Week 9 Day 1 Lecture Notes

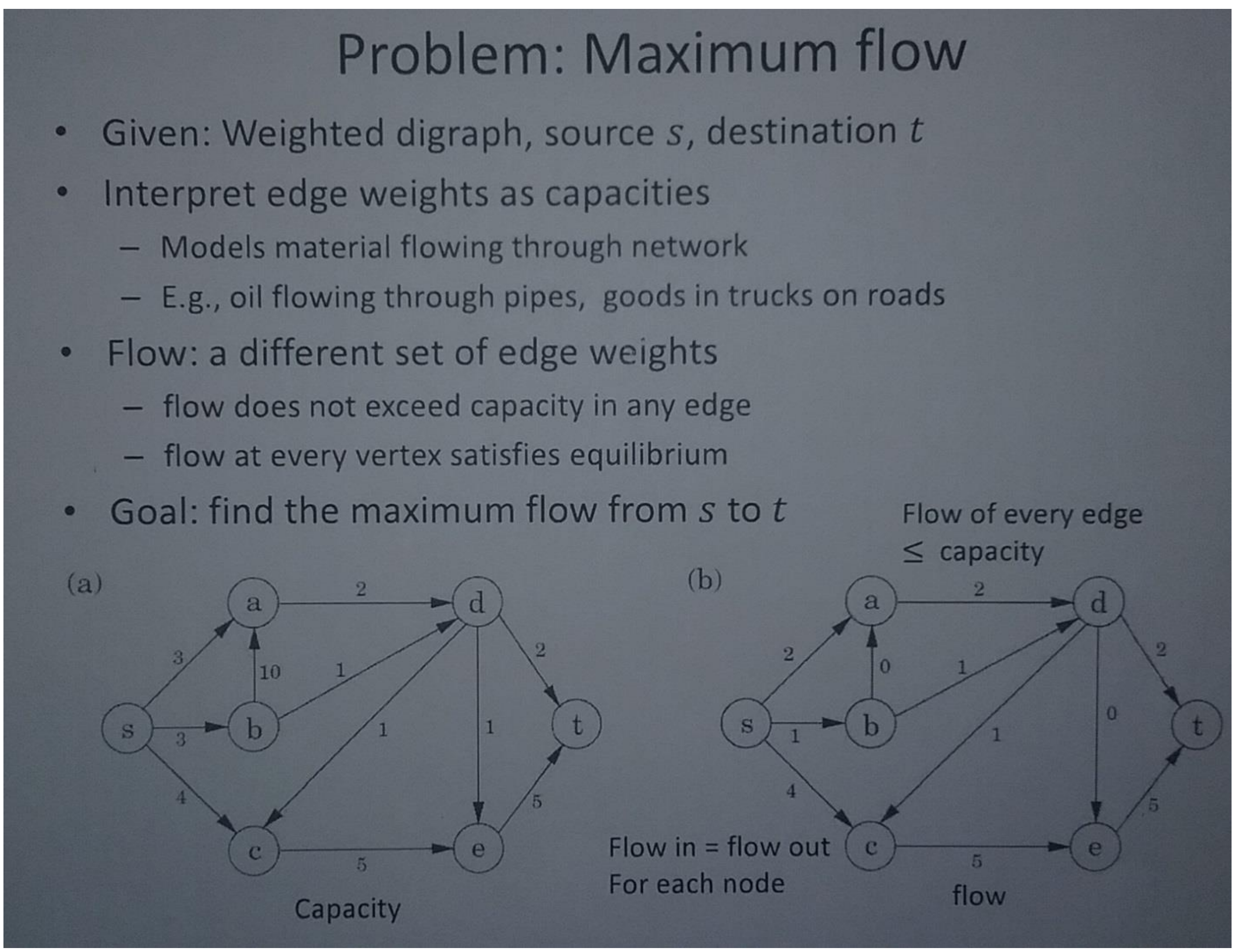
Prep:

- Review Linear Programming and reduction
- Quiz 4 grades are posted.
- Quiz 5 on Thursday.

