**ЗАДАЧА ПРО ПРИЗНАЧЕННЯ**

Задача про призначення — одна з фундаментальних задач комбінаторної оптимізації. Задача полягає в пошуку мінімальної суми дуг у зваженому орієнтованому дводольному графі.

***Якщо множина  вершин простого графа допускає розбиття на дві підмножини  та  Ø так, що не існує ребер, що поєднують вершини однієї і тієї ж підмножини, то граф називається дводольним або біграфом (рис. 6.5).***



Рисунок 6.5 – Дводольний граф

***Теорема 6.4 (критерій Кеніга дводольності графа). Зв'язний граф  є дводольним тоді й тільки тоді, коли він не має циклів непарної довжини.***

У найбільш загальній формі задача про призначення формулюється наступним чином:

**Маємо деяку кількість *робіт* та деяку кількість *виконавців*. Будь-який виконавець може бути призначений для виконання будь-якої (але тільки однієї) роботи, але з різними витратами. Треба розподілити роботи так, щоб виконати всі види робіт з мінімальними затратами.**

Якщо число робіт співпадає з числом виконувачів, то задача називається *лінійною задачею про призначення*. Зазвичай, якщо говорять про *задачу про призначення* без додаткових умов, мають на увазі саме *лінійну задачу про призначення*.

**Приклад**

**Припустимо, що таксомоторна компанія має три вільні машини (виконувачі), й три зазамовника (роботи), які бажають отримати таксі якнайшвидше. Фірма турбується про час доставки таксі до замовника, тому для кожної машини *вартість визначається* кількістю часу, протягом якого машина добереться до місця, визначеного замовником.**

Розв’язком задачі буде такий розподіл машин по замовникам, що сумарна вартість (сумарний час чекання) буде мінімальним.

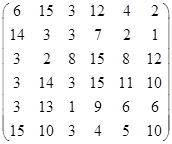
**Зауваження.** Задачу про призначення можна зробити більш гнучкою. У наведеному прикладі може виявитися не три, а чотири вільні таксі, а замовників як і раніше три. Можна призначити четвертого (фіктивного) замовника з нульовою вартістю, розподіл машини на фіктивного замовника означає— «нічого не роби».

Аналогічний прийом можна використовувати у випадку, коли число замовників перевищує число доступних машин, й інші варіанти, наприклад, поставити задачу *збільшення доходу*, а не *мінімізацію витрат*.

**Розв’язання задачі про призначення венгерським методом. Випадок задачі на максимум.**

Наприклад, «На роботу влаштовуються 6 кандидатів на 6 вакансій і вони отримали відповідні оцінки на співбесіді на кожну вакансію. Провести набір кандидатів так, щоб сумарна оцінка кандидатів була максимальною».

Будемо вважати, що дані занесено в матрицю. Треба вибрати по одній клітці в рядку і стовпчику так, щоб їх сума була *максимальною.*

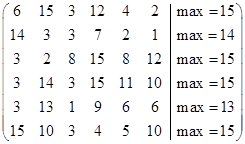


**Розв’язання:**

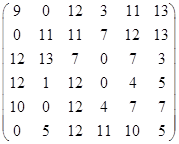
**Крок 1:**

**Зауваження:** перший крок буде тільки для задачі на максимум. Якщо треба розв’язати задачу на мінімум, то він пропускається.

Перетворимо матрицю, замінивши кожний елемент кожного рядка ріазицею максимального елемента цього рядка й самого елемента.

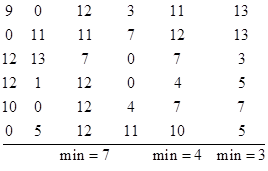


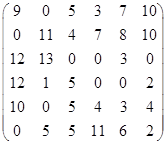
Отримаємо



**Крок 2.**

Треба отримати нулі в кожному рядку й в кожному стовпці. В третьому, п’ятому й шостому стовпцях нулів немає. Віднімемо з елементів цих столпців мінімальний елемент відповідного стовпця.



