

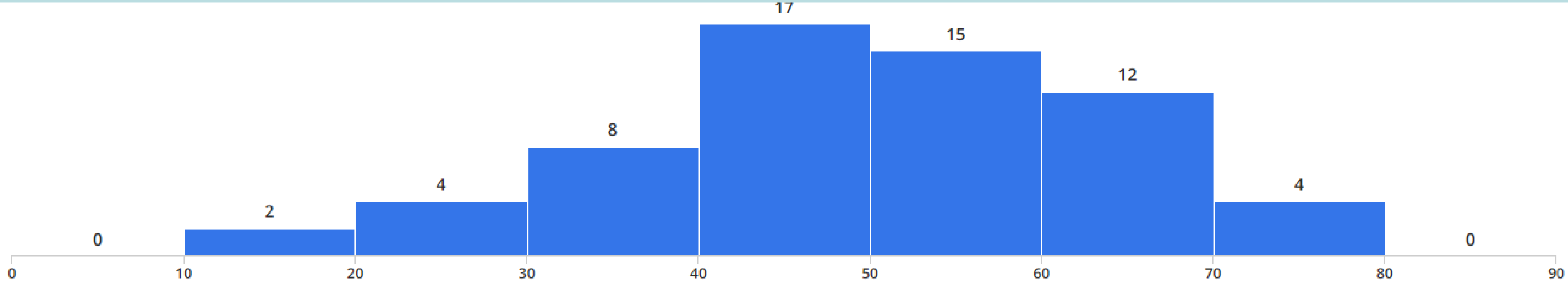
CSE 421
Introduction to Algorithms

Lecture 19

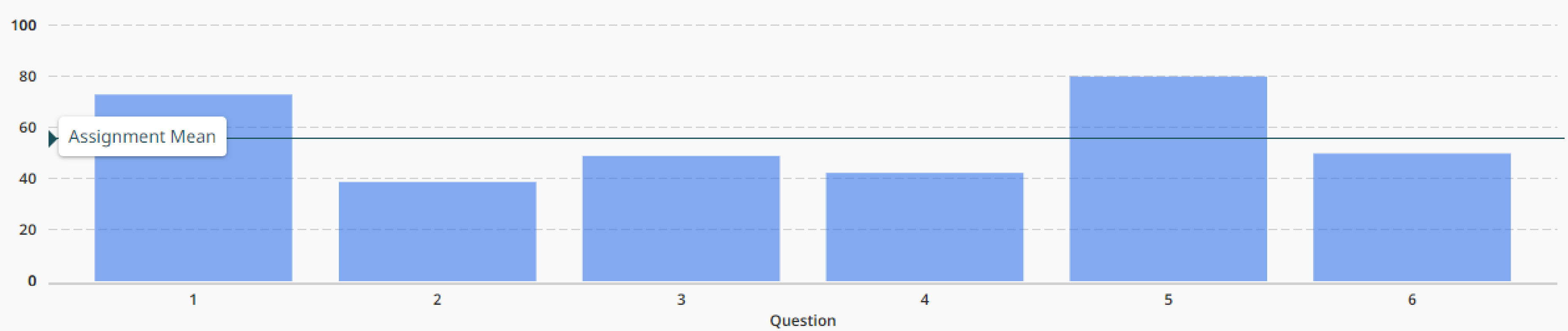
Winter 2024

Network Flow, Part 3

Midterm



Minimum	Median	Maximum	Mean	Std Dev
16.5	49.75	77.0	49.98	13.96



Outline

- ~~Network flow definitions~~
- ~~Flow examples~~
- ~~Augmenting Paths~~
- ~~Residual Graph~~
- ~~Ford Fulkerson Algorithm~~
- ~~Cuts~~
- ~~Maxflow-MinCut Theorem~~
- ~~Worst Case Runtime for FF~~
- Improving Runtime bounds
  - Capacity Scaling
  - Fully Polynomial Time Algorithms
- Applications of Network Flow

