Maximum Bipartite Matching Documentation

Algorithms Analysis and Design II Final Project

Aya Fathy Mohammed

2017/10861

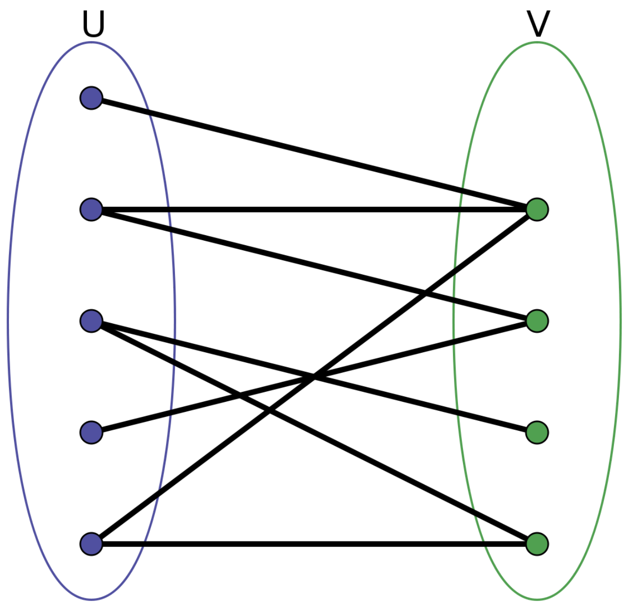
Supervised By: Dr. Fatty Salem, Eng. Shereen Essam

14.06.2020
Spring 2020

Introduction

• Bipartite Graph

A bipartite graph or a bigraph G = (V,E) is a graph decomposed on a set of nodes divided into two disjoint subsets U and V as shown in the figure. [1]



In each subset, nodes cannot be adjacent where an edge from can only connect a node belonging to subset V to a node belonging to subset U.

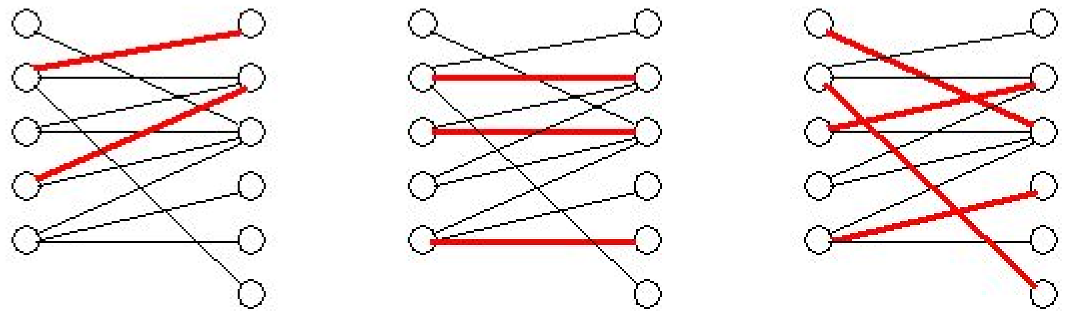
• Maximum Bipartite Matching

The objective of maximum bipartite matching is finding maximum cardinality matching for the proposed set of nodes.

In other words we need to find a globally optimal largest number of edges between the two subsets of nodes. In maximum matching, any two chosen edges cannot share the same endpoint,so we keep adding edges one by one until no more edges can be added without disturbing the matching.[2]

