CS180 Discussion

Week 9

Lecture Recap

- Edge disjoint paths
  - Menger's Theorem
- Supply and demand (already seen)
- NP Completeness
- Reductions

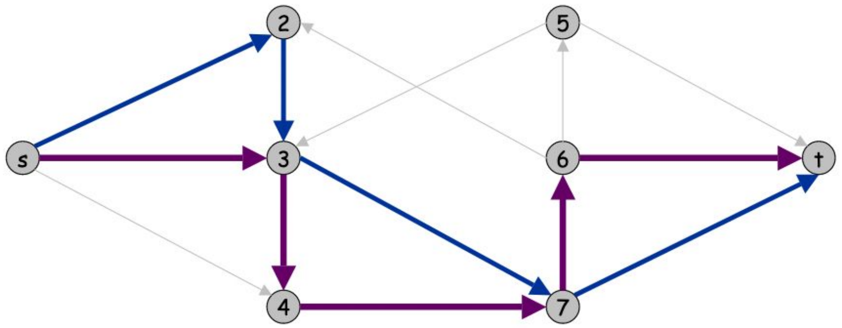
Final is on March 17 11:30am-2:30pm

Edge Disjoint Paths

Disjoint path problem. Given a digraph G = (V, E) and two nodes s and t, find the max number of edge-disjoint s-t paths.

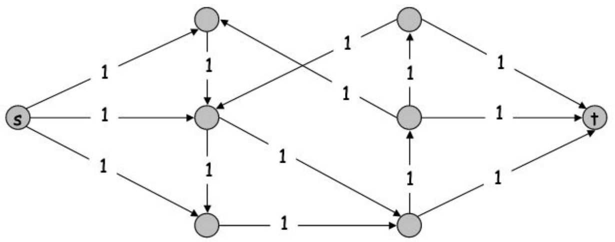
Def. Two paths are edge-disjoint if they have no edge in common.

Ex: communication networks.



Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.

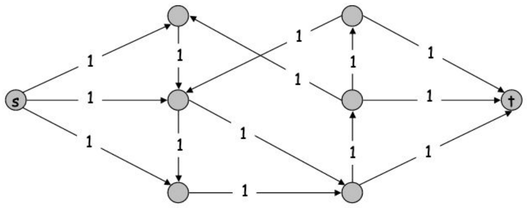


Theorem. Max number edge-disjoint s-t paths equals max flow value.

Pf. $\leq$

- Suppose there are k edge-disjoint paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if e participates in some path P_i ; else set $f(e) = 0$.
- Since paths are edge-disjoint, f is a flow of value k. ▪

Max flow formulation: assign unit capacity to every edge.



Theorem. Max number edge-disjoint s-t paths equals max flow value.

Pf. $\geq$

- Suppose max flow value is k.
- Integrality theorem $\Rightarrow$ there exists 0-1 flow f of value k.
- Consider edge (s, u) with f(s, u) = 1.
  - by conservation, there exists an edge (u, v) with f(u, v) = 1
  - continue until reach t, always choosing a new edge
- Produces k (not necessarily simple) edge-disjoint paths. ▪

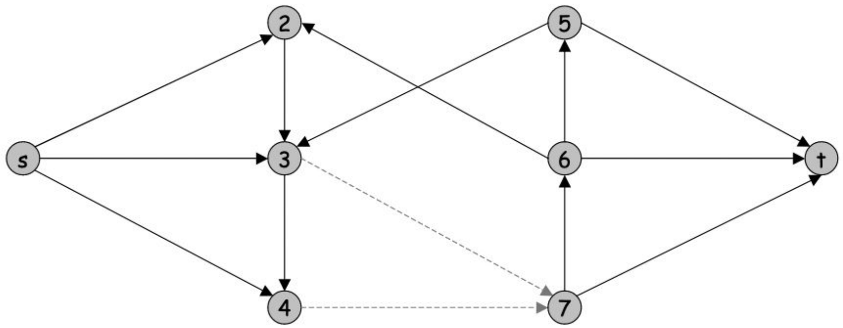
Network Connectivity

Network Connectivity

Network connectivity. Given a digraph $G = (V, E)$ and two nodes s and t, find min number of edges whose removal disconnects t from s.

Def. A set of edges $F \subseteq E$ disconnects t from s if all s-t paths uses at least on edge in F.

Menger's Theorem (1927). The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.