Ford-Fulkerson Algorithm (1956)

while not done

Construct residual graph G_R

Find an s-t path P in G_R with capacity $b > 0$

Add b units of flow along path P in G

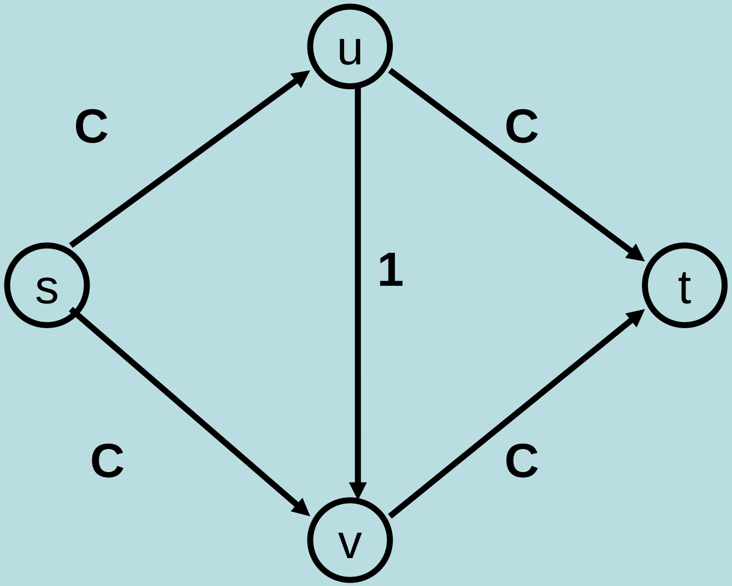
Ford Fulkerson Runtime

- Cost per phase   X   number of phases

- Phases
  - Capacity leaving source: C
  - Add at least one unit per phase
- Cost per phase
  - Build residual graph: O(m)
  - Find s-t path in residual: O(m)

Performance

- The worst case performance of the Ford-Fulkerson algorithm O(Cm)



Polynomial Time Algorithms

- Input of size n, runtime $T(n) = O(n^k)$
- Input size measures
  - Bits of input
  - Number of data items
- Maximum item magnitude C
  - $O(Cn^k)$: Exponential
  - $O(n^k \log C)$: Polynomial
  - $O(n^k)$: Fully polynomial

Better methods of finding augmenting paths

- Find the maximum capacity augmenting path
  – $O(m^2 \log(C))$ time algorithm for network flow
- Find the shortest augmenting path
  – $O(m^2 n)$ time algorithm for network flow
- Find a blocking flow in the residual graph
  – $O(mn \log n)$ time algorithm for network flow

Capacity Scaling Algorithm

- Choose $\Delta = 2^k$ such that all edges in G_R have capacity less than 2Δ

while $\Delta \geq 1$

while there is a path P in G_R with capacity Δ

Add Δ units of flow along path P in G

Update G_R

$\Delta = \Delta / 2$

Edmonds-Karp: Easier analysis than Max Capacity First

Analysis

- If capacities are integers, then graph is disconnected when $\Delta = ½$
- If largest edge capacity is C, then there are at most log C outer phases
- At the start of each outer phase, the flow is within $2m\Delta$ of the maximum
  - So there are at most 2m inner phases for each Δ

Shortest Augmenting Path

- Find augmenting paths by BFS

for k = 1 to n

  while there is a path P in G_R of length k and capacity b > 0

    Add b units of flow along path P in G

    Update G_R

- Need to show:
  - The length of the shortest augmenting path is non-decreasing
  - Each while loop finds at most m paths

Analysis

- Augmenting along shortest path from s to t does not decrease distance from s to t

Analysis

- The distance from s to t must increase in G_R after m augmentations by shortest paths

Improving the shortest augmenting path algorithm

- Find a blocking flow in one phase to increase the length of augmenting paths
  - Dinitz (Ефим Абрамович Диниц) Algorithm
  - $O(n^2m)$
- Dynamic Trees to decrease cost per augmentation
  - $O(nm \log n)$

