Flows in Networks: Bipartite Matching

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

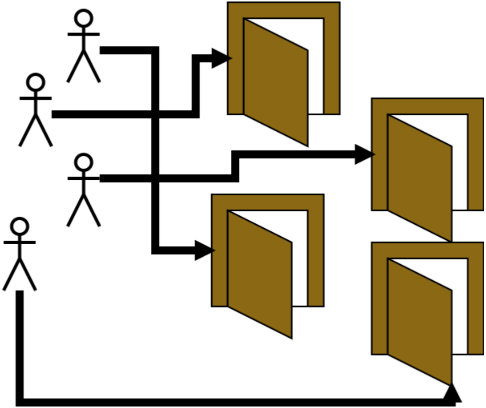
Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Discuss some problems that can be solved with bipartite matching.
- Understand the correspondence between bipartite matching problems and flow problems.
- Understand obstacles to finding large matchings.

Matching

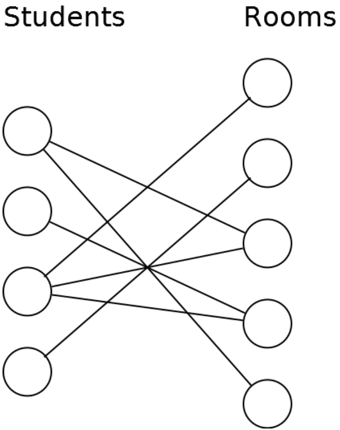
You are trying to coordinate housing in a college dormitory.



Matching

- Have n students and m rooms.
- Each student has a list of acceptable rooms.
- Want to place as many students as possible in an acceptable room.
- Cannot place more than one student in the same room.

Organizing Data

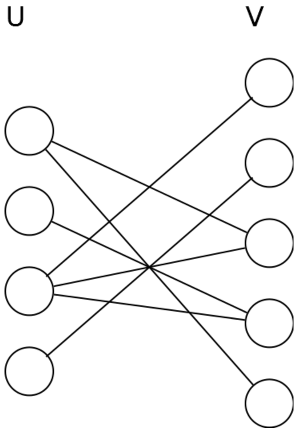


Bipartite Graph

Definition

A bipartite graph is a graph G whose vertex set is partitioned into two subsets, U and V, so that there all edges are between a vertex of U and a vertex of V.

Example

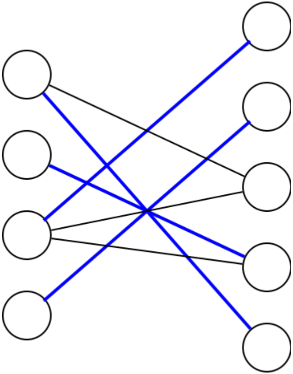


Matchings

Definition

Given a graph G, a matching on G is a collection of edges of G, no two of which share an endpoint.

Example



Bipartite Matching

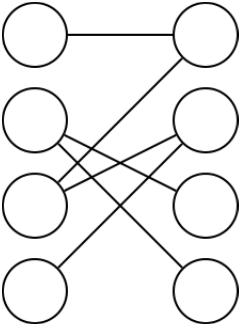
Bipartite Matching

Input: Bipartite graph G.

Output: A matching on G consisting of as many edges of possible (ideally pairing up all the vertices).

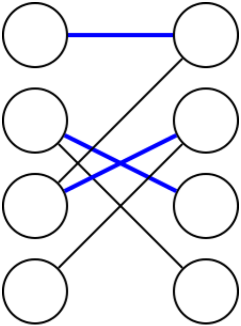
Problem

What is the size of the largest matching?



Solution

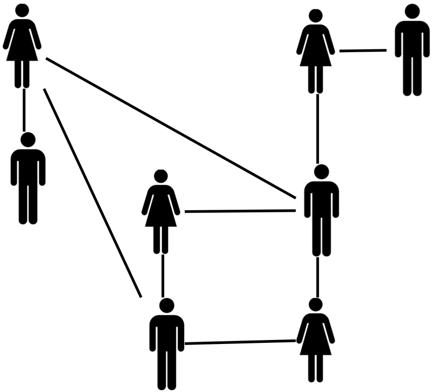
Best is 3.



Applications

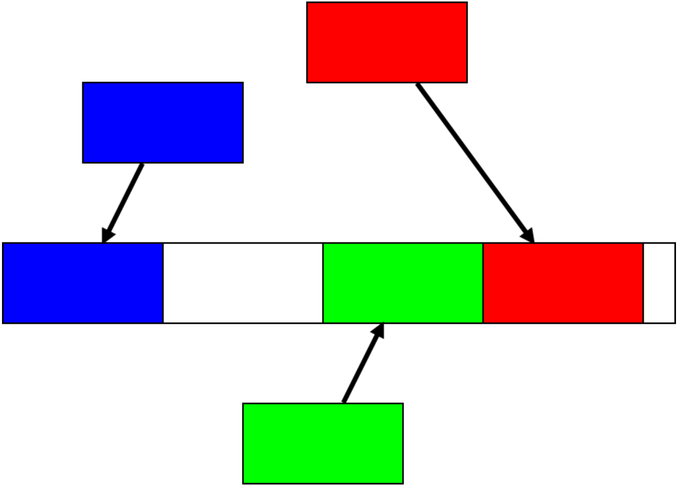
Bipartite matching has a number of applications.

