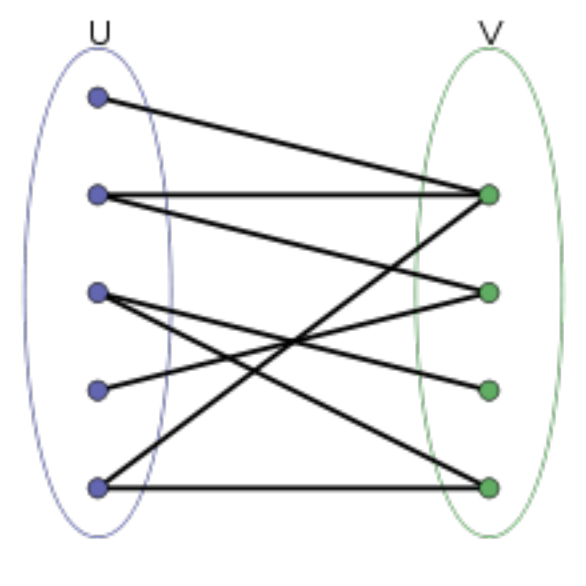
Advanced Computer Contest Preparation
Lecture 30

MAXIMUM BIPARTITE MATCHING

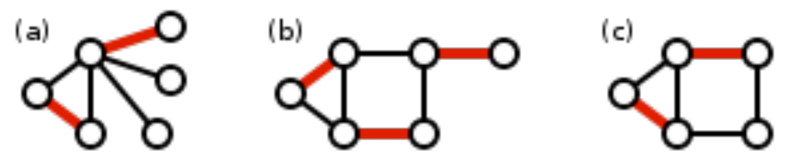
Bipartite Graphs

- A graph is **bipartite** if the nodes can be separated into two sets such that no edge connects two nodes in the same set



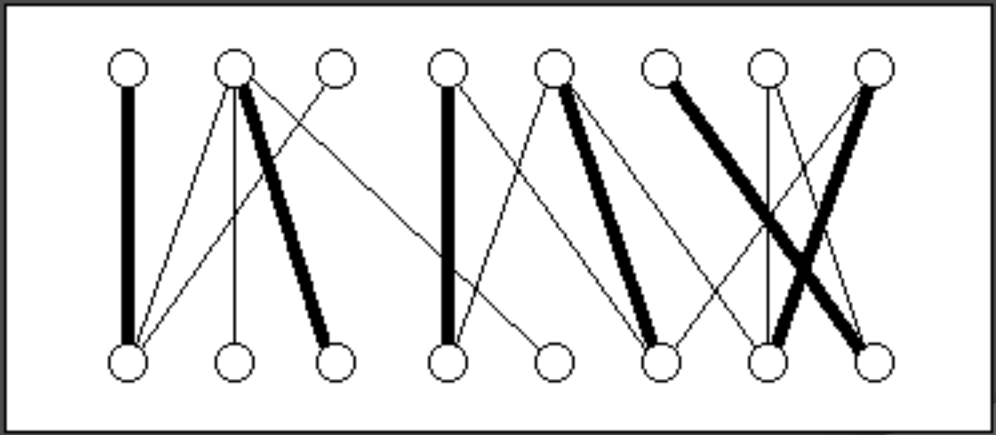
Matching

- A matching is a set of edges such that no two edges connect to the same node
- A matching is maximal if adding any edge not in currently in the set will invalidate this property
- A matching is maximum if it is a matching with the largest possible number of edges