Solving Linear Programs:

- Geometric way to ‘eyeball’ where the solution will be.
- Manipulate problems into linear programming forms.
- We rely on very sophisticated packages to solve large scale Linear Programs.
- We can view these as Black Boxes, and can run in polynomial time.

Maximum Flow Problem:



- Flow cannot exceed capacity.
- Linear program variables.
- For each edge there is a continuous variable we have to decide.

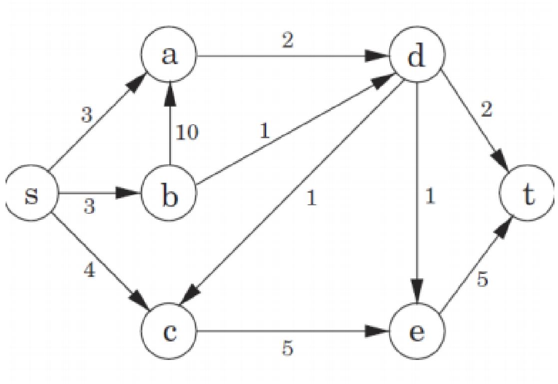
LP Formulation of the problem:

LP formulation of the problem

One variable per edge to model the flow

One inequality constraint per edge

One equality constraint per node (except for source and sink)



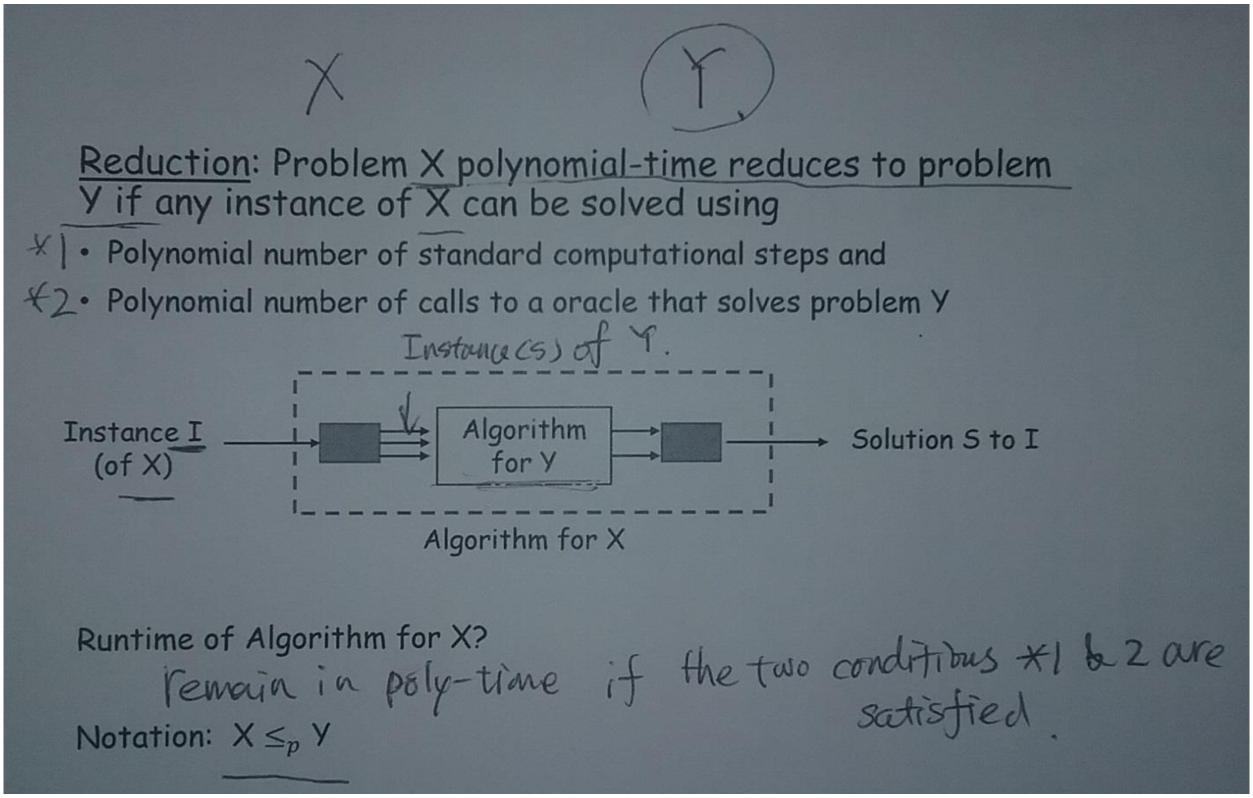
- How to calculate the total flow? Maximize the sum of the flow that is coming out of S or going into T.
- Objective: Maximize $X_{sa}+X_{sb}+X_{sc}$
- Objective, constraints, and variables. (all linear)

