CPSC 490 – Problem Solving in Computer Science

Lecture 23: Maximum Bipartite Matching

Jason Chiu and Raunak Kumar

2017/03/20

University of British Columbia

Last Time

- Introduced the Maximum Flow problem
- Discussed the Ford-Fulkerson algorithm
- Discussed the Max Flow - Min Cut Theorem

Problem 1 - Edge-Disjoint Paths

- You are given a directed graph G.
- Find the number of edge disjoint paths from s to t.
- 2 paths P_1 and P_2 edge disjoint if they don't share any edges.

Problem 1 - Solution

Just set the capacity of every edge to 1!

Problem 2 - Vertex-Disjoint Paths

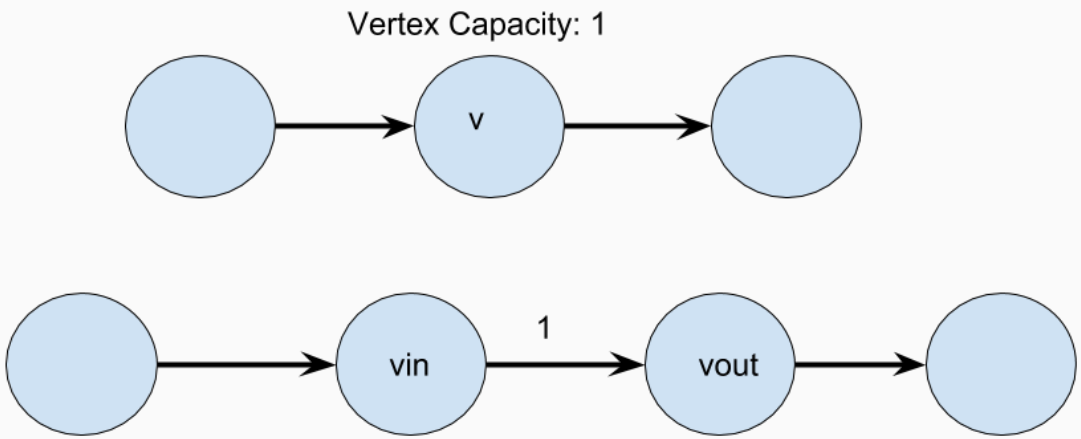
Find the number of vertex disjoint paths from s to t.

Problem 2 - Solution

- We don't want paths to share a vertex $\Rightarrow$ want vertex capacity 1
- But we only know how to deal with edge capacities...

Problem 2 - Solution

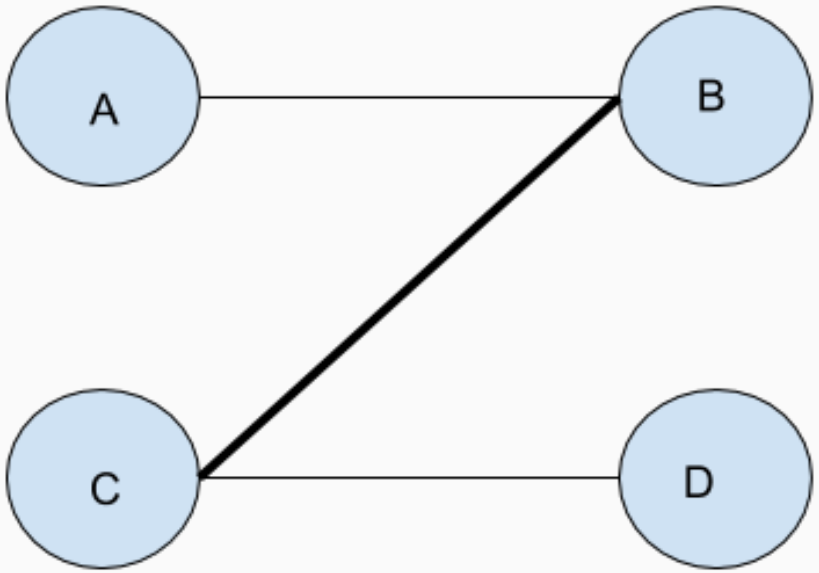
- Split each vertex v into v_{in} and v_{out}.
- Add an edge $v_{in} \rightarrow v_{out}$ with capacity 1.
- Change every edge $u \rightarrow v$ to $u_{out} \rightarrow v_{in}$



