**Задача комівояжера та алгоритми її розв’зання. Жадібні алгоритми**

Для всіх розглянутих нами раніше задач на графах існують алгоритми розв’язання з поліноміальною складністю. Але для великої кількості оптимізаційних задач на графах до теперішнього часу не знайдено алгоритмів з поліноміальною складністю (ефективних алгоритмів). Є навіть вагомі доводи вважати, що таких алгоритмів не існує. Такі задачі прийнято називати складнорозв’язуваними. **Задача комівояжера** – одна з таких задач.

Задача комівояжера є однією з цікавих задач комбінаторики. Вона була поставлена у 1934 році і історія її розв’язання подібна до історії розв’язання Великої теореми Ферма пов’язана з іменами найкращих математиків. В області оптимизаційних дискретних задач задача комівояжера є своєрідним полігоном, на якому тестуються всі нові методи.

**Формулювання**. Є *р* міст, відстані між якими відомі. Комівояжер повинен вийти з деякого міста, відвідати всі інші міста по одному разу і повернутися назад. Треба знайти такий маршрут, при якому пройдена відстань буде мінімальною.

Відомо, що у 1859 р. У. Гамільтон придумав гру «Навколосвітня подорож», в якій треба було знайти маршрут подорожі, який проходить через всі міста (вершини графа) рівно по одному разу і початок маршруту співпадає з кінцем.

Такі маршрути в графі називаються гамільтоновими циклами. Ця гра очевидно лежить в основі задачі комівояжера, яка як тепер можна сказати зводиться до **знаходження найкоротшого гамільтонового циклу в повному графі.**

Один із можливих підходів до складнорозв’язуваних задач полягає в побудові алгоритмів поліноміальної складності для отримання «гарного», але можливо не оптимального розв’язку. Одразу виникає проблема оцінки похибки, тобто наскільки знайдений розв’язок відрізняється від оптимального. Вона часто є більш складною, ніж побудова алгоритму.

Один з найпростіших алгоритмів побудови маршруту комівояжера називається Найближчій сусід (Nearest vertex):

* Вибираємо будь-яку вершину в якості початкової і проголошуємо її останньою включеною в маршрут.
* Нехай v – остання включена в маршрут вершина. Серед всіх поки не включених в маршрут вершин вибираємо найближчу до v вершину w, включаємо її в маршрут і оголошуємо останньою включеною в маршрут вершиною. Якщо всі вершини включено в маршрут, то повертаємось в початкову вершину.

Складність цього алгоритму *О(п2)*. Цей алгоритм не завжди знаходить оптимальний розв’язок. Наступна теорема дає оцінку похибки.



Ще один алгоритм називається «Найближча вставка» (Nearest insert):

**Крок 1**. Вибираємо будь-яку вершину *v* і вважаємо її поточним маршрутом *Т*.

**Крок 2**. Якщо всі вершини графа входять в *Т*, то СТОП (*Т* – маршрут комівояжера).

**Крок 3**. Інакше, серед всіх вершин, які не належать поточному маршруту *Т*, знаходимо таку вершину *v*, для якої величина *d(v,T)* мінімальна. Нехай *w* – вершина з *Т*, для якої *d(v,T)= d(w,T),* і *u* – вершина, наступна за *w* в маршруті *Т.*

**Крок 4**. Додаємо вершину *v* в маршрут *Т*, вставляючи її між *w* і *u*. Переходимо до кроку 2.



**Поняття про жадібну стратегію в алгоритмах на графах**

Алгоритми розглянутого типу називаються **жадібними алгоритмами**.

Жадібний алгоритм шукає розв’язок задачі шляхом здійснення вибору, який є найкращим для кожного кроку алгоритма. Ця стратегія є евристичною, вона не завжди приводить до оптимального розв’язку, але часто знайдений розв’язок є «близьким» до оптимального.

**Стратегія жадібного вибору – глобальний оптимальний розв’язок можна отримати, здійснюючи локальний оптимальний (жадібний) вибір.**



**Теоретичні основи жадібних алгоритмів**

Жадібні алгоритми – це загальна назва підходу до розв’язання задач оптимізації.

Жадібні (градієнтні) алгоритми діють за принципом "максимальний виграш на кожному кроці". Така стратегія не завжди веде док кінцевого успіху - іноді вигідніше на певному кроці зробити не найкращий вибір,щоб у підсумку отримати оптимальний розв’язок. Але, тим не менше, існує велика кількість задач, для яких застосування жадібних алгоритмів виявляється виправданим.

**Питання: коли жадібний алгоритм дає оптимальний розв’язок?**

Відповідь пов’язана з поняттям матроїду – це сучасна комбінаторна структура, теорія якої швидко та інтенсивно розвивається.