Moving to the topic of Reduction:

Intractability Reduction:

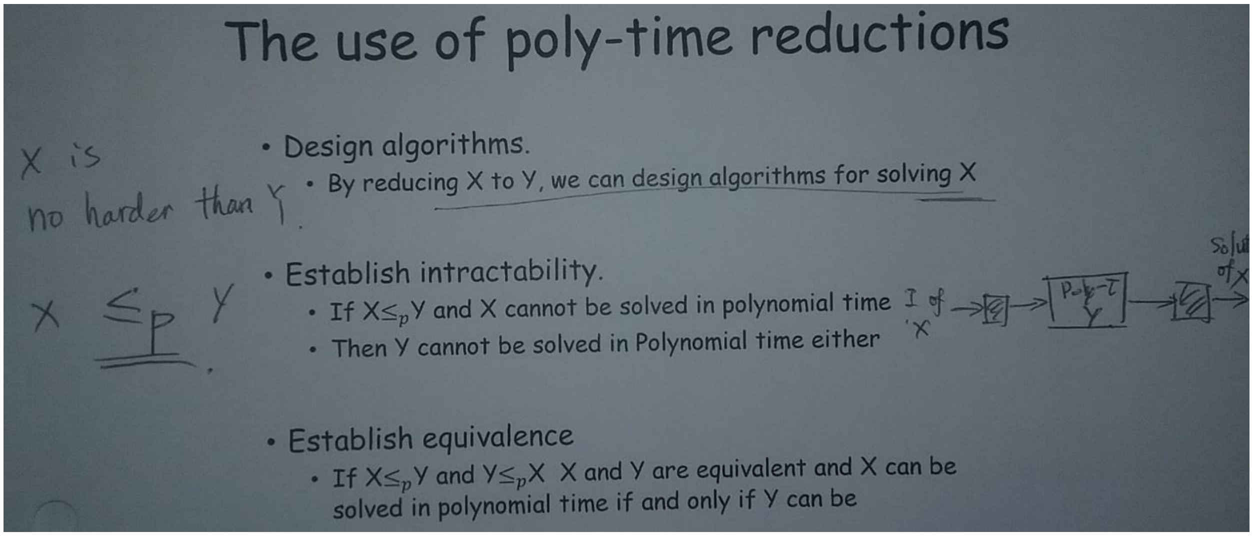
- What can we solve?
- Polynomial runtime.
- What else can we solve in poly runtime? Any problem we can reduce to problem Y we can also solve in Polynomial time.