Matchmaking



[Though if there are gay people, it becomes computationally much more complicated.]

Scheduling

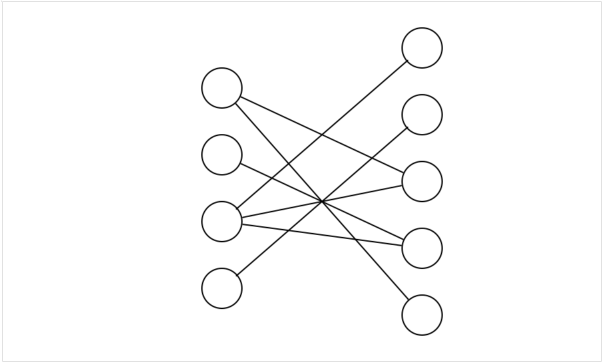


Connection to Flows

- Need to connect nodes on left to nodes on right without putting too many connections through any given node.
- Have source connect to left nodes.
- Have right nodes connect to sink.

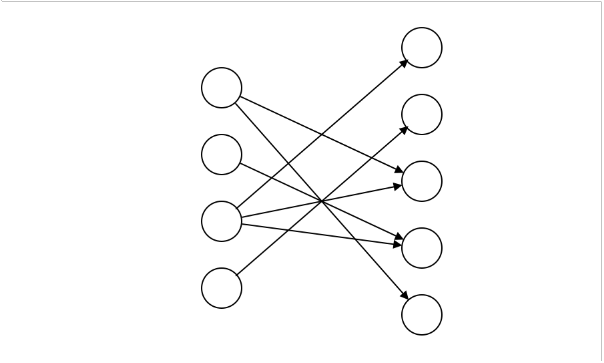
Convert to Network

Start with bipartite graph.



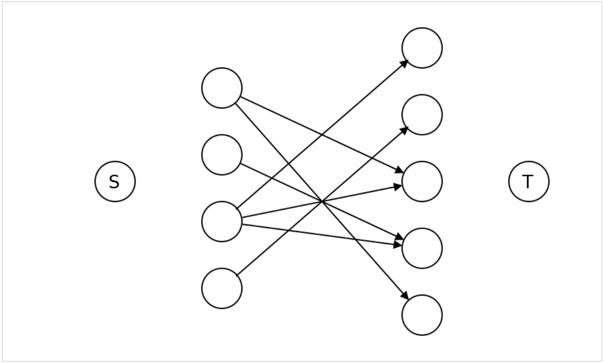
Convert to Network

Direct edges left to right.



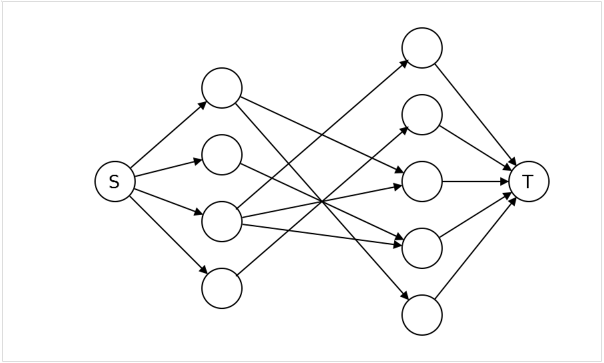
Convert to Network

Add source and sink.



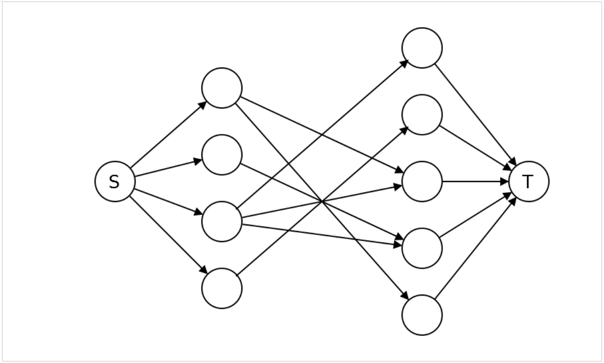
Convert to Network

Connect source/sink to vertices.



