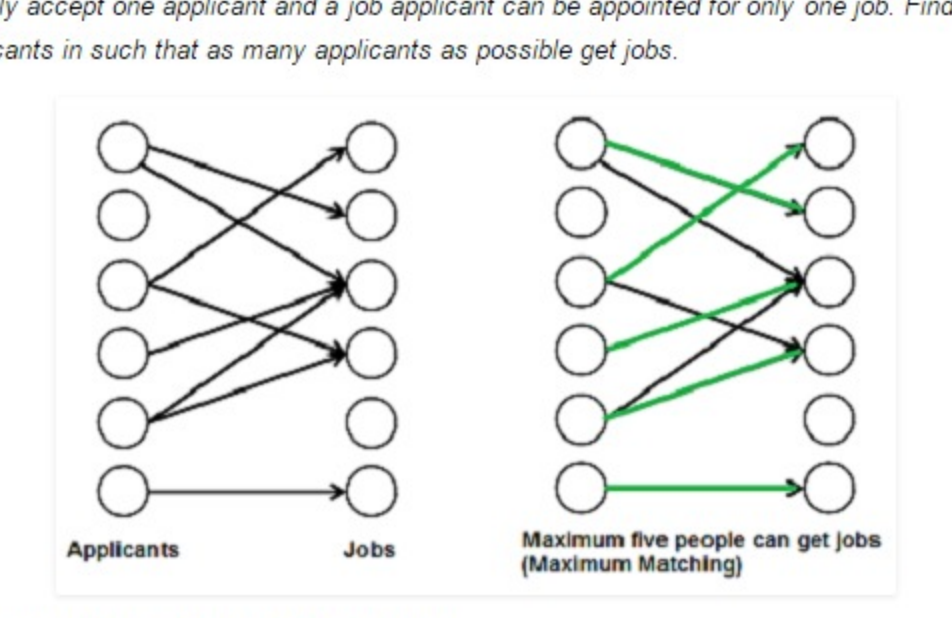
Maximum Bipartite Matching

A matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matchings for a given Bipartite Graph.

Why do we care?

There are many real world problems that can be formed as Bipartite Matching. For example, consider the following problem:

There are M job applicants and N jobs. Each applicant has a subset of jobs that he/she is interested in. Each job opening can only accept one applicant and a job applicant can be appointed for only one job. Find an assignment of jobs to applicants in such that as many applicants as possible get jobs.

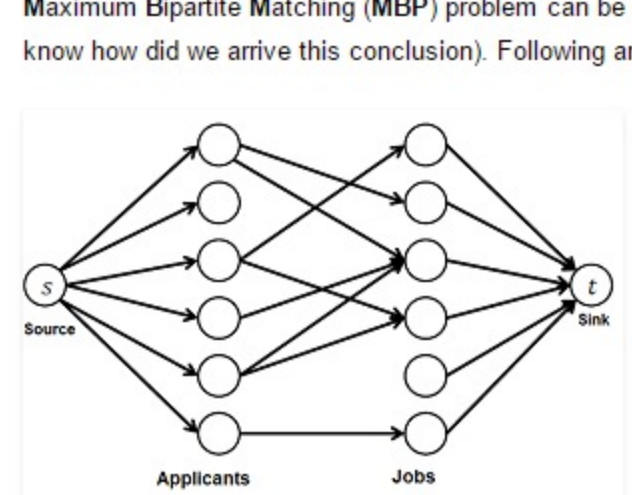


We strongly recommend to read the following post first.