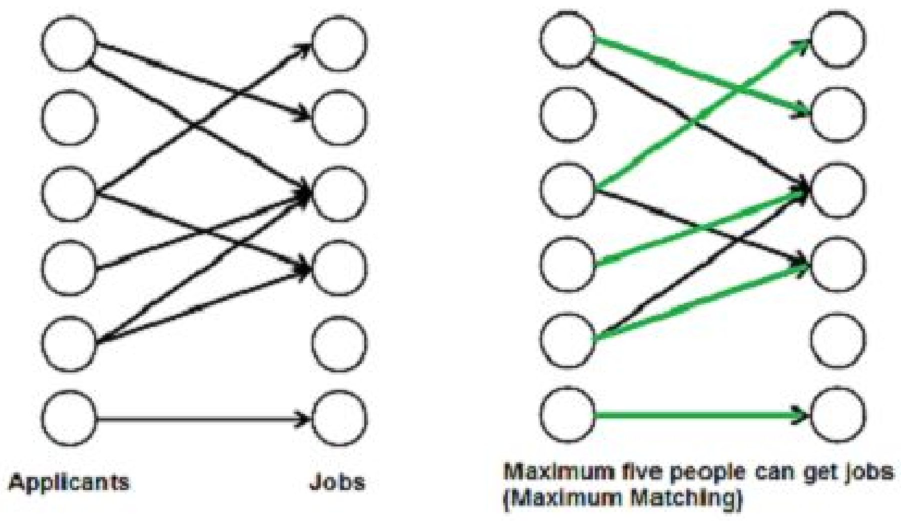
[1] "What is a bipartite graph? - Educative." https://www.educative.io/edpresso/what-is-a-bipartite-graph. Accessed 15 Jun. 2020.
[2] "Introduction to Algorithms, Third Edition | The MIT Press." https://mitpress.mit.edu/books/introduction-algorithms-third-edition. Accessed 15 Jun. 2020.

As shown in the figure below, this bipartite graph has several possible matching results.



• Example

There are so many day-to-day life problems that can be solved using maximum bipartite matching, for example here we have a number of applicants applying for a number of jobs, each job has a single vacancy that can be assigned to a single applicant. Our goal is to recruit as many applicants as possible and fill these vacancies.



Proposed Algorithm: Maximum Flow (Ford Fulkerson)

The below steps explain hoe Maximum flow is used to solve the applicants and jobs problem.

- ### Steps
  - User is asked to input Number of applicants R and the number of Number of jobs C.
  - User inserts values of R and C.

  - User is asked to input a set of boolean values (true or false) representing a bipartite graph of data type 2D boolean array: bipartiteGraph[R][C].
    - Each boolean value expresses whether applicant i is interested in job j or not

  - maxMatching(bipartiteGraph, R, C) function is called to find maximum bipartite matching for bipartiteGraph[R][C] :

    1. keep track of the applicants assigned to jobs in array matched[], initially all values = -1.
    2. Count of jobs assigned to applicants "result " is initially = 0.
    3. All jobs are marked as vacant: visited[i] = false

4. A recursive boolean function based on Depth First Search checkMatching(bipartiteGraph, u, visited, matched, C) is called to determine whether an applicant u can be matched to a job v they are interested in:
   - ➔ Check each job applicant u is interested in :