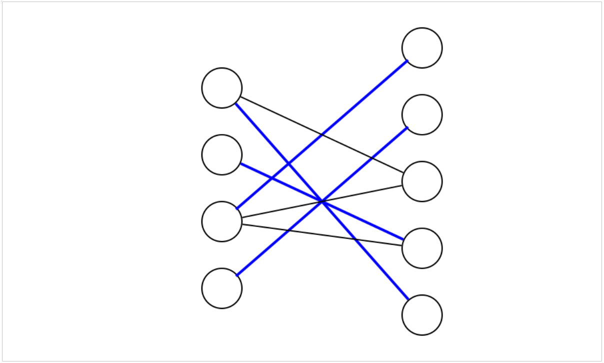
Convert to Network

All capacities are 1.



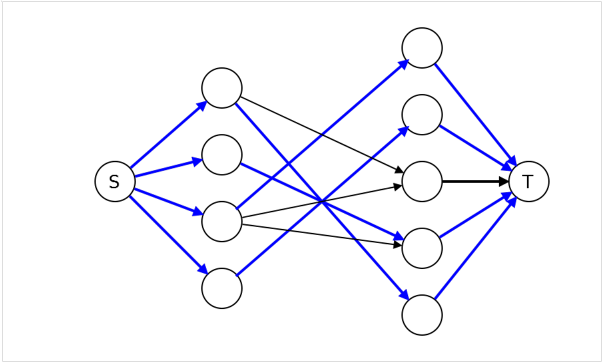
Correspondence

To every matching



Correspondence

To every matching we have a flow



Formally

Lemma

Let G be a bipartite graph and G' the corresponding network. There is a $1-1$ correspondence between matchings of G and integer-valued flows of G'.

Matching to Flow

- Run flow through each edge of the matching.
- Run flow from s to each utilized vertex of U.
- Run flow to t from each utilized vertex of V.

Flow to Matching

- Use middle edges with flow in them.
- Cannot have two with same vertex in U (not enough flow in).
- Cannot have two with same vertex in V (not enough flow out).

Algorithm

BipartiteMatching(G)

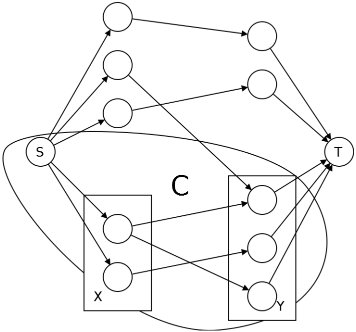
```
Construct corresponding network G′
Compute Maxflow(G′)
Find corresponding matching, M
return M
```

Maxflow-Mincut

We can also apply the Maxflow-Mincut theorem to the corresponding flow. This will tell us something useful about when we can find large matchings.

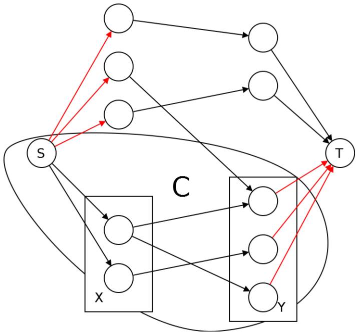
Mincut

- Let $X = \mathcal{C} \cap U$, $Y = \mathcal{C} \cap V$.
- Elements of V connecting to X in Y.



Cut Size

$$\text{Size} = |U \backslash X| + |Y|.$$



Another Viewpoint

- All edges of G connect to either Y or $U \backslash X$.
- Can bound matching size by finding such a set of vertices.

Kőnig's Theorem

Theorem

For G a bipartite graph, if k is the size of the maximal matching, there is a set $\mathcal{S}$ of k vertices so all edges of G are adjacent to $\mathcal{S}$.

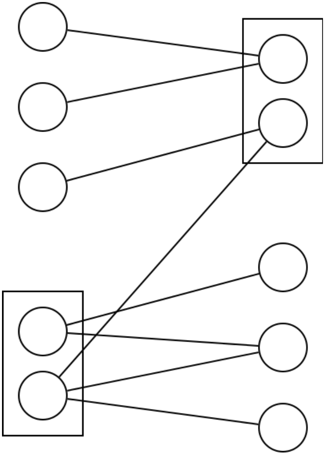
Kőnig's Theorem

Theorem

For G a bipartite graph, if k is the size of the maximal matching, there is a set $\mathcal{S}$ of k vertices so all edges of G are adjacent to $\mathcal{S}$.

Note that if you can find such an $\mathcal{S}$, the maximal matching has size at most $|\mathcal{S}|$.

Example



The Marriage Lemma

Theorem

Let G be a bipartite graph with n vertices on either side. Then there is a perfect pairing on G (a matching using all vertices) unless there is some set, $\mathcal{S}$ of m vertices of U, such that the total number vertices adjacent to a vertex in $\mathcal{S}$ is less than m.

Summary

- Solve maximum matching problems by reducing to flow problems.
- Maxflow-Mincut gives characterization of maximum matching.