Ford-Fulkerson Algorithm for Maximum Flow Problem

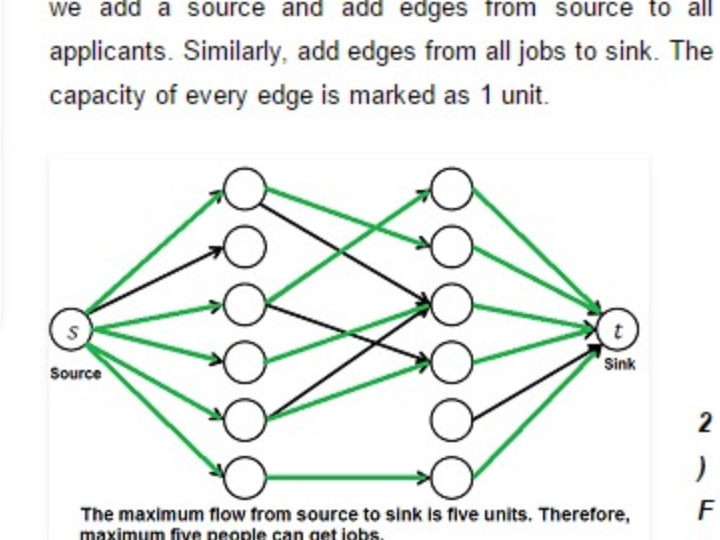
Maximum Bipartite Matching and Max Flow Problem

Maximum Bipartite Matching (**MBP**) problem can be solved by converting it into a flow network (See this video to know how did we arrive this conclusion). Following are the steps.



1) Build a Flow Network

There must be a source and sink in a flow network. So we add a source and add edges from source to all applicants. Similarly, add edges from all jobs to sink. The capacity of every edge is marked as 1 unit.



2) Find the maximum flow.

We use Ford-Fulkerson algorithm to find the maximum flow in the flow network built in step 1. The maximum flow is actually the MBP we are looking for.

How to implement the above approach?

Let us first define input and output forms. Input is in the form of Edmonds matrix which is a 2D array 'bpGraph[M][N]' with M rows (for M job applicants) and N columns (for N jobs). The value bpGraph[i][j] is 1 if i'th applicant is interested in j'th job, otherwise 0.

Output is number maximum number of people that can get jobs.

A simple way to implement this is to create a matrix that represents adjacency matrix representation of a directed graph with M+N+2 vertices. Call the fordFulkerson() for the matrix. This implementation requires O((M+N)*(M+N)) extra space.

Extra space can be be reduced and code can be simplified using the fact that the graph is bipartite and capacity of every edge is either 0 or 1. The idea is to use DFS traversal to find a job for an applicant (similar to augmenting path in Ford-Fulkerson). We call bpm() for every applicant, bpm() is the DFS based function that tries all possibilities to assign a job to the applicant.

In bpm(), we one by one try all jobs that an applicant 'u' is interested in until we find a job, or all jobs are tried without luck. For every job we try, we do following.

If a job is not assigned to anybody, we simply assign it to the applicant and return true. If a job is assigned to somebody else say x, then we recursively check whether x can be assigned some other job. To make sure that x doesn't get the same job again, we mark the job 'v' as seen before we make recursive call for x. If x can get other job, we change the applicant for job 'v' and return true. We use an array maxR[0..N-1] that stores the applicants assigned to different jobs.

If bmp() returns true, then it means that there is an augmenting path in flow network and 1 unit of flow is added to the result in maxBPM().

