Can everyone get married? Matching in Bipartite Graphs

Md Farhan Ishmam[1] Nejd Khadija [1]

[1]Islamic University of Technology

Introduction

Have you ever wondered whether all your friends will be able to get married after graduation? Surprisingly, this is a **graph problem**. A graph is a collection of objects called **vertices** that are connected by **edges** representing a connection or relationship between the objects. A graph is simply a representation of a problem or scenario.

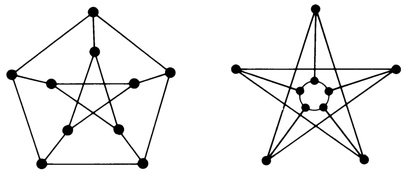

Figure 1. Two of the coolest graphs we have encountered. The black dots represent the vertices and the lines represent the edges. The two graphs are surprisingly the same graph (isomorphic). Can you find a way to rearrange the vertices and edges of the first graph to make the second one?

Bipartite Graph

Before formulating our marriage problem as a graph problem, we must first observe a key property of the marriage problem - a boy can only be married to a girl! And, the opposite is also true. Our condolences to the liberals, but in our problem, there will be no relationship (edge) between two boys or between two girls. Hence, the overall set of vertices can be divided into two *disjoint* and *independent* sets of vertices – one represents the boys, and the other represents the girls. A vertex of one set will have at most one edge with a vertex of the other set.

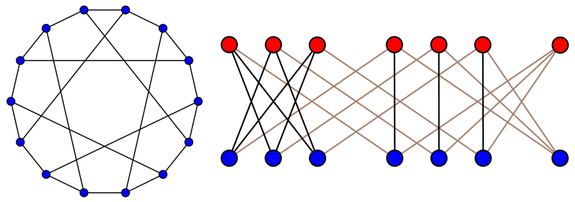

Figure 2. The Heawood graph (at the left) is one of the coolest graphs we have encountered and is surprisingly a bipartite graph (at the right).

Matching

Matching is a fundamental problem in graph theory that involves finding a set of edges that do not share a common vertex i.e. every edge is independent to each other. The concept of matching is essential to our marriage problem as a boy cannot (unfortunately) be married to more than one girl and vice versa. Randomly picking independent edges in a bipartite graph will give us some form of matching. However, we are interested in **maximum matching** i.e. the size of the matched set will be the largest. This ensures that *most* of our friends will end up getting married but not *all*. The matching where every vertex of our graph is included in the matched set is called **perfect matching** i.e. *all* the vertices or our friends end up married. But how do we ensure that perfect matching is possible in a scenario?

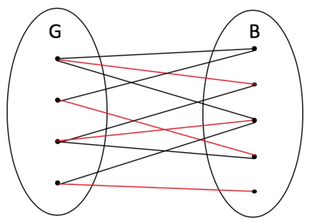

Figure 3. A relatively uncool bipartite graph where the disjoint independent sets of vertices G and B represent girls and boys respectively. The red edges represent the edges belonging to the largest matched set.

Hall's Marriage Theorem

Like us, the English mathematician Philip Hall was (probably) on a quest to determine whether all of his friends would get married. In pursuit of his goal, he introduced a theorem to find perfect matching for bipartite graphs by providing a *necessary* and *sufficient* condition. In simple terms, if Hall's condition is met then perfect matching is guaranteed!
Let us take a subset W from one of the bipartite sets of our bipartite graph. The set of vertices adjacent (directly connected) to the vertices of W is called the neighborhood of W or simply $N(W)$. Hall's condition is simply stated as:

$$|W| \leq |N(W)|$$

Every subset of our bipartite sets must have a neighborhood larger than itself to ensure perfect matching. It should be noted that the original theorem ensured perfect matching for only one bipartite set but repeating the procedure for both sets ensures overall perfect matching.

Figure 4. The English mathematician Philip Hall gave us Hall's Marriage Theorem. Mr. Hall seems like a pretty cool guy and is definitely someone with whom you should have a chat to discuss graph theory. Too bad you lost this opportunity in the 80s. (Philip Hall breathed his last in 1982 at the age of 78)

Algorithms

Hall's marriage theorem gives us the *condition* but not the *procedure*. To find which friend needs to marry whom to ensure everyone (or at least most people) get married, we need algorithms; one such algorithm is the **Hopcroft–Karp algorithm** which finds the largest (maximum) matching in a bipartite graph in $O(V\sqrt{E})$ time complexity.
The Hopcroft–Karp algorithm is a surprisingly efficient algorithm but not the most efficient one; some *randomized* and *approximation* algorithms work faster. In some special cases, for instance, if we have a *planar graph* (our whole graph can be drawn on a piece of paper), we can use that property to find the maximum matching even faster. Furthermore, the Hopcroft–Karp algorithm works with *unweighted* graphs only i.e. every relationship or edge in our graph has no value assigned to it and has equal preference.