Maximum Bipartite Matching

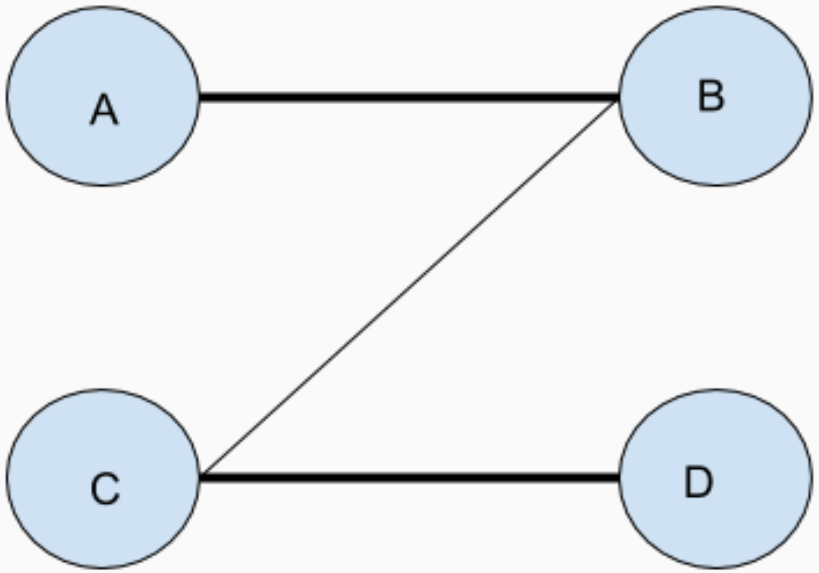
- A graph $G = (V, E)$ is bipartite if V can be partitioned into disjoint sets A and B such that all edges goes from A to B.
- A matching is a subset of edges $M \subset E$ such that no two edges touch same vertex
- Goal: find the maximum size matching in a bipartite graph.

Maximum Bipartite Matching



Size of the matching is 1.

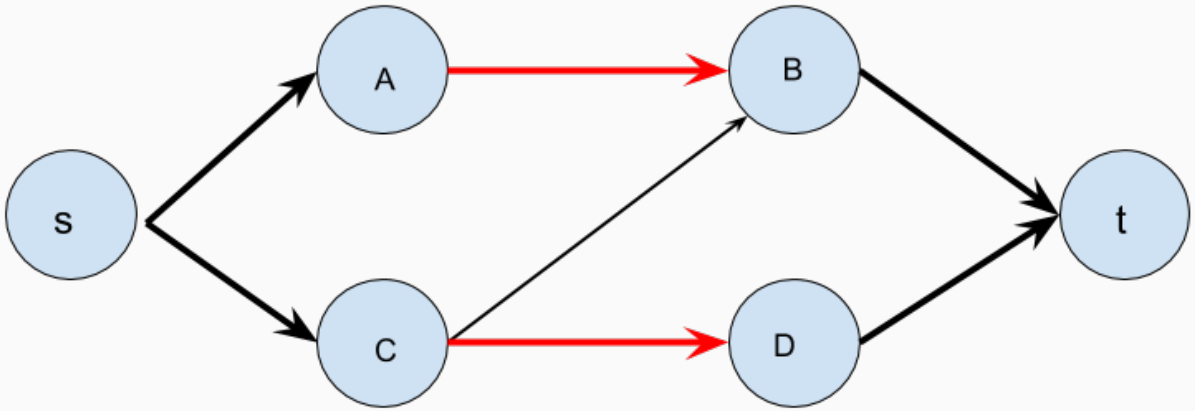
Maximum Bipartite Matching



Size of the maximum matching is 2.

Maximum Bipartite Matching

We can use maximum flow to solve this problem!



All edge capacities = 1.
Edges with flow = 1 are in the matching

Assume m vertices on one side, n on other side, $n \leq m$, then time is

Ford Fulkerson: $O(Ef) = O(mn^2)$

Dinic's: $O(mn\sqrt{n})$ (also known as Hopcroft–Karp in this case)

Problem 3 - Maximum Bipartite Matching

- There are m jobs and n applicants.
- Each applicant applies to a subset of the jobs.
- Each job can accept only 1 applicant
- Each applicant can only work at 1 job.
- What is the maximum number of applicants who get a job?

Problem 3 - Solution

Idea: match applicants to jobs!

- Construct a bipartite graph with vertices for applicants and jobs.
- Add edge $i \to j$ if applicant i applied for job j
- Find the maximum matching!
- Time Complexity: $O(nm(n+m))$.