   A. If the job is still vacant, we simply assign it to the applicant u and return true.

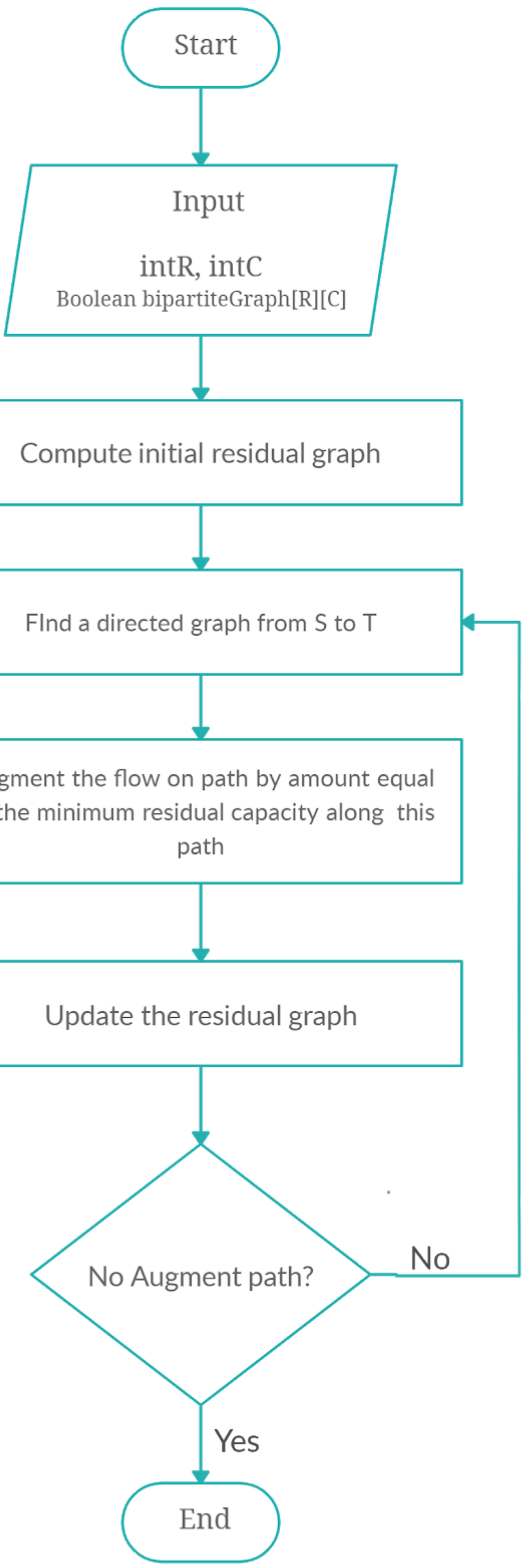
   B. If the job is assigned to another applicant w, Check if applicant w can be assigned to another vacancy:

      a. Mark job v as visited so w doesn't get the job again.
      b. Call the recursive function again for applicant w.
      c. If w can be assigned another vacancy then change applicant of job v from applicant w to u, and return true.

5. Count of jobs assigned to applicants "result" is incremented.

- Return result = Number of maximum bipartite Matching.

- Flow Chart


Start
Input
intR, intC
Boolean bipartiteGraph[R][C]
Compute initial residual graph
FInd a directed graph from S to T
Augment the flow on path by amount equal to the minimum residual capacity along this path
Update the residual graph
No Augment path?
No
Yes
End

- ## Input
    - Number of rows: int R
        - ➔ Size = 4 bytes = 32 bits
        - ➔ Value used in output screenshot below = 4
    - Number of columns : int C
        - ➔ Size = 4 bytes = 32 bits
        - ➔ Value used in output screenshot below = 4
    - Bipartite Graph: boolean bipartiteGraph[R][C]
        - ➔ Size =
        - ➔ Length = Number of Rows (R) = 4 in this case
        - ➔ Value used in output screenshot below =

