Hungarian Algorithm

Content

- Introduction
- Example
- Practice

Introduction

- **Hungarian Algorithm**
  - The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal–dual methods.
  - It was developed and published in 1955 by Harold Kuhn, who gave the name "Hungarian method" because the algorithm was largely based on the earlier works of two Hungarian mathematicians.

Example

- Given n workers and tasks, and an n × n matrix containing the cost of assigning each worker to a task, where rows represent workers and columns represent tasks. The purpose is to find the cost minimizing assignment.
- Sample:
- Given a 3 × 3 matrix A

	task1	task2	task3
worker1	8	25	50
worker2	50	35	75
worker3	22	48	150

where 8 at position (worker1, task1) represents the time cost of worker1 to complete task1.

Example

- **Step1**
  - The minimum element is taken and subtracted from each element in that row. This will lead to at least one zero in that row. This procedure is repeated for all rows. We now have a matrix with at least one zero per row.

	task1	task2	task3
worker1	8-8	25-8	50-8
worker2	50-35	35-35	75-35
worker3	22-22	48-22	150-22

The red box is the minimum value of each row.

Example

- **Step2**
  - After Step 1, since there is no zeros in column task3, which means none of the three workers is selected by the algorithm to finish the task3. It turn out that the matrix at this stage cannot be used for assigning.

	task1	task2	task3
worker1	0	17	42
worker2	15	0	40
worker3	0	26	128

  - To overcome this, we repeat the Step1 procedure for all columns - the minimum element in each column is subtracted from all the elements in that column, and then check if every column do have a zero element.

	task1	task2	task3
worker1	0	17	42-40
worker2	15	0	40-40
worker3	0	26	128-40

Example

- **Step3**
  - Mark all the zeros in the matrix with lines and use as few lines as possible.

	task1	task2	task3
worker1	0	17	2
worker2	15	0	0
worker3	0	26	88

We can use two red lines to cover all zeros in the matrix.

Apparently, the numbers of the lines is less than the length of the matrix. With this situation, we need to continue to Step 4 to adjust the matrix.

Example

- **Step4**
  - Find the minimum value from elements which are not marked by red lines. Subtract it from every unmarked element and add it to every marked non-zero element .

	task1	task2	task3
worker1	0	17-2	2-2
worker2	15+2	0	0
worker3	0	26-2	88-2

The minimum value which are not marked is 2.

Example

- Repeat Step 3–4 until n lines are marked(in this matrix, n is 3).

	task1	task2	task3
worker1	0	15	0
worker2	17	0	0
worker3	0	24	86

We use three red lines to cover all zeros.

- When the minimum number of lines used to cover all the zeros is equal to minimum of number of workers and tasks, the algorithm stop.

Example

- Choose the best matching with the processed matrix.

	task1	task2	task3
worker1	0	15	[✓ 0]
worker2	17	[✓ 0]	0
worker3	[✓ 0]	24	86

Finally, we choose the same positions of the best matching from the processed matrix as the best matching of the original matrix.

	task1	task2	task3
worker1	8	25	[✓ 50]
worker2	50	[✓ 35]	75
worker3	[✓ 22]	48	150

Example

- **Procedure to accomplish Step3**
  - At first, we need to find the row with the fewest zero elements. So, we can convert the previous matrix to the boolean matrix($0 \rightarrow$ True, Others $\rightarrow$ False).

	task1	task2	task3
worker1	True	False	False
worker2	False	True	True
worker3	True	False	False

Corresponding Boolean matrix.

  - Then we find the row with fewest True elements, for this example we choose the row worker1, since it have only one True element.

Example

- Second, mark any True elements on the corresponding row with green border and clean up its row and column, which means convert elements on the Boolean matrix to False.

	task1	task2	task3
worker1	True	False	False
worker2	False	True	True
worker3	True	False	False

	task1	task2	task3
worker1	False	False	False
worker2	False	True	True
worker3	False	False	False

Example

- Repeat the procedure until there is no True element in the matrix.

	task1	task2	task3
worker1	False	False	False
worker2	False	True	True
worker3	False	False	False

↓

	task1	task2	task3
worker1	False	False	False
worker2	False	False	False
worker3	False	False	False

Example

- The process will repeat several times until the elements in the boolean matrix are all set to False. The below table shows the marked True in the last procedure.

	task1	task2	task3
worker1	True 1	False	False
worker2	False	True 2	True
worker3	True	False	False

Example

step(1)、 Cross out rows that don’t have True element with green border.

	task1	task2	task3
worker1	True 1	False	False
worker2	False	True 2	True
worker3	~~True~~	~~False~~	~~False~~

step(2)、 Cross out column that contains True element without green border, from the row we crossed at step(1) or setp(3):

	task1	task2	task3
worker1	~~True 1~~	~~False~~	~~False~~
worker2	~~False~~	True 2	True
worker3	~~True~~	~~False~~	~~False~~

step(3)、 Check if the current crossed column at step(2), if there is True element with green border. Cross out the row that contains that True element, and back to step（2）.

Example

step(4)、 Draw lines on every uncrossed row and every crossed columns, that is the fewest lines to covers all zeros :

	task1	task2	task3
worker1	True 1	False	False
worker2	False	True 2	True
worker3	True	False	False

Practice

- You can get the initial value and the part of the code in *Hungarian_Algorithm.py,* please implement the Hungarian Algorithm.

Thank you !