C++ Java

```
// A Java program to find maximal Bipartite matching.
import java.util.*;
import java.lang.*;
import java.io.*;

class MaxBipartite
{
    // M is number of applicants and N is number of jobs
    static final int M = 6;
    static final int N = 6;

    // A DFS based recursive function that returns true if a
    // matching for vertex u is possible
    boolean bpm(boolean bpGraph[][], int u, boolean seen[],
                int matchR[])
    {
        // Try every job one by one
        for (int v = 0; v < N; v++)
        {
            // If applicant u is interested in job v and v
            // is not visited
            if (bpGraph[u][v] && !seen[v])
            {
                seen[v] = true; // Mark v as visited

                // If job 'v' is not assigned to an applicant OR
                // previously assigned applicant for job v (which
                // is matchR[v]) has an alternate job available.
                // Since v is marked as visited in the above line,
                // matchR[v] in the following recursive call will
                // not get job 'v' again
                if (matchR[v] < 0 || bpm(bpGraph, matchR[v],
                                         seen, matchR))
                {
                    matchR[v] = u;
                    return true;
                }
            }
        }
        return false;
    }

    // Returns maximum number of matching from M to N
    int maxBPM(boolean bpGraph[][])
    {
        // An array to keep track of the applicants assigned to
        // jobs. The value of matchR[i] is the applicant number
        // assigned to job i, the value -1 indicates nobody is
        // assigned.
        int matchR[] = new int[N];

        // Initially all jobs are available
        for(int i=0; i<N; ++i)
            matchR[i] = -1;

        int result = 0; // Count of jobs assigned to applicants
        for (int u = 0; u < M; u++)
        {
            // Mark all jobs as not seen for next applicant.
            boolean seen[] =new boolean[N] ;
            for(int i=0; i<N; ++i)
                seen[i] = false;

            // Find if the applicant 'u' can get a job
            if (bpm(bpGraph, u, seen, matchR))
                result++;
        }
        return result;
    }

    // Driver method
    public static void main (String[] args) throws java.lang.Exception
    {
        // Let us create a bpGraph shown in the above example
        boolean bpGraph[][] = new boolean[][]{
            {false, true, true, false, false, false},
            {true, false, false, true, false, false},
            {false, false, true, false, false, false},
            {false, false, true, true, false, false},
            {false, false, false, false, false, false},
            {false, false, false, false, false, true}
        };
        MaxBipartite m = new MaxBipartite();
        System.out.println( "Maximum number of applicants that can"+
                            " get job is "+m.maxBPM(bpGraph));
    }
}
```

Run on IDE

Output:

```
Maximum number of applicants that can get job is 5
```

You may like to see below also:

Hopcroft–Karp Algorithm for Maximum Matching | Set 1 (Introduction)

Hopcroft–Karp Algorithm for Maximum Matching | Set 2 (Implementation)

References:

http://www.cs.cornell.edu/~wdtseng/icpc/notes/graph_part5.pdf
http://www.youtube.com/watch?v=NlQmEXuiC8
http://en.wikipedia.org/wiki/Maximum_matching
http://www.stanford.edu/class/cs97si/08-network-flow-problems.pdf
http://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07NetworkFlowII-2×2.pdf
http://www.ise.ncsu.edu/fangroup/or766.dir/or766_ch7.pdf

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Company Wise Coding Practice   Topic Wise Coding Practice

36 Comments Category: Graph Tags: Graph

Related Posts:

- Stepping Numbers
- Clone an Undirected Graph
- Longest Path in a Directed Acyclic Graph | Set 2
- Find Shortest distance from a guard in a Bank
- Topological Sort of a graph using departure time of vertex
- Finding minimum vertex cover size of a graph using binary search
- Find the number of Islands | Set 2 (Using Disjoint Set)
- Check if two nodes are on same path in a tree

Previous post in category   Next post in category

(Login to Rate and Mark)
4.5 Average Difficulty : 4.5/5.0 Based on 6 vote(s)
Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

36 Comments GeeksforGeeks Login

Recommend 2 Share Sort by Newest

Join the discussion...

Lokesh Jain · 2 months ago
In the code I don't think this code will ever be executed for the second part of the OR operation.
if (matchR[v] < 0 || bpm(bpGraph, matchR[v], seen, matchR))
If matchR[v]>=0 then seen[v]=true already by a previous call for bpm function. But the outer if condition requires seen[v] to be false which is a contradiction.
Reply · Share ›

Amar Kaswan · 2 months ago
Anybody please suggest the time complexity!!
Reply · Share ›

Chau Thai · 5 months ago
For people who say the method doesn't use ford-fulkerson algorithm:
Please read the this topcoder tutorial https://www.topcoder.com/commu...