APPLICATIONS OF NETWORK FLOW

Problem Reduction

- Reduce Problem A to Problem B
  - Convert an instance of Problem A to an instance of Problem B
  - Use a solution of Problem B to get a solution to Problem A
- Practical
  - Use a program for Problem B to solve Problem A
- Theoretical
  - Show that Problem B is at least as hard as Problem A

Problem Reduction Examples

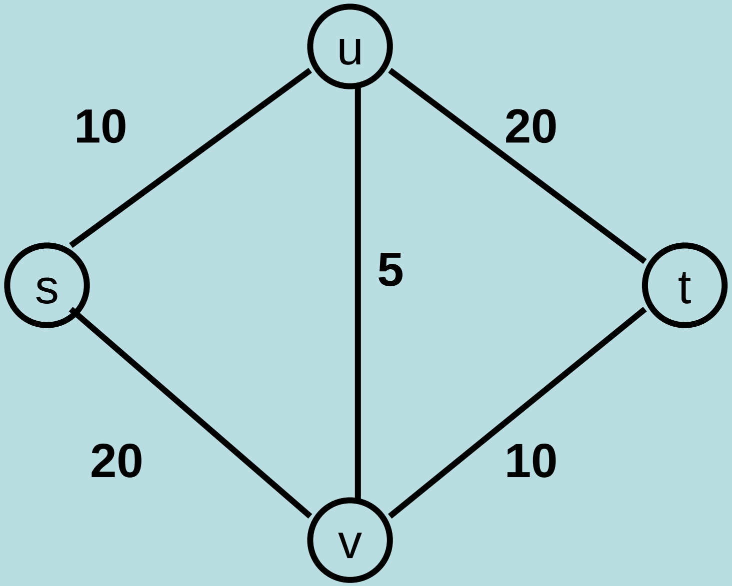
- Reduce the problem of finding the Maximum of a set of integers to finding the Minimum of a set of integers

Find the maximum of: 8, -3, 2, 12, 1, -6

Construct an equivalent minimization problem

Undirected Network Flow

- Undirected graph with edge capacities
- Flow may go either direction along the edges (subject to the capacity constraints)