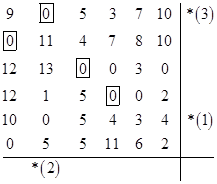


**Крок 3.**

Отримали матрицю, в якій в кожному рядку й кожному стовпці є нуль. Нам треба відмітити по одному нулю в кожному рядку й кожному стовпці. В цій матриці тільки перші чотири рядки й стовпці задовольняють вимозі. Відмітимо відповідні комірки рамкою.

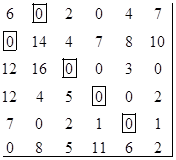
Відмітимо 5-й рядок як «незадоволений», в ньому ми такий нуль відмітити не змогли, й другий стовпчик, він містить нуль в п’ятому рядку. Але другий стовпчик також містить нуль у першому рядку. Відмітимо і його як «незадоволений». Перший рядок нулів більше не має, тобто процес відмічання «незадоволених» рядків завершено. Ми отримали ситуацію під назвою «вузьке місце».

В таблиці будемо відмічати «незадоволені» рязки й стовпці зірочками, а число поруч із зірочкою буде означати порядок відмічання (для кращого розуміння процесу) .



Виберемо мінімальний елемент в помічених (\* ) рядках поза рядків і стовпців з відміченими нулями. Це число 3 в п’ятому стовпці й п’ятому рядку. Віднімемо цей елемент з відмічених рядків й додамо до відмічених стовпців.

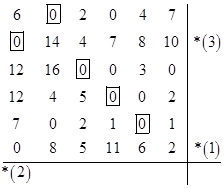




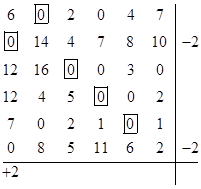
Зауважимо, що тепер можна відмітити нуль в п’ятому рядку й п’ятому стовпці.

**Крок 4.**

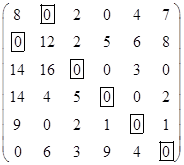
Ще немає нуля в 6-ому рядку. відмітимо його як незадоволений, він має нуль в першому стовпці, відмітимо його як незадоволений. Він, в свою чергу, містить нуль у другому рядку, відмітимо його, але в ньому нулів більше немає. Процес відмічання завершено.



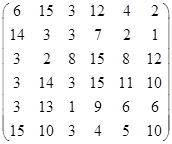
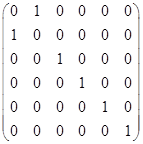
Виберемо мінімальний елемент у відмічених рядках поза відмічених стовпчиках. Це елемент 2 в шостому рядку і шостому стовпчику. Віднімемо двійку з другого та шостого рядків і додамо допершого стовпчика.



Замітимо, що тепер можна відмітити ще один нуль.



Отримали матрицю з шістьма нулями, по одному в кожному рядку й кожному стовпці. Отже, можна провести призначення (розподіл робіт тощо) за матрицею:



Вартість такого призначення дорівнює:

http://kontromat.ru/imag/vnazn/image026.png

**Задача о назначениях**

Рассматривается вычислительная система, состоящая из *n* вычислительных машин. Имеется  *m*  задач. Задана матрица  *T,*  определяющая время решения *i*-й задачи на *j*-й машине. Задачи решаются одновременно с некоторого момента *t0*. Найти такое распределение задач по вычислительным машинам, чтобы общее время решения всех задач было *минимальным* при условии, что на одной машине может решаться

только одна задача.

**Решение**.  
Исходная матрица имеет вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 5 | M | 2 | 2 |
| 7 | 4 | 2 | 3 | 1 |
| 9 | 3 | 5 | M | 2 |
| 7 | 2 | 6 | 7 | 8 |

**Замечание .** М - очень большое число, выражает, например, невозможность выполнения *і*-той работы *к*-тым работником.

1. Для устранения дисбаланса добавляем дополнительные строки. Проводим редукцию матрицы по строкам. В связи с этим во вновь полученной матрице в каждой строке будет как минимум один ноль.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 3 | M | 0 | 0 | **2** |
| 6 | 3 | 1 | 2 | 0 | **1** |
| 7 | 1 | 3 | M | 0 | **2** |
| 5 | 0 | 4 | 5 | 6 | **2** |
| 0 | 0 | 0 | 0 | 0 | **0** |

