CS122 Algorithms and Data Structures

MW 11:00 am - 12:15 pm, MSEC 101
Instructor: Xiao Qin
Lecture 17: Graphs (4)

1

Topological Sort

- Ordering of vertices in a directed acyclic graph(DAG)
- If there is a path from u to v, then v appears after u in the ordering
- Edge (v, w) indicates that activity v must be completed before w
- Course prerequisites
- Topological order of courses is any sequence that does not violate the prereq requirements

2

Algorithm

- Find any vertex v with no incoming edges
- Print and logically remove it
- Apply to the rest of the graph
- Do this by
- Compute the indegrees of all vertices
- Logically remove means that we lower the count of incoming edges for each vertex adjacent to v

3

Algorithm (cont.)

- choose any vertex with indegree 0 as next
- All acyclic graphs have a topological order
- This algorithm will find one
- If there are no 0 nodes, there must be a cycle

4

Alg analysis

- find next vertex is O(|V|)
- since there are O(|V|) such calls
- algorithm is O(|V|*|V|)
- Can lower to O(|V|+|E|) if we keep a structure of all nodes which have degree 0

5

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*.

6

Maximum Flow Algorithm

- We use three graphs, the original graph G, a flow graph G_f and a residual graph $G_r = G - G_f$.
- We proceed in stages. Each stage we choose a path in 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.

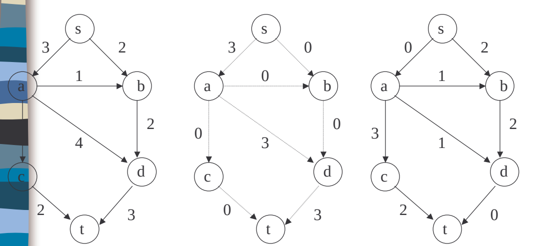
7

Discussion

- In the prior example we in fact 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.

8

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.

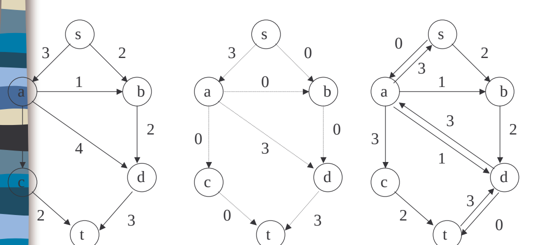
9

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.

10

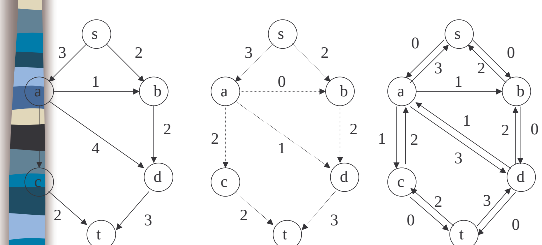
Example



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

11

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.

12

Conclusions

- This better greedy algorithm will always find the maximum flow solution if the edge capacities are rational numbers.
- Although we have used an acyclic graph in the example, the algorithm works on arbitrary graphs!

13