.



І знов таки, це треба дивитися яке саме визначення використовується.

Рис. 2

Теорема. Беремо матрицю суміжності графу G: нехай $A = A_G$, – і беремо суму всіх її степенів до $(n-1)$-го.: $C = A^0 + A^1 + ... + A^{n-1} = \|C_{ij}\|$. (*Що таке матриця A^0? З точки зору лінійної алгебри? Це одинична матриця. Тобто це матриця, яка створює нам рефлексивність*). І розглянемо матрицю: $D = \|d_{ij}\|$, де $d_{ij} = \lfloor c_{ij} = 0 \rfloor$, – матриця D, яка формується за таким принципом: якщо у вас елемент $c_{ij} = 0$, то і $d_{ij} = 0$. А, якщо $c_{ij} \neq 0$, то ставимо 1.

Тоді ця матриця $D \equiv D_G$ – це і буде матриця досяжності графа G.

(Ще раз: беремо обчислюємо степені матриці суміжності до $(n-1)$-го. Додаємо. Дивимося що одержали: якщо одержали не 0 – ставимо 1, якщо одержали 0, то ставимо – 0. Те, що вийшло – це буде матриця нулів та одиниць. І теорема стверджує, що це і буде матриця досяжності графу G. Тобто вона в точності опише, які вершини є досяжні, які ні. Між якими існує маршрут, між якими не існує). Зрозуміло?

Доведення

Доведення насправді досить просте, хоча виглядає дуже страшно.

Якщо у вас вершини досяжні, то між ними існує маршрут. Якщо між двома вершинами існує маршрут, то між ними існує простий ланцюг (ми це доводили). А скільки може бути вершин у простому ланцюгу? (у вас у простому ланцюгу вершини не повторюються). У графі у вас n-вершин. Перша і остання можуть повторюватися, тому всередині буде не більше, ніж $(n-1)$-на. Тому довжина простого ланцюга, який поєднує дві вершини не може перевищувати n (строго менше, ніж n). Тому не більш, ніж $(n-1)$. Тому ми просто обчислюємо кількість всіх маршрутів довжини 1, довжини 2, довжини 3, …, довжини $(n-1)$, – додаємо; і окремо ще обчислюємо нульовий степінь – це досяжність вершини самої з себе – для ізольованих вершин. І, якщо ми одержали не 0, то у вас існує якийсь маршрут, довжини не більш, ніж $(n-1)$. Тому ці вершини є поєднані. *Так? Тепер пишемо.*

Якщо вершина v_i досяжна з v_j, то $\exists$ маршрут між цими вершинами, а тому $\exists$ простий ланцюг між цими вершинами.

Довжина вашого незамкненого простого ланцюга $\leq n-1$, якщо у вас перша і остання вершини не співпадають. Якщо співпадають, то це окремий випадок: ми одразу фіксуємо, що ізольовані вершини не досяжні.

$\leq n-1$, тому що всі вершини повинні бути різними. Там максимум n вершин, тому найбільша кількість ребер – $(n-1)$. Звідси випливає твердження теореми (я його навіть писати не хочу… всі розуміють?).

Доведено

Приклад: граф відношення дружби на студентах потоку ФІКТ. 150 вершин… ☺

Це не самий ефективний спосіб обчислення матриці досяжності. Чому? Взагалі кажучи ось тут: $C = A^0 + A^1 + ... + A^{n-1} = \|C_{ij}\|$ – ми обчислимо загальну кількість маршрутів, не більш, ніж заданої довжини. Але мене не цікавить загальна кількість маршрутів. Мене цікавить наявність хоча б одного маршруту. Якщо такий маршрут $\exists$, то можна далі вже

нічого не обчислювати. Тому я можу звести ось це обчислення матриці D: $\mathrm{D} = \left\| \mathrm{d_{ij}} \right\|$, де $d_{ij} = \left[c_{ij} = 0 \right]$, – з арифметичних операцій у логічну.

Розглянемо дві $\mathrm{n} \times \mathrm{n}$ *-матриці*. Нехай A, B – $\mathrm{n} \times \mathrm{n}$ -матриці, яких елементи є 0 ата 1. І введемо альтернативну суму та альтернативне множення цих матриць.

Альтернативна сума для нас – це логічне «або» і ми його вводимо поелементно: $\mathrm{A} \vee B = \left\| a_{ij} \vee b_{ij} \right\|$.

