Лекція 14. Відстані на графах.

ПЛАН

1. Графи з числовими характеристиками ребер (дуг).

2. Найкоротші шляхи.

3. Алгоритм Дейкстри визначення відстані між вершинами на графі із довільними довжинами ребер.

4. Обхід графів

14.1. Графи з числовими характеристиками ребер (дуг).

У реальних задачах на графах часто потрібно брати до уваги додаткову інформацію — фактичну віддаль між окремими пунктами, вартість проїзду, час проїзду тощо. Для цього використовують поняття зваженого графа.

Зваженим називають простий граф *G = (V, E)*, кожному ребру $e \in E$ якого приписано дійсне число *w*(e). Це число називають *вагою* ребра *e*.

Аналогічно означають зважений орієнтований граф.

Довжиною шляху в зваженому графі називають суму ваг ребер (дуг), які утворюють цей шлях. Якщо граф не зважений, то вагу кожного ребра (кожної дуги) вважають рівною 1 й отримують раніше введене поняття довжини шляху як кількості ребер (дуг) у ньому.

14.2. Найкоротші шляхи.

Задача про найкоротший шлях полягає в знаходженні найкоротшого шляху від заданої початкової вершини $v_0 \in V$ до заданої вершини $u \in V$.

Наступні дві задачі — безпосередні узагальнення сформульованої задачі про найкоротший шлях.

1. Для заданої початкової вершини v_0 знайти найкоротші шляхи від v_0 до всіх інших вершин.

2. Знайти найкоротші шляхи між усіма парами вершин.

Виявляється, що майже всі методи розв'язання задачі про найкоротший шлях від заданої початкової вершини v_0 до заданої вершини u також дають

змогу знайти й найкоротші шляхи від вершини v_0 до всіх інших вершин графа. Отже, за їх допомогою можна розв'язати задачу 1 із невеликими додатковими обчислювальними витратами. З іншого боку, задачу 2 можна розв'язати або n разів застосувавши алгоритм задачі 1 із різними початковими вершинами, або один раз застосувавши спеціальний алгоритм.

Розглянемо два алгоритми: перший алгоритм призначений для розв'язування

задачі 1, другий — для задачі 2.

14.3. Алгоритм Дейкстри визначення відстані між вершинами на графі із довільними довжинами ребер.

Найефективніший алгоритм визначення довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої запропонував 1959 р. датський математик Е. Дейкстра (E. Dijkstra). Цей алгоритм застосовний лише тоді, коли вага кожного ребра (дуги) додатна. Опишемо докладно цей алгоритм для орієнтованого графа.

Нехай $G = (V,E)$ — зважений орієнтований граф, $w(u,v)$ — вага дуги $(u,v) \in E$. Почавши з вершини v_0, знаходимо віддаль від v_0 до кожної із суміжних із нею вершин. Вибираємо вершину, віддаль від якої до вершини v_0 найменша; нехай це буде вершина v. Далі знаходимо віддалі від вершини v_0 до кожної вершини, суміжної з v вздовж шляху, який проходить через вершину v. Якщо для якоїсь із таких вершин ця віддаль менша від поточної, то заміняємо нею поточну віддаль. Знову вибираємо вершину, найближчу до v й не вибрану раніше; повторюємо процес.

Описаний процес зручно виконувати за допомогою присвоювання вершинам *міток*. Є мітки двох типів — тимчасові та постійні. Вершини з постійними мітками групують у множину М, яку називають множиною позначених вершин. Решта вершин має тимчасові мітки, і множину таких вершин позначимо як Т, $T = V \setminus M$. Позначатимемо мітку (тимчасову чи

постійну) вершини v як $l(v)$. Значення постійної мітки $l(v)$ дорівнює довжині найкоротшого шляху від вершини v_0 до вершини v, тимчасової — довжині найкоротшого шляху, який проходить лише через вершини з постійними мітками.

Фіксованою початковою вершиною вважаємо вершину v_0; довжину найкоротшого шляху шукаємо до вершини u (або до всіх вершин графа).