Затем такую же операцию редукции проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | M | 0 | 0 |
| 6 | 3 | 1 | 2 | 0 |
| 7 | 1 | 3 | M | 0 |
| 5 | 0 | 4 | 5 | 6 |
| 0 | 0 | 0 | 0 | 0 |
| **0** | **0** | **0** | **0** | **0** |

После вычитания минимальных элементов получаем полностью редуцированную матрицу.

2. Методом проб и ошибок проводим поиск допустимого решения, для которого все назначения имеют нулевую стоимость.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | M | **[0]** | [-0-] |
| 6 | 3 | 1 | 2 | [-0-] |
| 7 | 1 | 3 | M | [-0-] |
| 5 | **[0]** | 4 | 5 | 6 |
| [-0-] | [-0-] | [-0-] | [-0-] | **[0]** |

Поскольку расположение нулевых элементов в матрице не позволяет образовать систему из 5-ти независимых нулей (в матрице их только 3), то **решение недопустимое.**

3. Проводим модификацию матрицы. Вычеркиваем строки и столбцы с возможно большим количеством нулевых элементов: строку 5, столбец 5, строку 1, столбец 2.

Получаем сокращенную матрицу (элементы выделены):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | M | 0 | 0 |
| **6** | 3 | **1** | **2** | 0 |
| **7** | 1 | **3** | **M** | 0 |
| **5** | 0 | **4** | **5** | 6 |
| 0 | 0 | 0 | 0 | 0 |

Минимальный элемент сокращенной матрицы есть 1, вычитаем 1 из всех элементов сокращенной матрицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | M | 0 | 0 |
| **5** | 3 | **0** | **1** | 0 |
| **6** | 1 | **2** | **M** | 0 |
| **4** | 0 | **3** | **4** | 6 |
| 0 | 0 | 0 | 0 | 0 |

Затем прибавляем минимальный элемент к элементам, расположенным на пересечениях вычеркнутых строк и столбцов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | **4** | M | 0 | **1** |
| 5 | 3 | 0 | 1 | 0 |
| 6 | 1 | 2 | M | 0 |
| 4 | 0 | 3 | 4 | 6 |
| 0 | **1** | 0 | 0 | **1** |

**второй цикл алгоритма:**

1. Проводим редукцию матрицы по строкам. В связи с этим во вновь полученной матрице в каждой строке будет как минимум один ноль.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | M | 0 | 1 | **0** |
| 5 | 3 | 0 | 1 | 0 | **0** |
| 6 | 1 | 2 | M | 0 | **0** |
| 4 | 0 | 3 | 4 | 6 | **0** |
| 0 | 1 | 0 | 0 | 1 | **0** |

Затем такую же операцию редукции проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | M | 0 | 1 |
| 5 | 3 | 0 | 1 | 0 |
| 6 | 1 | 2 | M | 0 |
| 4 | 0 | 3 | 4 | 6 |
| 0 | 1 | 0 | 0 | 1 |
| **0** | **0** | **0** | **0** | **0** |

После вычитания минимальных элементов получаем полностью редуцированную матрицу.

2. Методом проб и ошибок проводим поиск ***допустимого решения***, для которого все назначения имеют нулевую стоимость.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | M | **[0]** | 1 |
| 5 | 3 | **[0]** | 1 | [-0-] |
| 6 | 1 | 2 | M | **[0]** |
| 4 | **[0]** | 3 | 4 | 6 |
| **[0]** | 1 | [-0-] | [-0-] | 1 |

***исходная матрица***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 5 | M | 2 | 2 |
| 7 | 4 | 2 | 3 | 1 |
| 9 | 3 | 5 | M | 2 |
| 7 | 2 | 6 | 7 | 8 |

C min = 2 + 2 + 2 + 2 + 0 = 8