Maximum Bipartite Matching

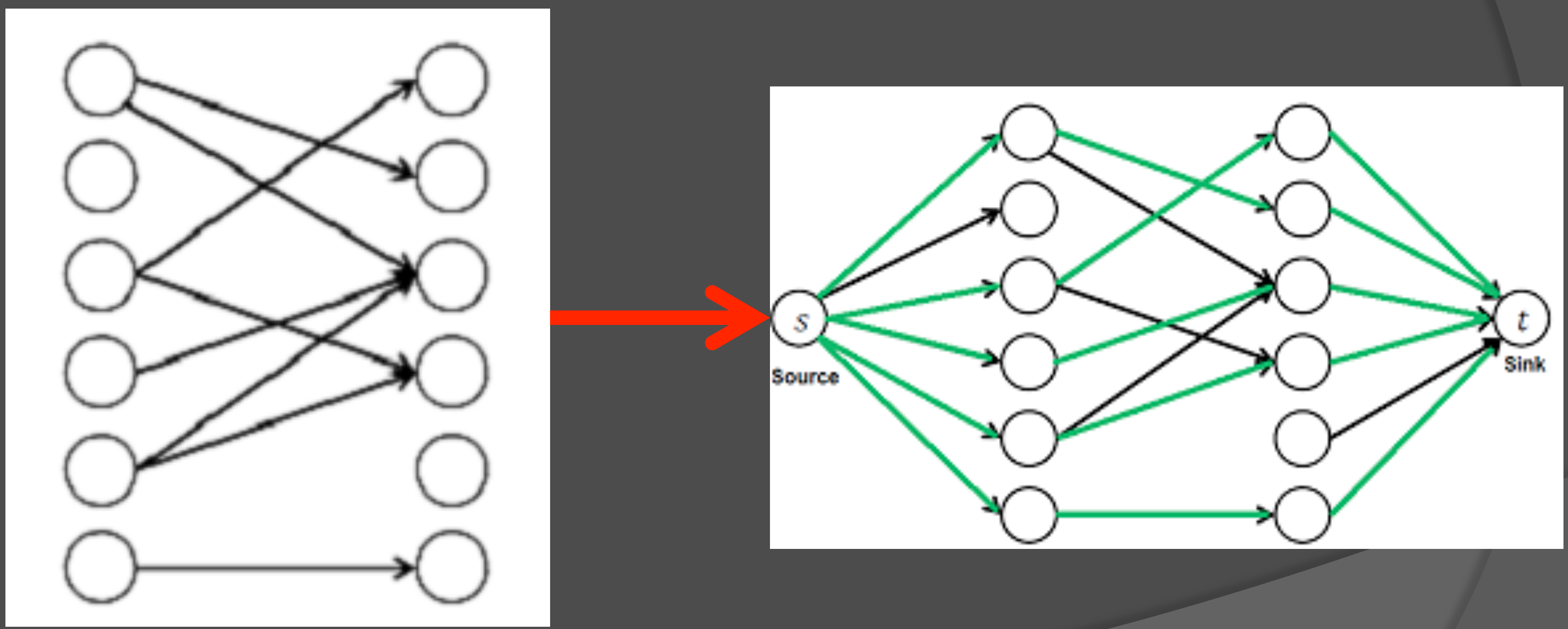
- Find a maximum matching of a bipartite graph



Problem Reduction

- Note that all nodes in the left set (u) can have at most one outgoing edge
- All nodes in the right set (v) can have at most one incoming edge
- Reduce this problem to a max flow problem:
  - Connect nodes in u to a source with edges of capacity 1
  - Connect nodes in v to a sink with edges of capacity 1
- The max flow value is the size of the maximum matching

Problem Reduction



Proof

- Let M be a matching in the bipartite graph, f be a flow in the modified graph
- If there is a matching of size $|M|$, there is a flow with that value
  - Each edge in the matching has 1 flow through it; the nodes it connects have 1 flow going from source/to sink (valid flow network)
- If there is an integral flow of $|f|$, there is a matching of size $|f|$
  - Since the capacity of each edge is 1, we can make it so that up to one edge can go in/out of a node (valid matching)
- Therefore, $max(|M|) = max(integer\ |f|)$
- Since the flow network only contains integer capacities, $max(|f|)$ is an integer
- Therefore, $max(|M|) = max(|f|)$

Analysis

- We can use the Ford–Fulkerson method
- Runtime: $O(Ef)$
- Due to the nature of the problem, the edge capacities are integers, and f is at most V
- Therefore, the runtime is $O(VE)$
- Simple to code, relatively fast runtime compared to other standard max flow algorithms

THANK YOU!