This method may seem like backtracking, but actually is an optimized version of ford-fulkerson only for Bipartite Matching.
Reply · Share ›

Deepak Mehta · 5 months ago
For python enthusiasts, here is the code :). http://pastebin.com/uteYf348
Reply · Share ›

rita sakhuja · 7 months ago
Well explained using ford fulkerson at this link
http://www.codebytes.in/2015/1...
Reply · Share ›

Billionaire · a year ago
Mark! Network Flow problem
Reply · Share ›

brahim · a year ago
"A maximum matching is a matching of maximum size (maximum number of edges). In other words, a matching is maximum if any edge is added to it, it is no longer a matching."
The two sentences are not equivalent. You can have a matching where you can not add any other edge without it being a matching of maximal size.
Reply · Share ›

GeeksforGeeks Mod → brahim · a year ago
We have updated the definition.
Reply · Share ›

utkarsh · a year ago
How to print all such matchings ??
Reply · Share ›

Johan Luxx Burstrom · 2 years ago
Wow this really explained it in a simple way, thanks!
Reply · Share ›

kaushik Bhattacharya · 2 years ago
Thanks Prasenjit, Your answer made me doubt clear.
Reply · Share ›

kaushik Bhattacharya · 2 years ago
How are you creating the adjacency matrix for this? The matrix given in the code does not match with the graph given in the diagram. Can you explain? I am gone out of ideas. Please help
Reply · Share ›

nguyen thanhtruc · 2 years ago
{{1, 1, 1, 1, 0},

{0, 0, 0, 1, 0},

{1, 1, 1, 0, 0},

{0, 1, 1, 1, 1},

{0, 1, 1, 1, 1}

};

Try the above matrix, I am afraid that the result is wrong!
Reply · Share ›

torreto · 2 years ago
how to find all such matchings >
Reply · Share ›

Prasenjit Mondal · 2 years ago
The method used here does not use the ford-fulkerson algorithm at all instead the problem is solved using backtracking.
2 Reply · Share ›

kaushik Bhattacharya → Prasenjit Mondal · 2 years ago
Hi Prasenjit, I have a question on the matrix shown in the example. I cant understand how the matrix is written from the figure shown
Reply · Share ›

Prasenjit Mondal → kaushik Bhattacharya · 2 years ago
The matrix is wrong in the example. Currect matrix would be

{0, 1, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 0},

{1, 0, 0, 1, 0, 0},

{0, 0, 1, 0, 0, 0},

{0, 0, 1, 1, 0, 0},

{0, 0, 0, 0, 0, 1}

M = 6
N = 6

Give numbers 0-5 to M applicants node and same for job nodes. The matrix is adjacency matrix representation where matrix[i][j] = 1 if (i,j) belongs to an Edge. 1 is the capacity here. Hope it clears your doubt.
Reply · Share ›

Amit Handa · 2 years ago
pardon my understanding: I dont think ford fulkerson is being used here. Ford method uses DFS/BFS to select a path and subtract the path from the graph to construct a residual graph.
In this method, the matching is being improved via backtracking. Its not the same.
3 Reply · Share ›

ms → Amit Handa · 2 years ago
I too feel this backtracking!!
1 Reply · Share ›

DD · 2 years ago
how can I know the pairs on the

Maximum

situation
Reply · Share ›

zz35 · 2 years ago
How can we count total number of maximum matchings possible in a general bipartite graph?
Reply · Share ›

Programmer · 2 years ago
What is order of This algorithm
3 Reply · Share ›

gurudas · 2 years ago
In the recursion call why are we sending the array seen which has true for all the jobs it has visited including the current job. It has to be sent only with the current job as true right...
1 Reply · Share ›

GOPI GOPINATH · 2 years ago
Why do we need to recursively check whether x can be assigned some other job ? once if we find a job (matchR[v] < 0), we can assign rite ??
Reply · Share ›

gurudas → GOPI GOPINATH · 2 years ago
we are calling recursively only if we cant find a job for the applicant.
Reply · Share ›

stefyvolt · 2 years ago
the matrix for the example is not the one you posted. you have to drag the fifth line above the second, like this:

{0, 1, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 0},

{1, 0, 0, 1, 0, 0},

{0, 0, 1, 0, 0, 0},

{0, 0, 1, 1, 0, 0},

{0, 0, 0, 0, 0, 1}
2 Reply · Share ›

rihansh · 3 years ago
i think as you said above matchr keeps track for the applicant for each job it shoud be matchR[M] not matchR[N]
correct me if i am wrong
1 Reply · Share ›

pulkit mehra → rihansh · 3 years ago
It will be matchR[N] since it keeps track of an applicant assigned to a job, so array will be from [0..N] for jobs and its values will be from 0-M keeping track of applicants.
1 Reply · Share ›

green_bear · 3 years ago
memset(matchR, -1, sizeof(matchR)); wont work as expected unless matchR is of char type, because memset will set every byte to -1 and int has 4 bytes.
1 Reply · Share ›

akhilu → green_bear · 3 years ago
It works for -1 and 0, because they are stored as all 1s and all 0s respectively.
Reply · Share ›

rahul23 · 3 years ago
Time complexity for this???m*n!???
plz tell
12 Reply · Share ›

Rubin Agrawal · 3 years ago
who
Reply · Share ›

atul · 3 years ago
how is space complexity is O((M+N)*(M+N)) extra space ??

```
/* Paste your code here (You may delete these lines if not writing code) */
```

Reply · Share ›

Rahul Sharma → atul · 3 years ago
You need to store all the nodes in the adjacency matrix which requires a 2D matrix of size (N+M+2)*(N+M+2) for the representation of the directed graph.
1 Reply · Share ›

Guest → Rahul Sharma · 3 years ago
But the space used in program is O(M*N) ...???
Reply · Share ›

Himanshu Srivastava · 3 years ago
Thanks a lot was wating for it!
Reply · Share ›

Subscribe   Add Disqus to your site   Privacy   DISQUS

Popular Posts
- Top 10 Algorithms and Data Structures for Competitive Programming
- Top 10 algorithms in Interview Questions
- How to begin with Competitive Programming?
- Reflection in Java
- Memory Layout of C Programs
- Push Relabel Algorithm
- Heavy Light Decomposition
- Sorted Linked List to Balanced BST
- Generics in Java
- Aho-Corasick Algorithm for Pattern Searching
- Binary Search , QuickSort , MergeSort , HeapSort

- Common Interview Puzzles
- Interview Experiences
- Advanced Data Structures
- Design Patterns
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Geometric Algorithms
- Searching
- Sorting
- Hashing
- Analysis of Algorithms
- Mathematical Algorithms
- Randomized Algorithms
- Recursion

Tags
Accolite Adobe Advance Data Structures Advanced Data Structures Amazon array Bit Magic C++ CN c puzzle D-E-Shaw Directi Divide and Conquer Dynamic Programming Flipkart GATE GATE-CS-DS-&-Algo GATE-CS-Older GFacts Goldman Sachs Google Graph Greedy Algorithm Hashing Interview Experience Java MakeMyTrip MAQ Software MathematicalAlgo Matrix Microsoft Morgan Stanley Operating systems Oracle Pattern Searching puzzle Python Recursion Samsung SAP Labs SnapDeal stack Stack-Queue STL Zoho

Recent Comments

Vineet Bhargava Thanks Rahul. To his point see this code:...
Is assignment operator inherited? · 26 minutes ago

Vineet Bhargava can ->* be overloaded?
What are the operators that cannot be overloaded in C++? · 1 hour ago

Sai Bhargav Why can't we run activity selection multiple...
Minimum Number of Platforms Required for a Railway/Bus Station · 1 hour ago

Shalini Salomi Bodapati The below code prints 5.why?As a is local...
Output of C Programs | Set 2 · 1 hour ago

Vineet Bhargava @GeeksForGeeks: Great article but I would like...
When should we write our own assignment operator in C++? · 1 hour ago

Deepankar Singh Wrong logic, this will not work. check for...
Transform One String to Another using Minimum Number of Given Operation · 1 hour ago

All Categories
Select Category
- GeeksQuiz
- GeeksforGeeksIDE
- GeeksforGeeks Practice
- Data Structures
- Algorithms
- C Programming
- C++ Programming
- Java Programming
- Books
- Interview Experiences
- GATE CS
- GATE CS Forum
- Android App