CSE 300

Technical Writing and Presentation Sessional

Report: Maximum Bipartite Matching

Lab Section - A1

Group - 09

9th March, 2024

Members of the Group:

i. 2005012 - Abrar Jahin Sarker

ii. 2005017 - Abdullah Muhammed Amimul Ehsan

iii. 2005020 - Mostafa Rifat Tazwar

Contents

1 Introduction

The concept of maximum bipartite matching, a fundamental problem in graph theory, holds significant relevance across diverse fields due to its ability to model various real-world scenarios. Bipartite graphs serve as the foundational structure for this problem. In essence, a maximum bipartite matching seeks to pair elements from one subset with elements from the other subset in such a way that maximizes the number of paired elements. From applications in network flow optimization, job scheduling, and medical residency programs to its role in solving matching problems in societal contexts like marriage and kidney exchange programs, the utility of maximum bipartite matching extends across numerous domains, making it a subject of continued research and innovation.

1.1 Definitions

1.1.1 Bipartite Graph

A bipartite graph is one whose vertices can be split into two independent groups U,V such that **every edge connects vertices of different groups.**

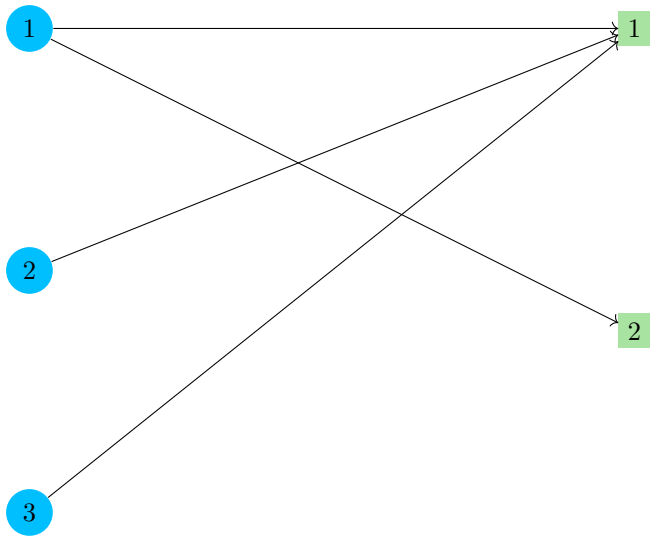
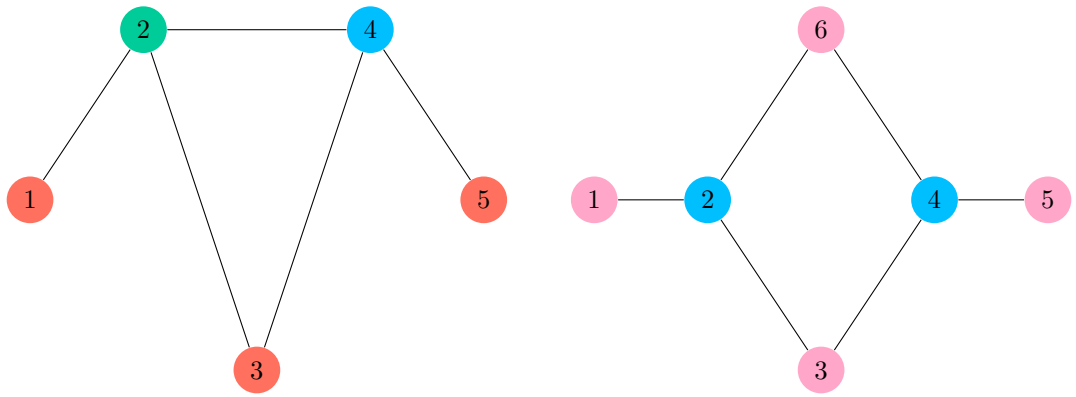


Figure 1: Visualization of bipartite graph

Properties of a Bipartite graph

- There can not be any edge between any two vertices of U or any two vertices of V .
- The graph is two colourable and doesn't have cycles of odd length.