Reduction: Problem X can be reduced to problem y IF any instance of x can be solved using:



- Polynomial number of standard computational steps
- And Polynomial number of calls to an oracle that solves problem y.
- Runtime notation for X? X<= Y
- Abstract example. How do we turn an instance of X into an Instance of Y? (pre-processing time we add to the Y solver).
- **Runtime of X will remain in Polynomial time** if both conditions are satisfied:
  1. • Polynomial number of standard computational steps and
  2. • Polynomial number of calls to a oracle that solves problem Y

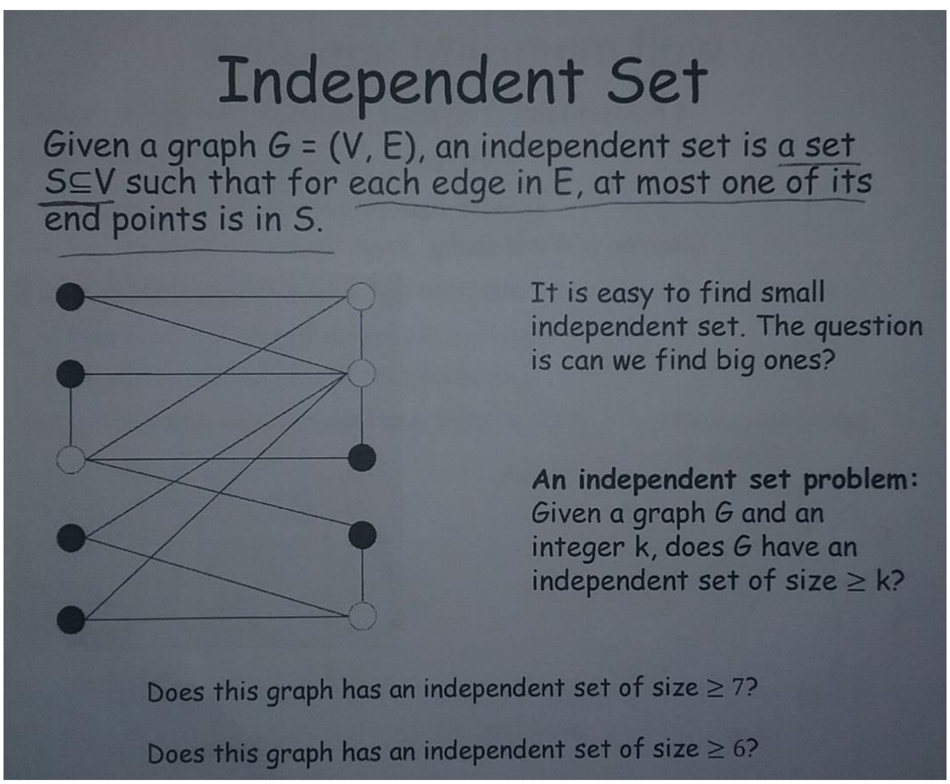
Basis for Reduction:



- We have a **Problem Y that can be solved in Polynomial time**.
- We have a Problem X, that can be **reduced (changed) so that we can use the solver for Problem Y**. than the runtime will remain polynomial if:

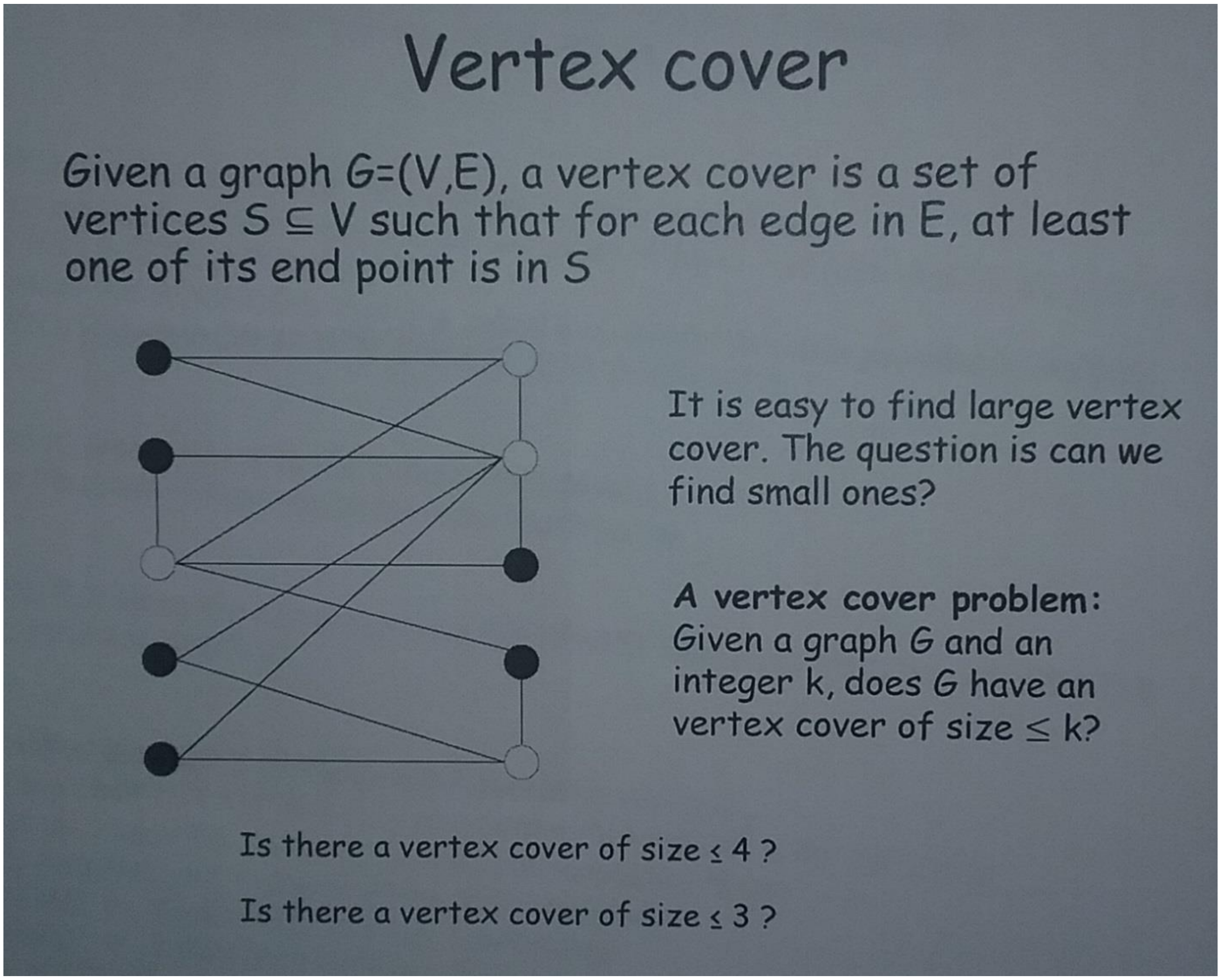
1. • Polynomial number of standard computational steps and
2. • Polynomial number of calls to a oracle that solves problem Y

Independent Set:



- Graph as a maximum independent set of size 6. (Black highlighted nodes as the independent set)
- Each node cannot touch (by edge) another node in the independent set. This is why it's called **an independent set.**
- There can be multiple Independent sets with the largest size.

Vertex Cover:



- A vertex cover is such that every node has at least one node in the set.
- For this example all of the White Nodes are the smallest Vertex Cover.
- It Happens that the nodes that are not used in the **Independent Set** Are used for the smallest Vertex Cover.

