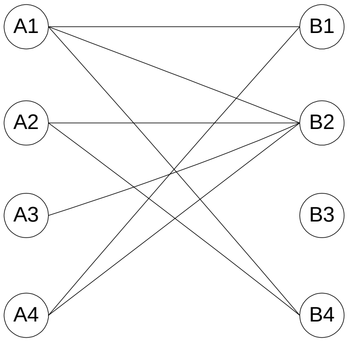
Hungarian Algorithm 匈牙利算法

Hungarian algorithm is method to solve maximal matching problem in bipartite graph. The theory is difficult to understand, but I will illustrate the procedure this algorithm.

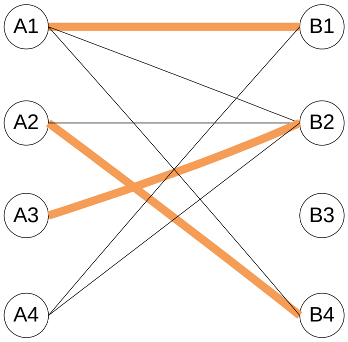
In a bipartite graph, left side is set A, right side is set B. The element in A can choose only 1 element in B from its connected candidates. Once mapping, the two connected elements cannot be reused. We want to calculate the maximal mappings in this bipartite.

For example, the bipartite is listed as follows:



Original graph

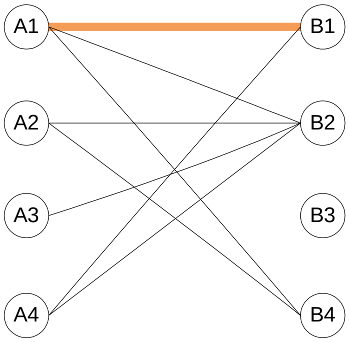
The maximal mappings is 3 as listed as follows:



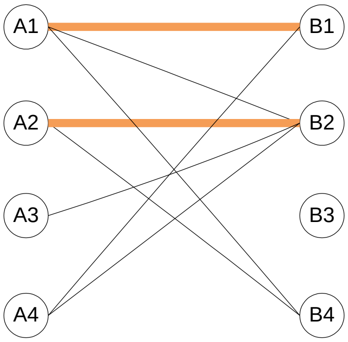
Maximal mapping

The Hungarian algorithm is a methodology to find the maximal mapping. The detail is listed as follows:

1） When A1 and A2 comes seperately

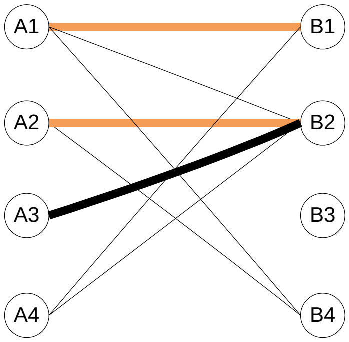


A1: A1

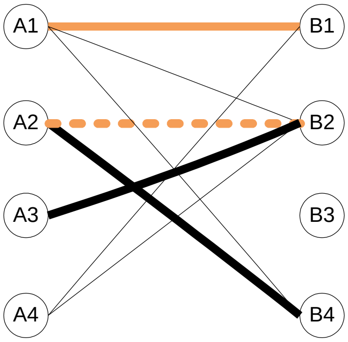


A2

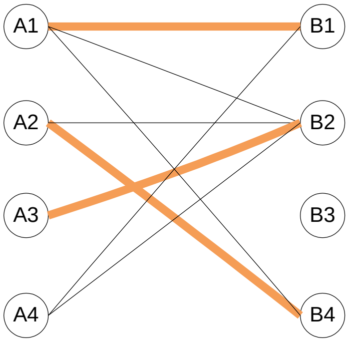
2) When A3 comes,



A3: Step 1



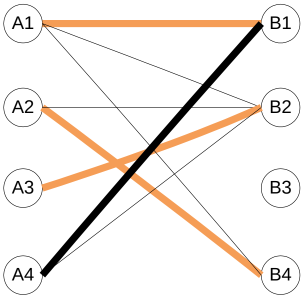
A3: Step 2



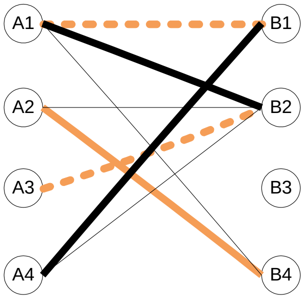
A3: Step 3

Comments:
A3 backtraces to B2 where B2 is occupied by A2
B2 backtraces to A2 where A2 has another connection to B4
B4 is free and not occupied. Assign B4 to A2.

3) When A4 comes,

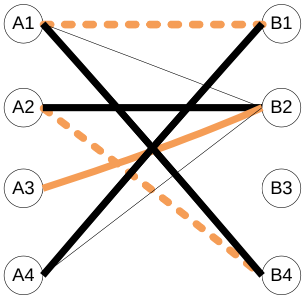


A4: Step 1



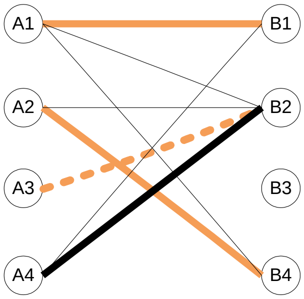
A4: Step 2

Comments:
A4 backtraces to B1 where B1 is occupied by A1
B1 backtraces to A1 where A1 has another connection to B2
B2 backtraces to A3 where B2 is occupied by A3
A3 has no more connection execpt B2. (B2 does not work)
So fail in this trial as we want to keep A1, A2, A3 connected.