Поняття матроїда було введено Уітні в середині 20 ст.

**Означення** Нехай Х – довільна скінченна множина (носій), *І* – сукупність її підмножин (), яка задовольняє умовам (аксіомам):

1. ,
2.  (аксіома спадковості),
3.  (аксіома заміни).

Тоді Х разом з *І* називається *матроїдом*, елементи з *І* називаються *незалежними множинами*, решта підмножин з Х – *залежними*. Якщо виконуються тільки аксіоми 1) та 2), то отримаємо *передматроїд*.

**Приклади матроїдів.**

1. *Векторний матроїд*

Носій матроїда - - векторний простір над полем , . (1,2,0),(2,3,0), (4,5,1) (4,5,1), (2,3,0)

Розглянемо *І* – сукупність всіх можливих лінійно незалежних систем векторів й вектора , який буде виконувати роль порожньої множини й забезпечить виконання аксіоми 1). Аксіома 2) теж виконується, так як «будь-яка підсистема лінійно незалежної системи векторів теж лінійно незалежна система векторів». Аксіома заміни теж виконується. Дійсно, нехай системи А та В векторів лінійно незалежні і в системі А векторів більше хоча б на 1 (звідси випливає, що В не є максимальною лінійно незалежною). Відомо, що будь-яку не максимальну лінійно незалежну систему можна доповнити ще хоча б одним вектором так, щоб властивість лінійної незалежності збереглась. Якщо би жоден з векторів системи А не підійшов, то це означало б, що система А лінійно залежна, що суперечить умові.

Таким чином, векторний простір  разом із сукупністю *І* всіх лінійно незалежних систем векторів в ньому є матроїдом. Крім назви *векторний матроїд,* для нього використовують терміни *лінійний матроїд, матричний матроїд.*

**Зауваження.** Можна розглядативекторний простір над будь-яким полем, в тому числі й над скінченим полем, наприклад, над полем .

1. *Графовій матроїд.*

Носій – множина Е всіх ребер довільного неорієнтованого графа. *І* – сукупність всіх можливих ациклічних підмножин множини E (наприклад, дерев, лісів). Аксіоми 1) і 2) виконуються. Для перевірки 3) аксіоми розглянемо ациклічні множини А та В, причому потужність першого більша за потужність другого. Розглянемо 2 графа  та  - перший містить тільки ребра з А, другий – тільки ребра з В. нехай спочатку граф  порожній *п*-вершинний, тобто має *п* компонент зв’язності. Будемо поступово додавати ребра з А. Після кожного кроку число компонент зв’язності зменшується на 1, а після останнього кроку їх число дорівнює К()=*п*-. Аналогічно,

К()=*п*-. Так як , то К. Значить, існує компонента зв’язності в , яка містить вершини не менше ніж 2 компонент зв’язності з . В ній є ребро е, яке ми й додамо до В, при цьому властивість ациклічності не порушується.

**Теорема** Графовий матроід ізоморфний векторному матроїду над полем .

**Їдея доведення.** Беремо матрицю інцидентності графа, розглядаємо її стовпці як вектори векторного простору над полем  з операцією додавання по модулю 2 й звичайним множенням на числа з поля. Тоді для ребер графа, які утворюють цикл, сума відповідних векторів (по модулю 2) дорівнює 0, тобто вони утворюють лінійно залежну систему. Якщо ж вектори відповідають ациклічним підмножинам ребер в графі, то їх сума по модулю 2 не дорівнює 0, а значить вони лінійно незалежні.

*3) Матроїд паросполучень*

*4) Різнокольоровий матроїд*

**Означення** Нехай А – довільна множина з Е. Будь-яку максимальну незалежну підмножину В, яка міститься в А, будемо називати *базою* множини А. Бази множини Е будемо називати *базами матроїда М.*

**Приклади.**

* Якщо А- підмножина векторного матроїда, то базою А є базис лінійної оболонки, натягнутої на А.
* Якщо А – підмножина графового матроїда, то базою А є остовний ліс для цієї множини.

**Означення.** *Зваженим матроїдом* називається матроїд  разом з деякою ваговою функцією . Якщо А – множина з Х, то число  називається *вагою множини* А.

Тоді виникають оптимізаційні задачі на матроїдах, наприклад, знайти базу мінімальної ваги. Вона розв’язується за допомогою жадібних алгоритмів (алгоритмів Радо-Едмондса):

1. Сортуємо Х по зростанню ваги.
2. В=

Вибираємо 

Якщо , то приєднуємо *х* до В.

**Приклади.** Алгоритм Краскала пошуку мінімального остовного дерева в графі, алгоритм пошуку досконалого паросполучення у дводольному графі.