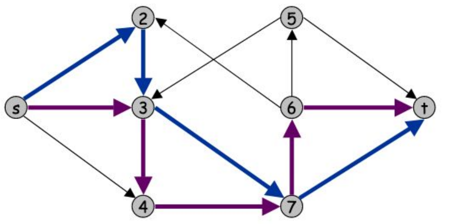
Network Connectivity

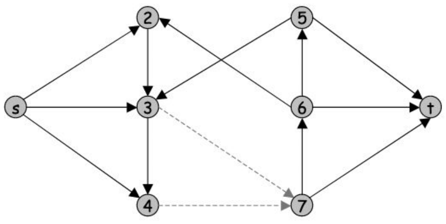
Edge Disjoint Paths and Network Connectivity

Menger's Theorem (1927). The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Pf. $\leq$

- Suppose the removal of $F \subseteq E$ disconnects t from s, and $|F| = k$.
- All s-t paths use at least one edge of F. Hence, the number of edge-disjoint paths is at most k. ▪



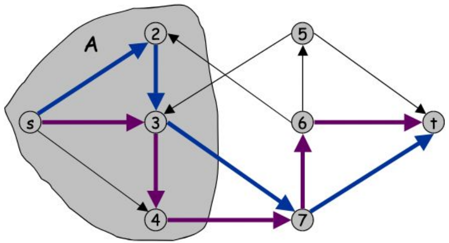


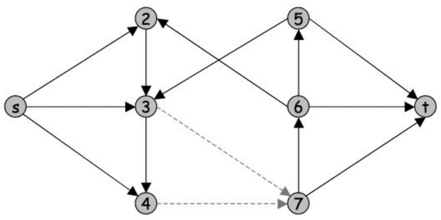
Disjoint Paths and Network Connectivity

Menger's Theorem (1927). The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Pf. $\geq$

- Suppose max number of edge-disjoint paths is k.
- Then max flow value is k.
- Max-flow min-cut $\Rightarrow$ cut (A, B) of capacity k.
- Let F be set of edges going from A to B.
- $|F| = k$ and disconnects t from s. ▪





NP-Completeness

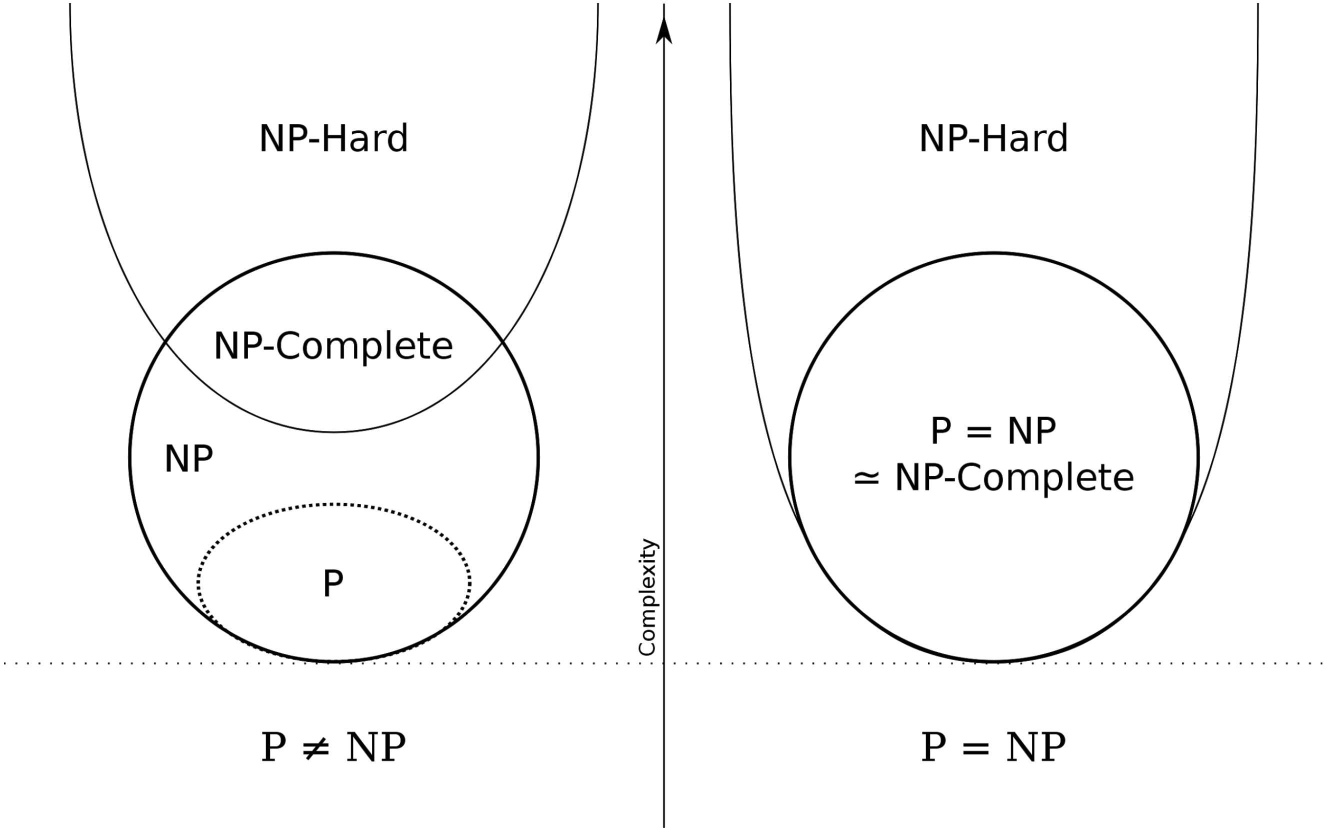
P is the class of decision problems which can be solved in polynomial time by a deterministic Turing machine.

