# CS122 Algorithms and Data Structures

MW 11:00 am - 12:15 pm, MSEC 101

Instructor: Xiao Qin

Lecture 18: Graphs (5)

1

## Network Flow Problems

- Consider the graph *G* to be a network, and the costs on edges to be "flow capacities".
- A network is a directed graph with two special vertices: a source *s* and a sink *t*.
- The flow through an edge *e* cannot be greater than its capacity
- Total flow coming to a vertex *v* must equal the flow going out.
- Maximum flow problem: find the maximum amount of flow that can pass from *s* to *t*.

2

## The Ford-Fulkerson Method

- It is a "method" rather than an "algorithm"
- Two important ideas:
  – Residual networks
  – Augmenting paths
- We use three graphs, the original graph *G*, a flow graph $G_f$ and a residual graph $G_r = G - G_f$ .

3

## The Ford-Fulkerson Method (cont.)

- Give an initial flow: f(u,v)=0 for all u,v $\in$ V
- We proceed in stages. Each stage we choose a augmenting path in the residual graph $G_r$ from *s* to *t*. The minimum edge on this path is the amount of flow that can be added to every edge on that path.
- We do this by adjusting $G_f$ and recomputing $G_r$ .
- We continue until there are no paths from *s* to *t*.
- We can't follow any edges that have capacity 0.

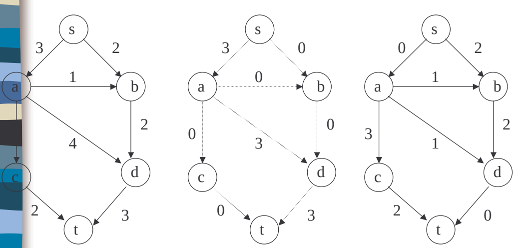
4

## Discussion

- We can obtained the maximum flow. However, how did I choose the paths?
- If we choose a greedy algorithm, we'd be tempted to choose paths that allows the maximum amount of flow to be added at each step. This might not work.
- For example, let's choose (s, a, d, t) first, because this allows 3 units of flow to be added.

5

## Example




There are no paths from s to t in the residual graph, so we are done, but we have not obtained the maximum possible flow.

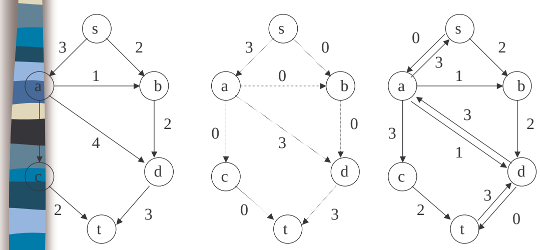
6

## A Better Algorithm

- We can make the algorithm work by allowing the algorithm to change its mind.
- In effect we allow the algorithm to undo its decisions by sending flow back in the opposite direction. This is best seen by example. We have to modify the residual graph.

7

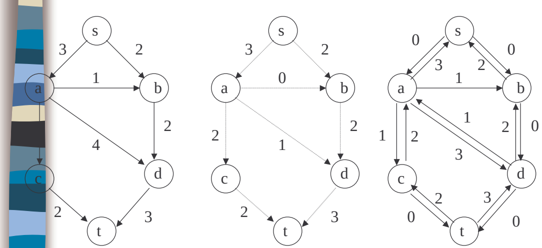
## Example



We again choose the path (s, a, d, t). But note we now allow the flow to backup in the residual graph.

8

## Example



We can now follow the path (s, b, d, a, c, t). The minimum flow along this path is 2. Note the flow from d to a is now 1 = 3 – 2. We are done. This is the maximum flow solution.

9

## Networks with multiple sources and sinks

- A maximum-flow problem has several sources and sinks
- Example: A company has a set of *m* factories and *n* warehouse.
- How to determine a maximum flow in such a network?

10

## Networks with multiple sources and sinks (cont.)

- Reduce the problem to an ordinary maximum-flow problem
- A flow network has m sources $S = \{s_1, s_2, \ldots, s_m\}$, and n sinks $T = \{t_1, t_2, \ldots, t_m\}$
- Add a supersource s and a directed edge $(s, s_i)$ with capacity cap(s, si) = $\infty$
- Add a supersink t and a directed edge $(t_i, t)$ with capacity $cap(t_i, t) = \infty$

11

## Matching

- What is a matching an undirected graph?
- Matched and unmatched vertices
- Finding a maximum matching in a graph.
- The maximum bipartite matching problem

12

## Finding a Maximum Bipartite Matching

- Use the Ford-Fulkerson method
- Construct a flow network in which flows correspond to matchings

13

## Construct a Flow Network

- Given an undirected bipartite graph $G = (V, E)$, define the corresponding flow network $G' = (V', E')$
- $V' = V \cup \{s, t\}$
- $E' = \{(u,v): u \in L, v \in R, (u,v) \in E\} \cup \{(s,u): u \in L\} \cup \{(v,t): v \in R\}$
- Assign unit capacity to each edge in $E'$

14

## Finding a Maximum Bipartite Matching

- A maximum matching in G corresponds to a maximum flow in its corresponding flow network G'
- Run the Ford-Fulkerson algorithm on G'

15