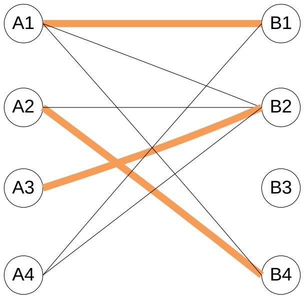
A4: Step 3

Comments:
A4 backtraces to B1 where B1 is occupied by A1
B1 backtraces to A1 where A1 has another connection to B4
B4 backtraces to A2 where B4 is occupied by A2
A2 backtraces to B2 where A2 has connection to B2
But in step 2, the trail from B2 does not work



A4: Step 4

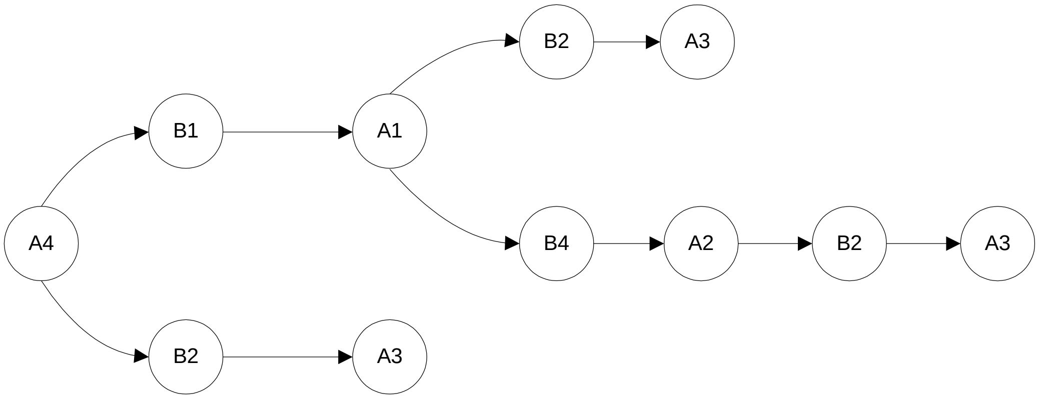
Comments:
A4 backtraces to B2 where B2 is occupied by A3
But in step 2, the trail from B2 does not work



A4 is not assigned to any point in B set

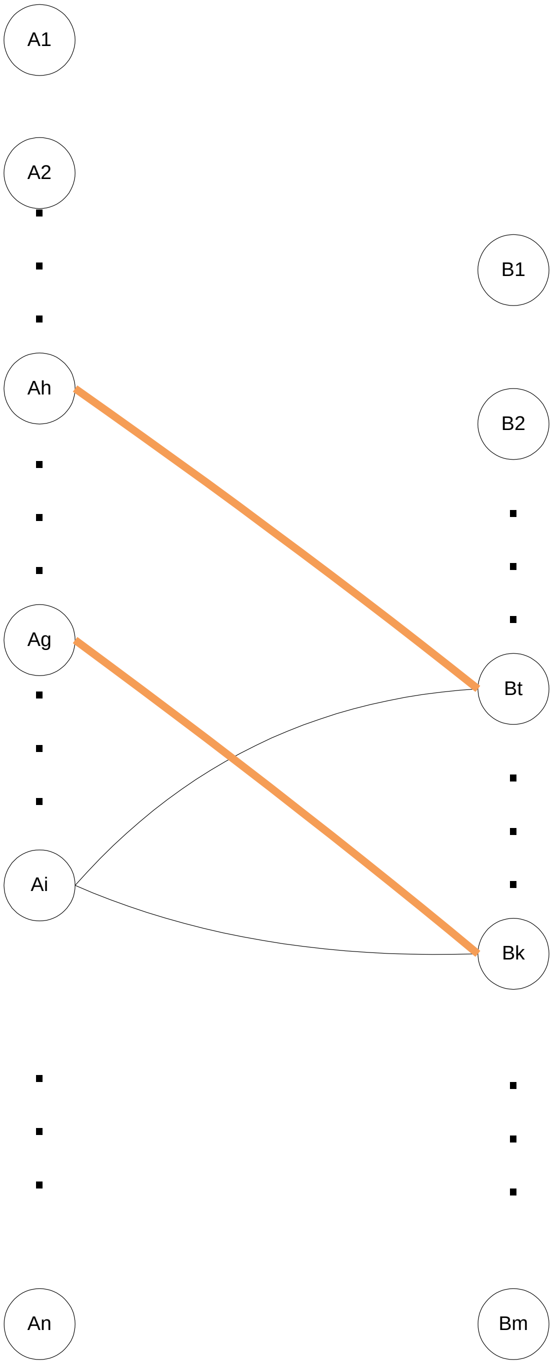
A4: Step 5

The A4 search path is as follows:



A4 search path

For general situation,



When Ai comes, it can be backtraced to Bt and Bk.
For Bt, it is occupied by Ah, Can Ah find another connected point in B set? If yes, set Ai connected Bt. If no, go to Bk.
For Bk, it is occupied by Ag, Can Ag find another connected point in B set? If yes, set Ai connected Bk.

```
For Ai comes, where i belong 1 to n.
For j = 1 to m
        If (connection[i][j]==1) // has connection.
        {
                If （Bj is used） // remember Fig A4: Step 4, “B2 doesnot work”
                // if Bj is used, then from this start point to backtrace, it means previously search from Bj is proved to be not working.
            Continue;
        If (Bj is not used) // we only process the first time.
        {
            Set Bj to used.
            If Bj is not assigned to any point in A,
            {
                Bj assigned to Ai
                Return found;
            }
            Else if Bj is assigned to point P in A
            {
                Can P found another connected point in B?        (recurse process)
                If yes, Bj assigned to Ai and return found
                If no, continue;
            }
        }
    }

    Return NotFound;
```