Ось альтернативне множення – це буде якась матриця С: $\mathrm{A} \,\&\, \mathrm{B} = \mathrm{C} = \left\| c_{ij} \right\|$. Тут треба формулу для добутку матриць переписати у логічних операціях: $c_{\mathrm{ij}} = \bigvee_{t=1}^{n} \left(a_{it} \,\&\, b_{\mathrm{tj}} \right)$ – (*це буде сума по всіх* t *від 1 – до* n a_{it} *на* b_{tt}*).*

Дивіться, коли ми, скажімо, обчислюємо степінь $\mathrm{A}^{\mathrm{n-1}}$ – це було $\mathrm{A}^{\mathrm{n-2}} \cdot A$. Ми б обчислювали кількість всіх шляхів до однієї вершини – і множили ні кількість всіх шляхів до кінцевої вершини. Але, якщо мене не цікавить кількість всіх шляхів. Мене цікавить лише досяжність. То дві вершини є досяжними, якщо між ними існує хоча б 1 вершина, що перша досяжна до цієї вершини – і ця досяжна до кінцевої. $c_{\mathrm{ij}} = \bigvee_{t=1}^{n} \left(a_{it} \,\&\, b_{\mathrm{tj}} \right)$ – *i* досяжна з t, t досяжна з j. Ставимо логічне "&", щоб це були одночасні умови. А потім я просто перебираю всі можливі вершини, бо мене цікавить наявність хоча б однієї такої вершини.

Тобто маємо

Теорема. $\mathrm{D_G} = A^0 \vee A^1 \vee \ldots \vee A^{n-1}$.

Чи можна просто обчислити матрицю досяжності, якщо просто у формулі: $C = A^0 + A^1 + \ldots + A^{n-1} = \left\| C_{ij} \right\|$ – замінити всі арифметичні операції на логічні.

Що буде простіше?

Візьмемо штат Windows і процесор. Операція додавання регістрів займає, умовно, 10 тактів. Операція логічного «або» – 1 такт. Коли ви підносите матрицю до степеня – там числа множаться, множаться, множаться… – вони все ростуть, ростуть, ростуть… І самі арифметичні операції – із ростом довжини числа – стають все складнішими і складнішими. А тут: $\mathrm{D_G} = A^0 \vee A^1 \vee \ldots \vee A^{n-1}$ – ніякого росту не відбувається. Воно все, знов таки, за 1 такт.

Більш того в регістрах в нас 32 біта або 64 біти. Можна ці операції робити пачками. Тобто паралельно 64 логічні «або» на біта.

Так ось. Формула: $C = A^0 + A^1 + \ldots + A^{n-1} = \left\| C_{ij} \right\|$ – це ще не самий оптимальний шлях обчислення матриці досяжності. Можна ще швидше.

Алгоритм Уоршелла. Він буде виглядати незвично. Але на виході ви одержите матрицю досяжності. І я спробую вам пояснити чому.

Отже, на вході в алгоритм у нас повинен бути граф G, представлений його матрицею суміжності ($\mathrm{A_G}$). *Тобто граф для цього алгоритму представляється у вигляді матриці суміжності.*

На виході ми маємо послідовність матриць: $\mathrm{W}_0, W_1, \ldots, W_n$, де n – це кількість вершин у графі і, відповідно, розміри матриці суміжності. Де матриця $\mathrm{W}_0 = A_G$ (матриця W_0 – це буде матриця суміжності графа), а матриця $W_n = D_G$ (матриця W_n – це буде матриця досяжності графа).

Алгоритм.

1. Присвоїмо матриці W_0 значення A_G: $\mathrm{W}_0 = A_G$.
2. Пробігаємо всі індекси матриць W: для всіх k від 1 до n.

3. Потім пробігаємо всі індекси елементів матриць W : для всіх i від 1 до n.
4. для всіх j від 1 до n.
5. І виконуємо таку операцію: $W_k[i,j]=$ **(*я буду позначати елементи в програмістському стилі; це* i,j*-ий елемент матриці* W_k**). Треба взяти:** $W_k[i,j]=W_{k-1}[i,j]\vee(W_{k-1}[i,k]\&W_{k-1}[k,j])$ *– основний рядок цього алгоритму.*
Тобто 2-4 – потрійний цикл *for*. Його потрібно пробігти. Обчислити 5. І, якщо мені потрібна рефлексивність; а в графі можуть бути ізольовані вершини, то наприкінці просто проставляємо на діагоналі одиниці:
6. $W_n = W_n \vee E$, де Е-одинична матриця.
Добре. Давайте розбиратися що тут відбувається.
Це весь алгоритм.
Він справді виглядає дивно (як «чорна магія»), але це тому, що ви не знаєте алгоритм Беллмана-Форда, який дізнаєтеся потім
Починаємо розбір ось цієї формули: $W_k[i,j]=W_{k-1}[i,j]\vee(W_{k-1}[i,k]\&W_{k-1}[k,j])$.
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?
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»?