Problem 4 - Match What to What?

$M \in \{0,1\}^{n \times n}$ is rearrangable if you can make all the diagonal entries 1 by swapping rows and columns. Determine if M is rearrangable.

1	1	1
1	0	0
1	0	0

Not rearrangable

1	1	1
1	1	0
1	0	0

Rearrangable

Problem 4 - Solution

Observation: want to pick n 1's such that each row has exactly a single 1, each column has exactly a single 1

$\Rightarrow$ match rows to column $\Rightarrow$ Bipartite Matching!

- Create a bipartite graph where left vertex = row, right vertex = col.
- Add edge $i \to j$ if $M[i][j] = 1$.
- Determine if there is a matching of size n.

Problem 4 - Solution

Observation: want to pick n 1's such that each row has exactly a single 1, each column has exactly a single 1

$\Rightarrow$ match rows to column $\Rightarrow$ Bipartite Matching!

- Create a bipartite graph where left vertex = row, right vertex = col.
- Add edge $i \rightarrow j$ if $M[i][j] = 1$.
- Determine if there is a matching of size n.
- If row i matches column c_i, then we can swap column i and c_i and rearrange the matrix.
- Time Complexity: $O(n^3)$

Problem 5 - Match What to What? (Take 2)

- There are n airports and m flights.

Problem 5 - Match What to What? (Take 2)

- There are n airports and m flights.
- It takes t_{ij} time to fly from airport i to j.
- Each flight has a departure time.

Problem 5 - Match What to What? (Take 2)

- There are n airports and m flights.
- It takes t_{ij} time to fly from airport i to j.
- Each flight has a departure time.
- What is the minimum number of planes that we need to buy? We can add unscheduled flights to move planes around.

Hint: think of each flight as a vertex, what are the edges?

Problem 5 - Solution

Idea: match a flight to its “next flight” so construct bipartite graph

- Each flight is a vertex on both left and right side
- What are the edges?

Problem 5 - Solution

Idea: match a flight to its "next flight" so construct bipartite graph

- Each flight is a vertex on both left and right side
- What are the edges?
  - $i \to j$ if flight i can finish it's trip, take the shortest path from $i.destination$ to $j.source$ in time for departure at $j.source$
  - Use Floyd-Warshall to figure out shortest path!
  - This is where we may need to add an unscheduled flight.

Then we find maximum matching M: what does this tell us?

- Matching $\Leftrightarrow$ set of valid disjoint flight sequences
- Unmatched on left side $\Rightarrow$ last flight of a plane
- $\Rightarrow$ Answer is # unmatched vertices $= m - |M|$

Time Complexity: $O(n^3 + m^3)$

Problem 6 - Is It A Matching?

- We are all supporters of a basketball team, Team 490.
- There are n teams in the league, and we know the number of wins and losses for each team.

Problem 6 - Is It A Matching?

- We are all supporters of a basketball team, Team 490.
- There are n teams in the league, and we know the number of wins and losses for each team.
- We know the list of the G games remaining to be played.
- The team(s) with the highest number of wins at the end win.
- Is it possible for Team 490 to emerge as a winner?

Problem 6 - Is It A Matching?