Odd length cycle

Not 2 colorable

Not a bipartite graph

Even length cycle

2 colorable

Bipartite graph

Figure 2: Comparison between Bipartite and Non-bipartite graph

1.1.2 Maximum Matching

Given a bipartite graph, a matching is a subset of the edges for which **every vertex belongs to exactly one of the edges**.
And, a maximum matching is a matching of **maximum number of edges**. In a maximum matching, if any edge is added to it, it is no longer a matching.

2 The Problem

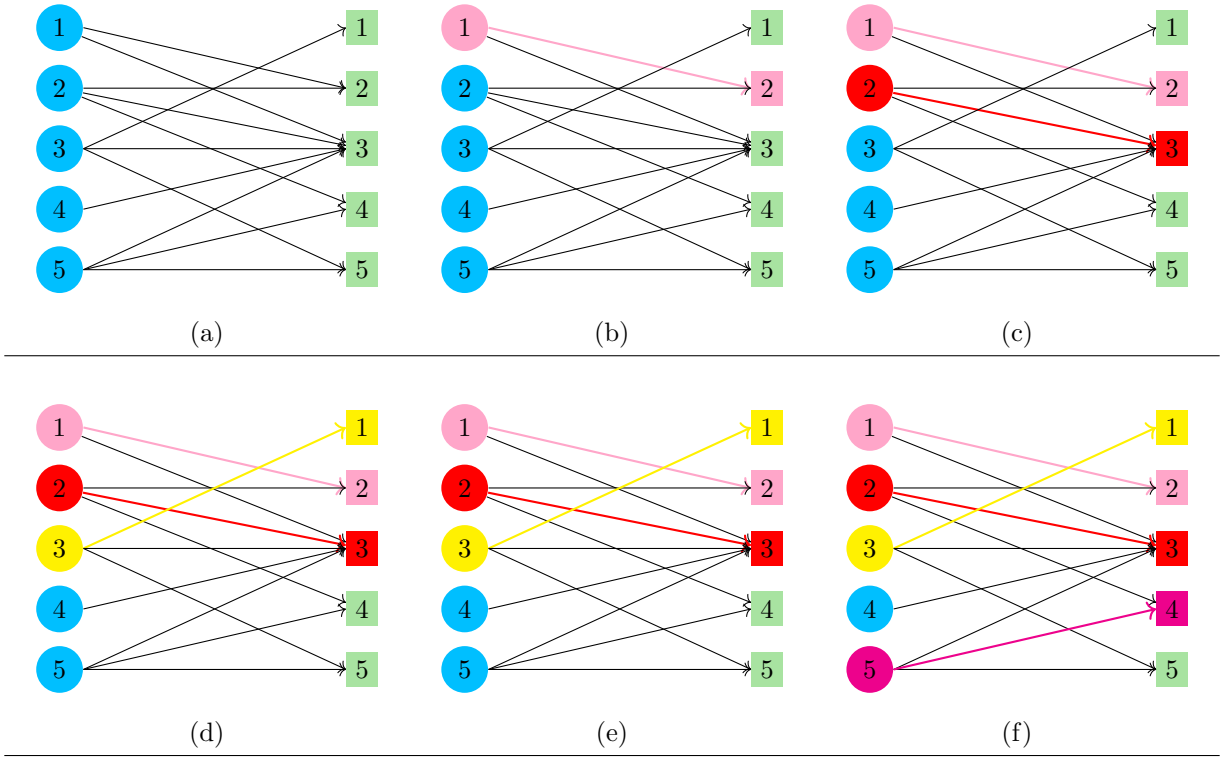
Given a bipartite graph $G = (A \cup B, E)$, *find an* $\{ S \subseteq A \times B : S$ *is a matching and is as large as possible.* $\}$ [1]

3 Scenario

In a picnic,there are 5 people and 5 food items.Some people express interest in some of the items.How can we satisfy **maximum number of people** while wasting **minimum number of items**?