          {true, true, false, true},

          {false, true, true, false},

          {false, false, true, false},

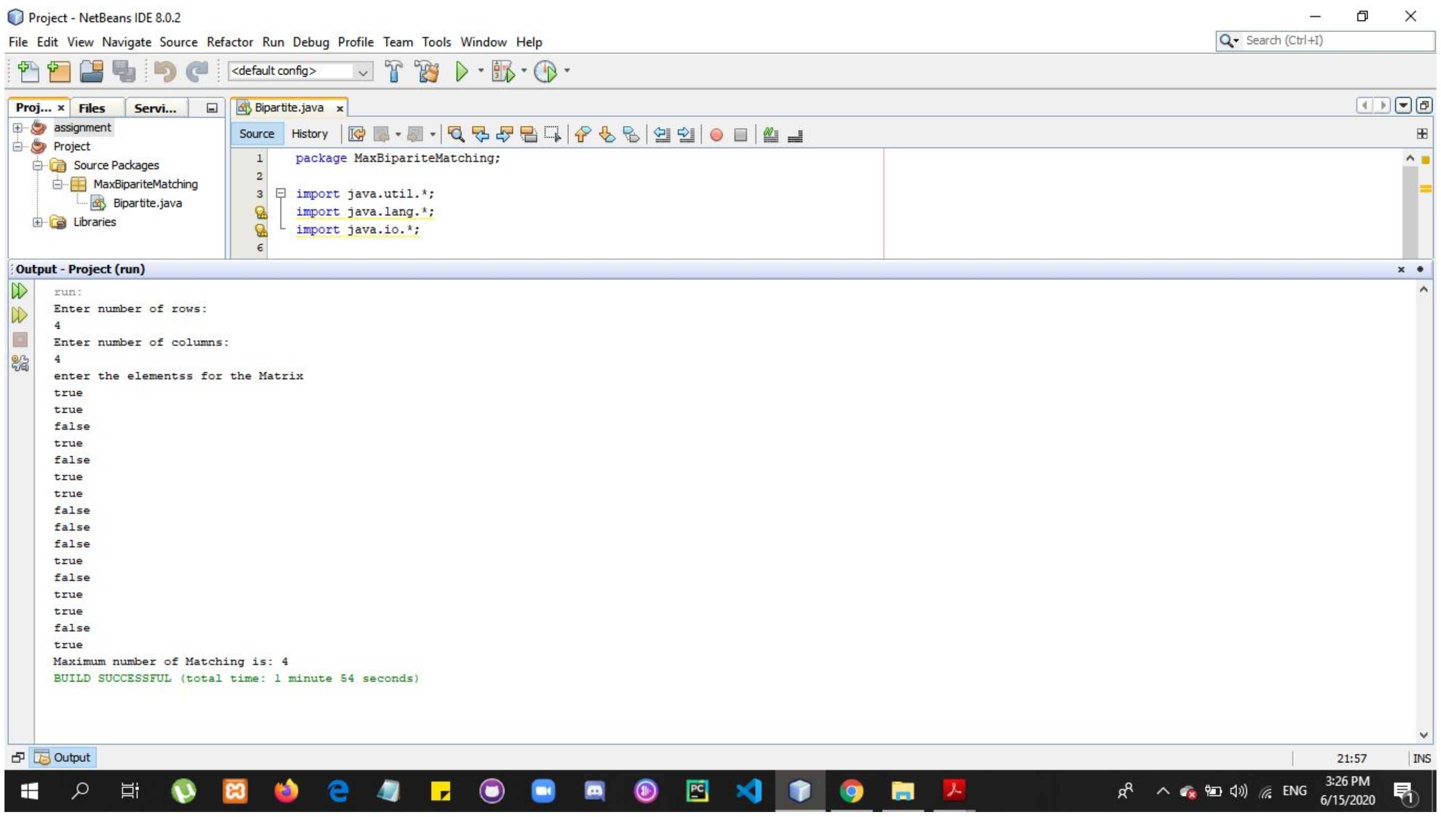
          {true, true, false, true}

- ## Output

The program returns the Maximum number of matching in the Bipartite graph from set R to set C.

In the output screenshot below, Maximum Bipartite matching = 4.