Тепер формально опишемо алгоритм Дейкстри.

Крок 1. Присвоювання початкових значень. Виконати $l(v_0)=0$ та вважати цю мітку постійною. Виконати $l(v)=\infty$ для всіх $v\neq v_0$ і вважати ці мітки тимчасовими. Виконати $x=v_0,\ M=\{v_0\}$.

Крок 2. Оновлення міток. Для кожної вершини $v\in\Gamma(x)\setminus M$ замінити мітки: $l(v)=\min\{l(v);l(v)+w(x,v)\}$, тобто оновлювати тимчасові мітки вершин, у які з вершини *x* іде дуга.

Крок 3. Перетворення мітки в постійну. Серед усіх вершин із тимчасовими мітками знайти вершину з мінімальною міткою, тобто знайти вершину v' з умови $l(v')=\min\{l(v)\},\ v\in T,$ де $T=V\setminus M$.

Крок 4. Вважати мітку вершини v' постійною й викопати $M=M\cup\{v'\},\ x=v'$ (вершину v' включено в множину M).

Крок 5. Закінчення. а) Для пошуку шляху від v_0 до u: якщо $x=u$, то $l(u)$ — довжина найкоротшого шляху від v_0 до u, зупинитись; якщо $x\neq u$, то перейти до кроку 2.

б) Для пошуку шляхів від v_0 до всіх інших вершин: якщо всі вершини

отримали постійні мітки (включені в множину M), то ці мітки дорівнюють довжинам найкоротших шляхів, зупинитись; якщо деякі вершини

мають тимчасові мітки, то перейти до кроку 2.

14.4. Обхід графів

Існує багато алгоритмів на графах, які ґрунтуються на систематичному переборі їх вершин або обході вершин, під час якого кожна вершина одержує унікальний порядковий номер. Алгоритми обходу вершин графа називають методами пошуку.

1. *Пошук углиб у простому зв'язному графі* .

Розглянемо метод пошуку в простому зв'язному графі, який являє собою одну з основних методик проектування алгоритмів, пов'язаних із графами. Цей метод називають пошуком углиб. Нехай *G= (V. E)* — простий зв'язний граф, усі вершини якого позначено попарно різними символами. У процесі пошуку вглиб вершинам графа *G* надають номери та певним способом позначають ребра. У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають *стеком*. Зі стеку можна вилучити тільки той елемент, котрий було додано до нього останнім: стек працює за принципом „останнім прийшов — першим вийшов". Інакше кажучи, додавання й вилучення елементів у стеку відбувається з одного кінця, який називають верхівкою стеку.

2. *Пошук ушир у простому зв'язному графі* .

У процесі пошуку вшир вершини графа проглядають в іншій послідовності, ніж у методі пошуку вглиб. Під час пошуку рухаються вшир, а не вглиб: спочатку проглядають усі сусідні вершини, після цього — сусіди сусідів і так далі. У ході реалізації алгоритму використовують структуру даних для збереження множин, яку називають *чергою*. Із черги можна вилучити тільки той елемент, який перебував у ній найдовше: працює принцип „першим прийшов — першим вийшов". Елемент включається у хвіст черги, а виключається з її голови. Пошук ушир, узагалі кажучи, відрізняється від пошуку вглиб заміною стеку на чергу. Після такої модифікації, що раніше відвідується вершина (включається в чергу), то

раніше вона використовується (і виключається з черги). Використання вершини полягає в перегляді одразу всіх іще не відвіданих її сусідів.

Література.

1. Бондаренко М.Ф. Комп’ютерна дискретна математика : Підручник / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: “Компанія СМІТ”, 2004. – 480 с.

4. Нікольский Ю.В. Дискретна математика: Підручник / Ю.В.Нікольский, В.В. Пасічник, Ю.М. Щербина. – Львів: “Магнолія—2006”, 2010. – 432 с.

5. Новиков Ф.А. Дискретная математика для программистов / Ф.А. Новиков. – СПб.: Питер, 2000. –304 с.