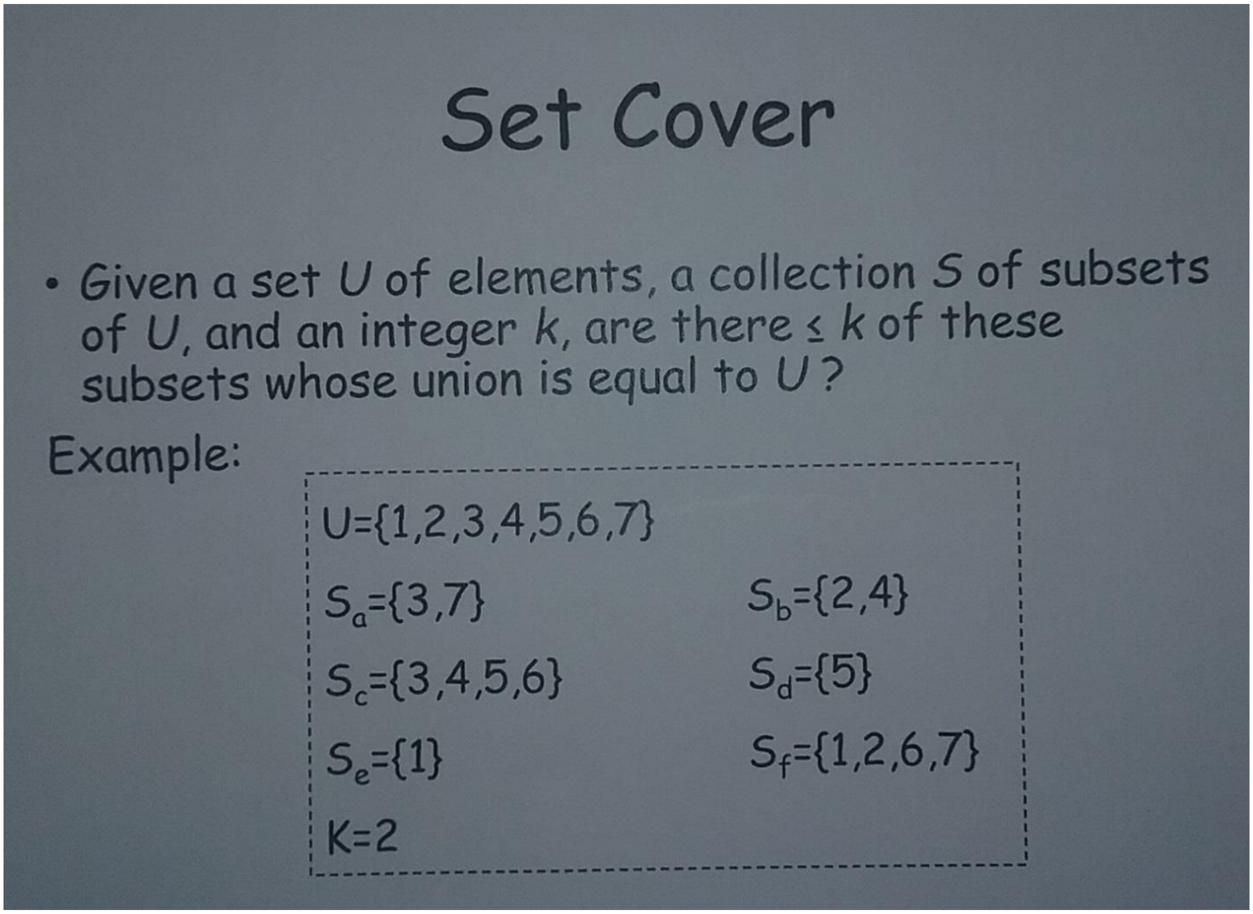
Independent Set and Vertex Cover can be reduced to one another:

- Because the nodes not used in the independent set are nodes for a vertex cover.
- Max Independent Set implies Smallest Vertex Cover.

Reducing Independent Set to Vertex Cover: (picture*)

- **Summary: Solving the Vertex Covert of Independent Set will give a solver to the other.**

Set Cover:



- **Given a Degree with Requirements U = {1,2…7}**
- How few courses can we take to fulfill all the requirements for the Degree set U?

Next time:

- **Read DPV 8.1-8.3**
- Work on Implementation 3. (Due 3/17)
- Quiz 5 on Thursday (3/9)
- **Recitation Wednesday usual time.**
- Continue reviewing for Final

End of Week 9 Day 1 Notes

~Information composed by Notetaker Scott Russell for CS 325 **DAS** students