➔ Screenshots

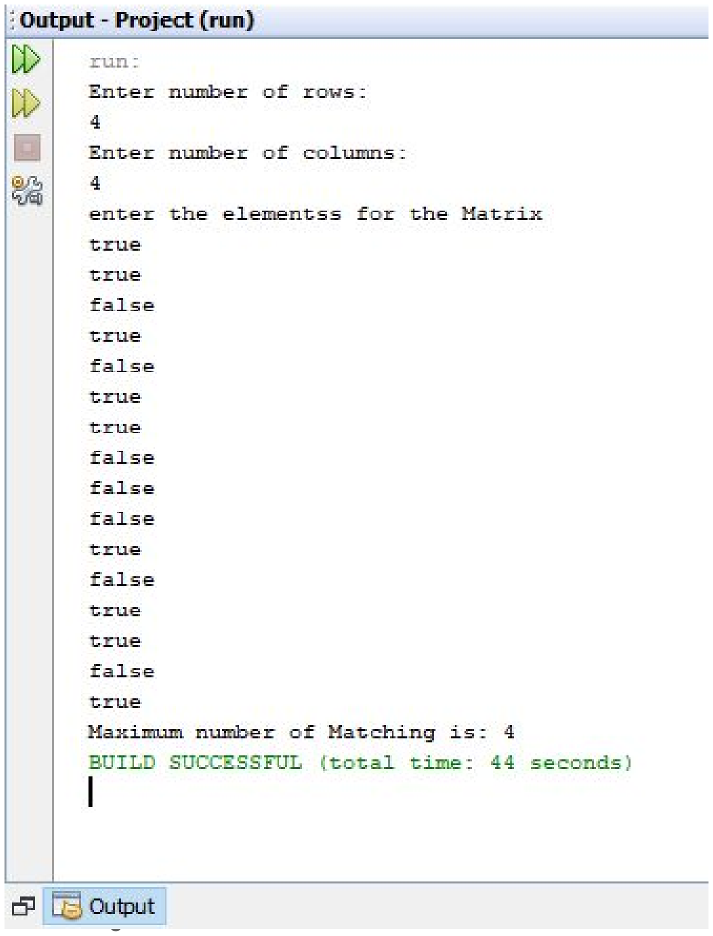


Project - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Bipartite.java
Source History
Output - Project (run)

```
package MaxBipariteMatching;

import java.util.*;
import java.lang.*;
import java.io.*;
```

```
run:
Enter number of rows:
4
Enter number of columns:
4
enter the elementss for the Matrix
true
true
false
true
false
true
true
false
false
false
true
false
true
true
false
true
Maximum number of Matching is: 4
BUILD SUCCESSFUL (total time: 1 minute 54 seconds)
```



```
run:
Enter number of rows:
4
Enter number of columns:
4
enter the elementss for the Matrix
true
true
false
true
false
true
true
false
false
false
true
false
true
true
false
true
Maximum number of Matching is: 4
BUILD SUCCESSFUL (total time: 44 seconds)
```

- ## Time Complexity

Maximum bipartite matching by Max flow uses a recursive DFS based function to traverse through graph nodes and find a match from Source to Sink. DFS algorithm takes $O(E)$ time, where E is the number of edges. The whole program takes $O(E * F)$ where F is maximum flow or maximum matching. Worst cases time complexity of this algorithm is , where V is the number of vertices.

Other Possible Solutions

1. Naive Greedy Algorithm

Assigns applicant to the first job it finds. This algorithm has better time complexity = O(N) than Maximum flow but it's not technically optimal because more applicants could have been assigned jobs too.[3] [4]

2. The Hopcroft-Karp bipartite matching algorithm

Hopcroft-karp is a push-relabel max flow based algorithm, when it gets close to optimum state while matching nodes it switches to Hopcroft-karp method. It runs with time complexity $O(\sqrt{V}\ x\ E)$, where V is number of nodes and E is number of edges.[5]

[3] "A Guide to Algorithm Design: Paradigms, Methods, and ...." https://books.google.com/books/about/A_Guide_to_Algorithm_Design.html?id=LyHSBQAAQBAJ. Accessed 15 Jun. 2020.
[4] "Classroom Assignment 3 Maximum Bipartite Matching." 31 Oct. 2017, https://www2.cs.duke.edu/courses/fall17/compsci330/lecture16note.pdf. Accessed 15 Jun. 2020.
[5] "Introduction to Algorithms, Third Edition | The MIT Press." https://mitpress.mit.edu/books/introduction-algorithms-third-edition. Accessed 15 Jun. 2020.

REFERENCES

1. "What is a bipartite graph? - Educative." https://www.educative.io/edpresso/what-is-a-bipartite-graph. Accessed 15 Jun. 2020
2. "Introduction to Algorithms, Third Edition | The MIT Press." https://mitpress.mit.edu/books/introduction-algorithms-third-edition. Accessed 15 Jun. 2020.
3. "A Guide to Algorithm Design: Paradigms, Methods, and ...." https://books.google.com/books/about/A_Guide_to_Algorithm_Design.html?id=LyHSBQAAQBAJ. Accessed 15 Jun. 2020.
4. "Classroom Assignment 3 Maximum Bipartite Matching." 31 Oct. 2017, https://www2.cs.duke.edu/courses/fall17/compsci330/lecture16note.pdf. Accessed 15 Jun. 2020.