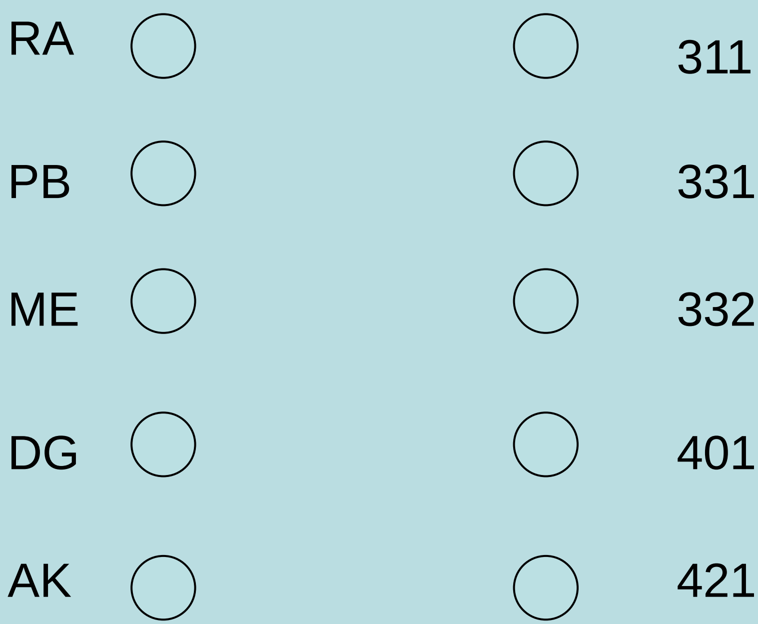
Construct an equivalent flow problem

Bipartite Matching

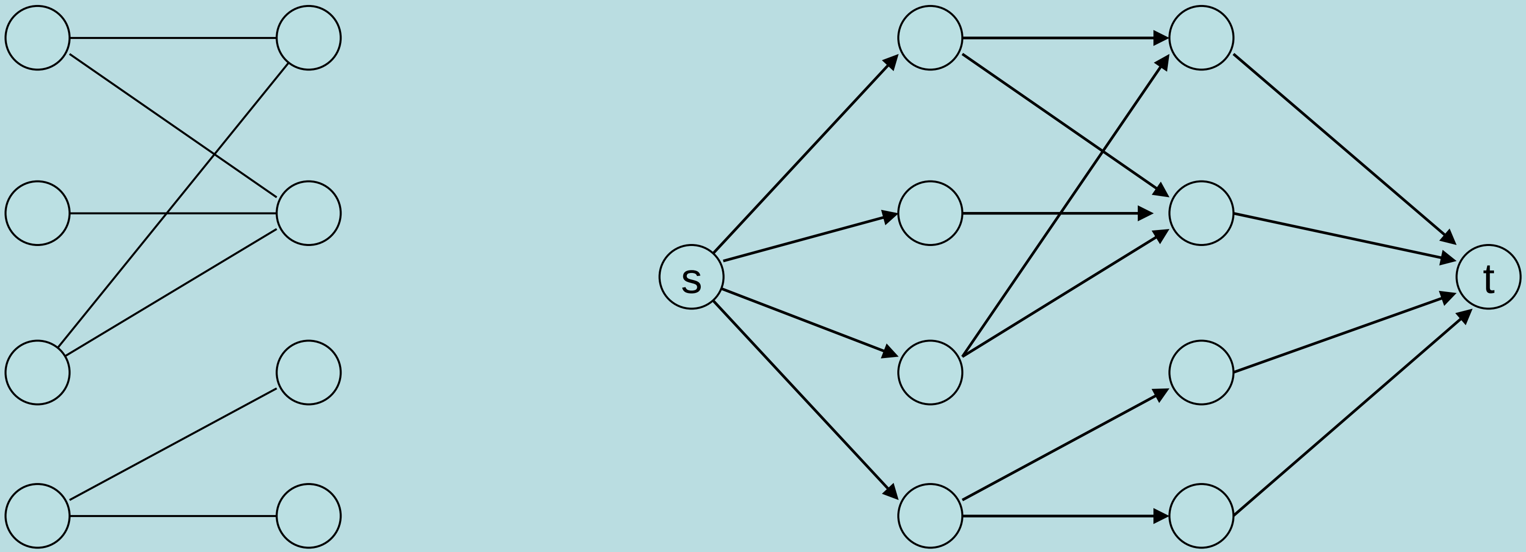
- A graph G=(V,E) is bipartite if the vertices can be partitioned into disjoints sets X,Y
- A matching M is a subset of the edges that does not share any vertices
- Find a matching as large as possible