4 Greedy Approach

Greedy approach seeks to find the **first available item** among the items in which the person is interested in and assigns that item to the person. Here's a simulation for the above scenario :-



a) We define the problem using a graph, where any blue vertex represents a person to whom no item has been assigned and any green vertex represents an available item. Edges from a person to an item depicts their interest in that item.
b) Person 1 initiates the process by selecting the first available item (item 2).
c) Person 2's only viable option is item 3, as item 2 has already been chosen.
d) Person 3 selects item 3.
e) Person 4 cannot be assigned an item, as all options previously chosen by them have been allocated.
f) Person 5 discovers that item 4 is available for selection.

Bipartite matching : $\{(1, 2), (2, 3), (3, 1), (5, 4)\}$
Although greedy provides us with a valid matching , it is not maximal.

5 Time Complexity

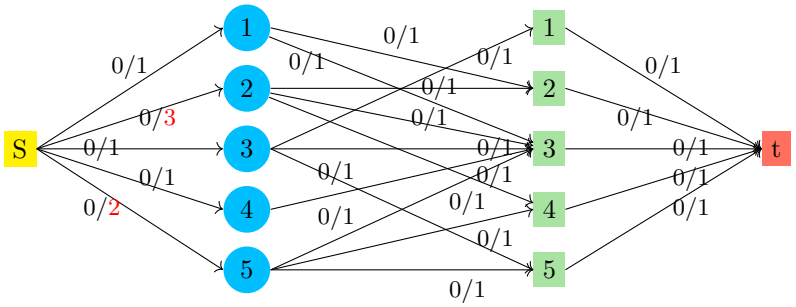
Edmond Karp's algorithm uses BFS as we need the shortest augmenting path from source to sink. Running time of BFS is $O(V + E)$. Path search can take upto $O(VE)$ time. Some calculation and reduction leads us to a time complexity of $O((V + E)VE) = O((E)VE) = O(VE^2)$.

6 Extension

Now some special cases will be discussed where the algorithm needs to be extended to solve the problem.

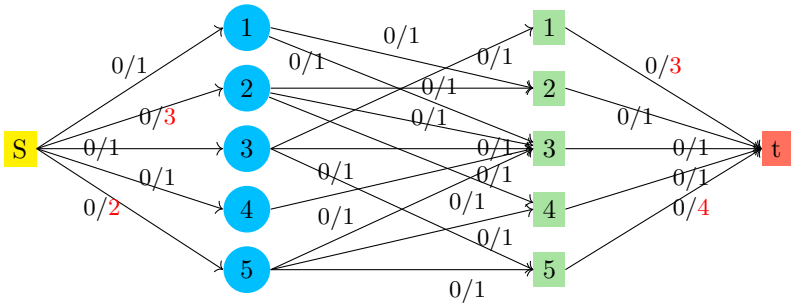
6.1 Allowing multiple items for a person

In this case we will make the capacity of the edges from source to person, equal to the number of items they can take. Then we can use the same algorithm to find the maximum matching.
For example if Person 2 is allowed to take 3 items and person 5 is allowed to take 2 items:



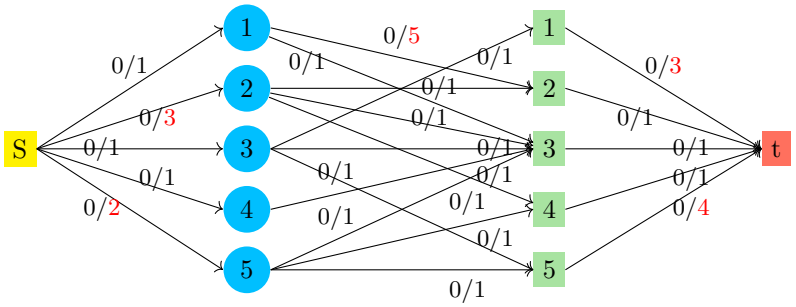
6.2 Multiple copies of item available

This case almost similar to the previous one. We will make the capacity of the edges from item to sink, equal to the number of copies of the item available. Then we can use the regular network flow algorithm to find the maximum matching.
For example if item 1 has 3 copies and item 5 has 4 copies:



6.3 Taking same item multiple times

We have to make the edges from person to item have capacity, the number of times that particular person can take that particular item. Then we can use our network flow algorithm to solve the problem.
For example if Person1 can take Item 2 upto 5 times:



7 References

References

[1] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.