Terminologies used in Hopcroft–Karp Algorithm

Augmenting Path

A simple path is an alternating sequence of vertices and edges where no edge or vertex is repeated. It is, simply, a way to travel from one vertex to another without revisiting a vertex. An **augmenting path** is a simple path that starts and ends with unmatched vertices but alternates between matched and unmatched edges.

Breath-First Search (BFS)

Visiting all the nodes of a graph isn't a straightforward task as there can be loops in a graph that might result in the continuous revisiting of the nodes. Hence, an algorithm is required to keep track of the visited nodes and to choose which node has to be picked first. **Breadth-first search** is such a *graph traversal* algorithm that visits all the vertices of a graph level by level i.e. if we have a start vertex then the algorithm will first visit all the vertices adjacent to it, then the vertices adjacent to the adjacent vertices, and so on.

Hopcroft–Karp Algorithm

We begin with an empty matching set and repeatedly add matched edges by finding augmenting paths. Each iterative matching set is called partial matching. As the augmented path contains edges that are not part of the partial matching, we can iteratively include those edges to obtain maximum matching. To find the augmenting paths, we use the BFS from the unmatched vertices in one set. If it finds an unmatched vertex in the other set, it has found an augmenting path and updates the matching accordingly. If no augmenting path is found, the algorithm terminates and returns the current matching as the maximum.

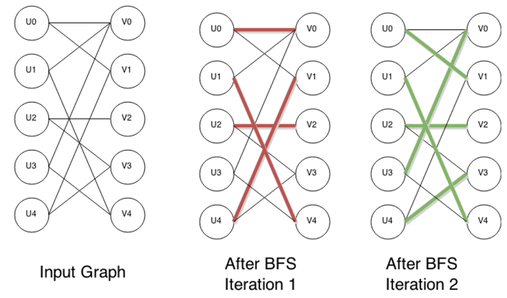

Figure 5. The execution of Hopcroft–Karp Algorithm for the input graph, an intermediate iteration 1 where partial matching is obtained, and the final iteration 2 where the maximum matching is obtained

Applications

The Hopcroft–Karp algorithm told us who has to marry whom so most of us end up married. While research shows that marriage does have a strong correlation with psychological well-being, our thoughts should not be limited to marriage only. Hence, we included a few applications of bi-partite graph matching for readers who wish to go beyond marriage:

- **District Representative Problem:** The first bipartite graph matching problem that Philip Hall gave a look and came up with a bunch of theorems. There are 5 senators and 3 committees and our goal is to find if it's possible to pick one senator from each committee to represent that district i.e. if perfect matching is possible or not.
- **Graduation Problem:** A less classical problem that involves a student who needs to take a certain number of classes/courses from a set of classes/courses to graduate. Let's say, we have two sets of classes "ABCD", and "DXYZ" where an ASCII character represents a class. The student has already taken "AD" and must take two classes from each set. Our problem is to find the minimum number of additional classes he has to take to graduate.
- **Hitchcock problem:** A transportation problem that involves multiple supply sources – each has units of supply, and multiple sinks – each has units of demand. There is a shipment cost from a source to a sink and our goal is to satisfy the demands while minimizing the cost. We can use bipartite graph matching as a subproblem to solve this problem.

References

[1] N. Deo.
Graph theory with applications to engineering and computer science.
Courier Dover Publications, 2017.

[2] F. L. Hitchcock.
The distribution of a product from several sources to numerous localities.
Journal of mathematics and physics, 20(1-4):224–230, 1941.

[3] J. E. Hopcroft and R. M. Karp.
An n^5/2 algorithm for maximum matchings in bipartite graphs.
SIAM Journal on computing, 2(4):225–231, 1973.

[4] C. M. Kamp Dush, M. G. Taylor, and R. A. Kroeger.
Marital happiness and psychological well-being across the life course.
Family relations, 57(2):211–226, 2008.

[5] D. B. West et al.
Introduction to graph theory, volume 2.
Prentice hall Upper Saddle River, 2001.

[6] Wikipedia.
Matching (graph theory) — wikipedia, the free encyclopedia, 2023.
[Online; accessed 27-March-2023].

Department of Computer Science and Engineering | Islamic University of Technology, Boardbazar, Gazipur | Email: farhanishmam@iut-dhaka.edu