NP is the class of decision problems which can be solved in polynomial time by a non-deterministic Turing machine. Equivalently, it is the class of problems which can be verified in polynomial time by a deterministic Turing machine.

NP-hard is the class of decision problems to which all problems in NP can be reduced to in polynomial time by a deterministic Turing machine.

NP-complete is the intersection of NP-hard and NP. Equivalently, NP-complete is the class of decision problems in NP to which all other problems in NP can be reduced to in polynomial time by a deterministic Turing machine.

NP-Completeness



Reduction

What is reduction

- Problem A is reducible to problem B if an algorithm for solving problem B efficiently (if it existed) could also be used as a subroutine to solve problem A efficiently.
- When this is true, solving A cannot be harder than solving B.

Example:

Polynomial Problems

Suppose: $Y \leq_P X$, and there is an polynomial time algorithm for X. Then, there is a polynomial time algorithm for Y.

NP completeness

If Y is NP-complete, and

1. X is in NP
2. $Y \leq X$
3. then X is NP-complete.

Practice

Vertex Cover - Given a graph G and a number k, does G contain a vertex cover of size at most k.

Set Cover - Given a set U of elements and a collection S1,...,Sm of subsets of U, is there a collection of at most k of these sets whose union equals U?

We want to show that Set Cover is NP-complete.

Practice

Vertex Cover - Given a graph G and a number k, does G contain a vertex cover of size at most k (find at most k vertices that are neighbors of all other vertices).

Set Cover - Given a set U of elements and a collection S1,...,Sm of subsets of U, is there a collection of at most k of these sets whose union equals U?

We want to show that Set Cover is NP-complete.

1. Choose an NP-complete problem, for example Vertex Cover.
2. Show that Set Cover $\in$ NP
3. Vertex Cover $\leq_P$ Set Cover, and therefore that Set Cover is NP-complete.

Practice

Thm. Vertex Cover $\leq_P$ Set Cover

Proof. Let $G = (V, E)$ and k be an instance of $\textsc{Vertex Cover}$. Create an instance of $\textsc{Set Cover}$:

- $U = E$
- Create a S_u for for each $u \in V$, where S_u contains the edges adjacent to u.

U can be covered by $\leq k$ sets iff G has a vertex cover of size $\leq k$.

Why? If k sets $S_{u_1}, \ldots, S_{u_k}$ cover U then every edge is adjacent to at least one of the vertices $u_1, \ldots, u_k$, yielding a vertex cover of size k.

If $u_1, \ldots, u_k$ is a vertex cover, then sets $S_{u_1}, \ldots, S_{u_k}$ cover U. $\square$

Not done yet

We still have to show that Set Cover is in **NP**!

The certificate is a list of k sets from the given collection. We can check in polytime whether they cover all of U. Since we have a certificate that can be checked in polynomial time, Set Cover is in **NP**.

Interview Questions!!!

Interview Questions!!!

Interview Questions!!!

terview Questions!!!

Trapping Rain

Trapping Rain Water or Water collected between towers : Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

```
Input - {1,5,2,3,1,7,2}
Output - 9

          |
          |
  | - - - |
  | - - - |
  | - | - |
  | | | - | |
| | | | | | |
1 5 2 3 1 7 2
Maximum water in above land can be 9 se - lines between buildings
```

Trapping Rain Solution

Algorithm:

1: With given towerHeight array, create 2 arrays, maxSeenRight and maxSeenLeft.

maxLeft[i] – max height on left side of Tower[i].

maxRight[i] – max height on right side of Tower[i].

2: Calculate for each tower:

rainwater[i] = rainwater[i] + Max(Min(maxLeft[i], maxRight[i]) – towerHeight[i], 0);