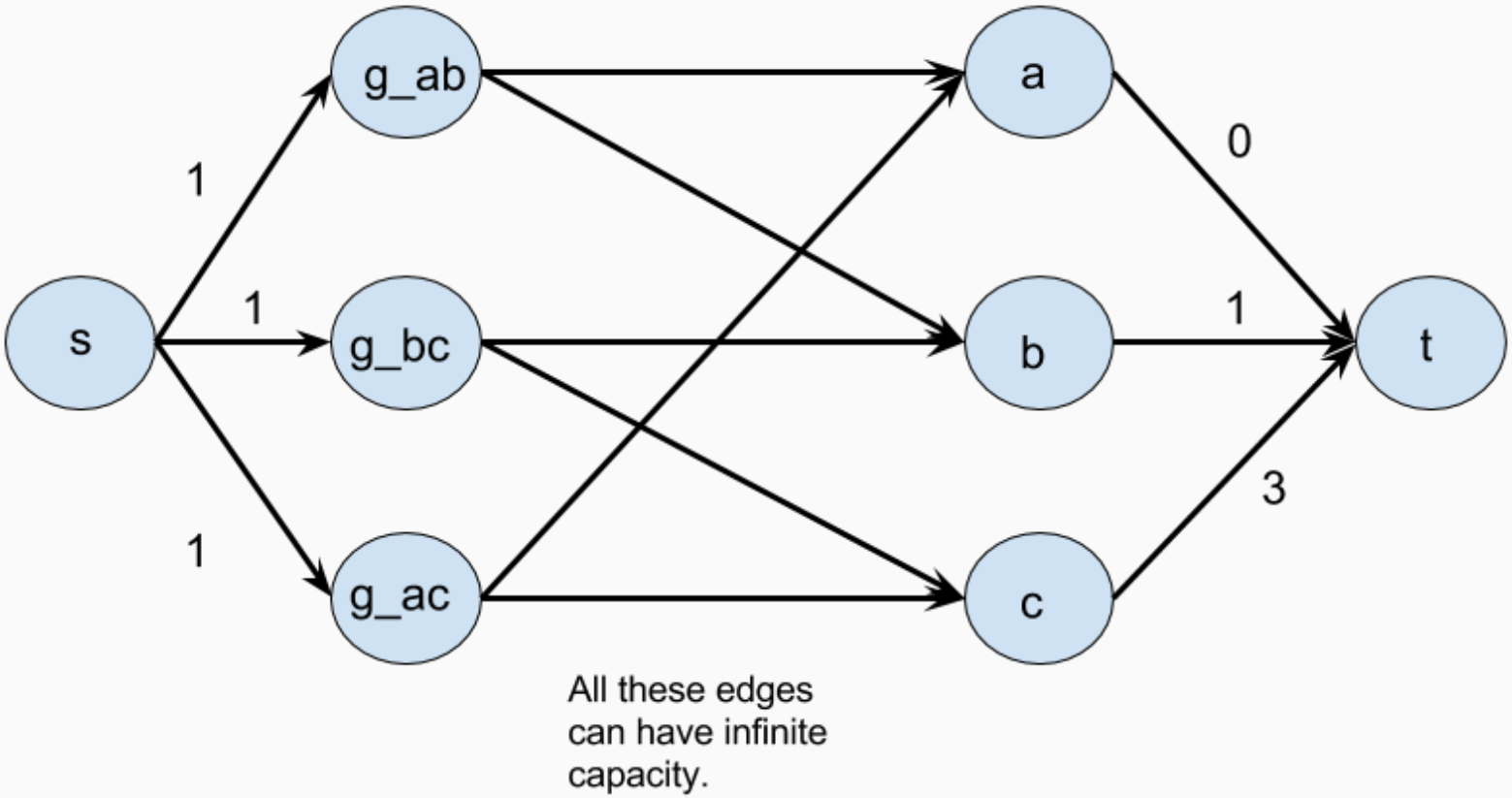
Team	Number of Wins
490	2
A	3
B	2
C	0

Games to be played
490 - A
A - B
B - C
C - A

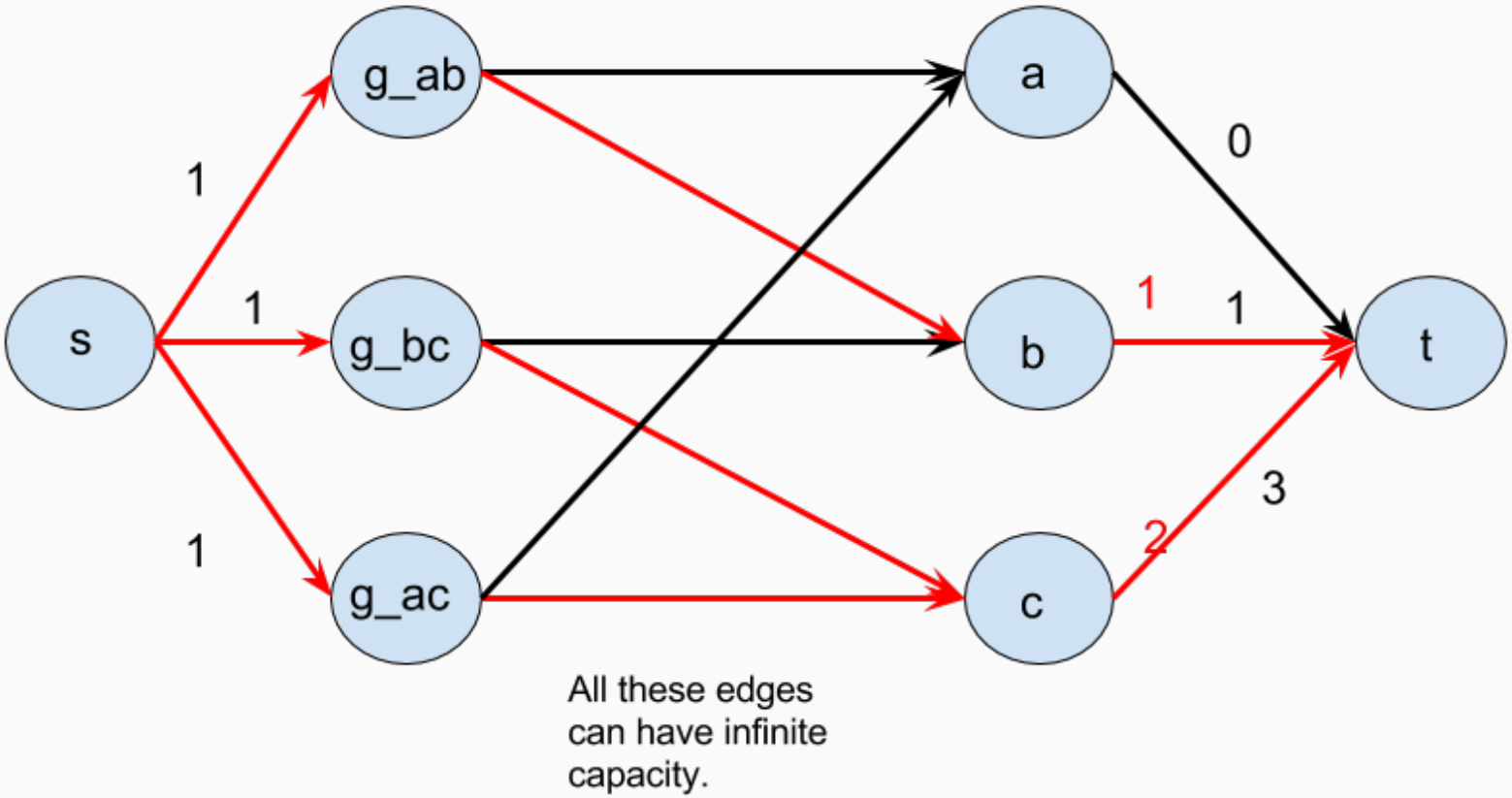
Can team 490 be a winner in this scenario? Yes!

- Team 490 wins its remaining 1 game to end up with 3 wins.
- *B* wins its match with *A*, and *C* wins its two matches.

Problem 6 - Solution



Problem 6 - Solution



The flows represent the winners of the games.

Problem 6 - Solution

- Let Team 490 win all its remaining games to get total of w wins.
- If $w < w_i$ for some team i, then it's not possible.
- Otherwise, use flow!

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.
- Construct a bipartite graph with
  - Left side: the remaining games g_{ij} not involving 490
  - Right side: the teams i

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.
- Construct a bipartite graph with
  - Left side: the remaining games g_{ij} not involving 490
  - Right side: the teams i
- Add edge source $\to g_{ij}$ with capacity 1.

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.
- Construct a bipartite graph with
  - Left side: the remaining games g_{ij} not involving 490
  - Right side: the teams i
- Add edge source $\to g_{ij}$ with capacity 1.
- Add edges $g_{ij} \to i$ and $g_{ij} \to j$ with capacity $= \infty$

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.
- Construct a bipartite graph with
  - Left side: the remaining games g_{ij} not involving 490
  - Right side: the teams i
- Add edge source $\to g_{ij}$ with capacity 1.
- Add edges $g_{ij} \to i$ and $g_{ij} \to j$ with capacity $= \infty$
- Add edge $i \to$ sink with capacity $= w - w_i$.

Problem 6 - Solution

Idea: match games to winners, but upper bound how many games each team can win

- Let g_{ij} denote the game being played between teams i and j.
- Construct a bipartite graph with
  - Left side: the remaining games g_{ij} not involving 490
  - Right side: the teams i
- Add edge source $\to g_{ij}$ with capacity 1.
- Add edges $g_{ij} \to i$ and $g_{ij} \to j$ with capacity $= \infty$
- Add edge $i \to$ sink with capacity $= w - w_i$.
- Possible for Team 490 to win $\Leftrightarrow$ flow $=$ # games remain

Next Class

Konig's Theorem