Application

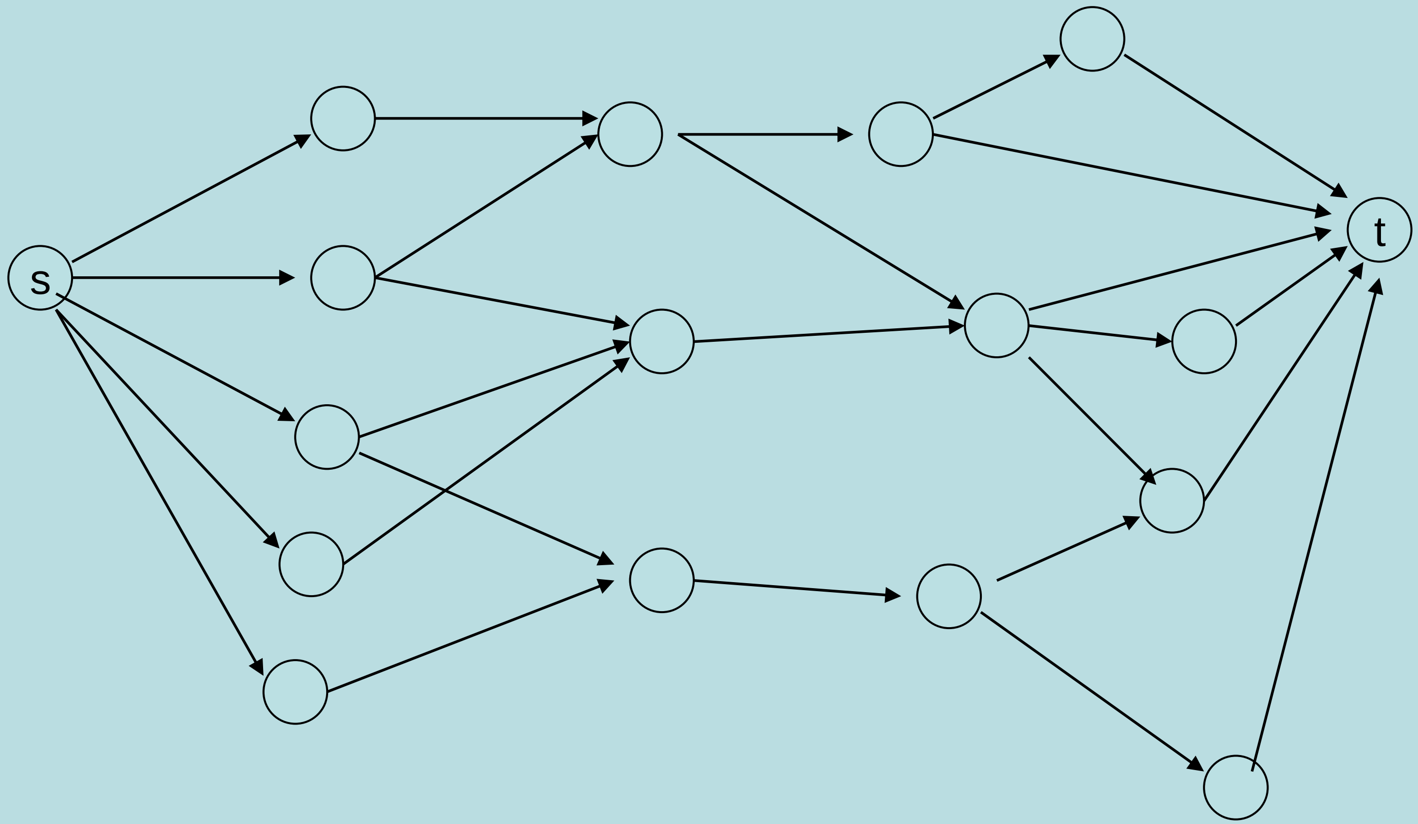
- A collection of teachers
- A collection of courses
- And a graph showing which teachers can teach which courses



Converting Matching to Network Flow



Finding edge disjoint paths



Construct a maximum cardinality set of edge disjoint paths

Multi-source network flow

- Multi-source network flow
  - Sources s_1, s_2, ..., s_k
  - Sinks t_1, t_2, ... , t_j
- Solve with Single source network flow

Resource Allocation: Assignment of reviewers

- A set of papers $P_1, . . ., P_n$
- A set of reviewers $R_1, . . ., R_m$
- Paper P_i requires A_i reviewers
- Reviewer R_j can review B_j papers
- For each reviewer R_j, there is a list of paper $L_{j1}, . . ., L_{jk}$ that R_j is qualified to review

Resource Allocation: Illegal Campaign Donations

- Candidates C_i, . . ., C_n
  - Donate b_i to C_i
- With a little help from your friends
  - Friends F_1, . . ., F_m
  - F_i can give a_{ij} to candidate C_j
  - You can give at most M_i to F_i