

Week 9 Day 1 Lecture Notes

Prep:

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

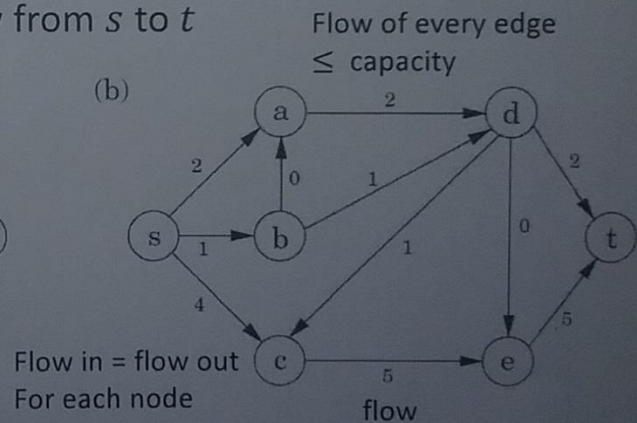
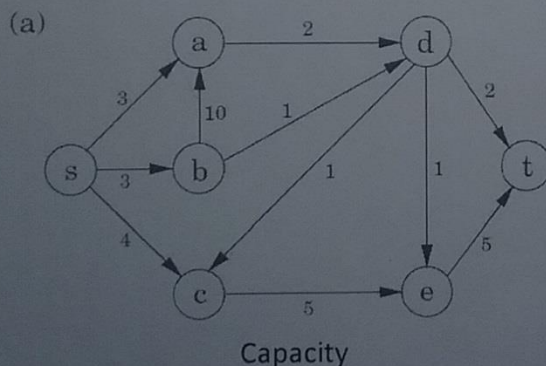
Solving Linear Programs:

- 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:

Problem: Maximum flow

- Given: Weighted digraph, source s , destination t
- Interpret edge weights as capacities
 - Models material flowing through network
 - E.g., oil flowing through pipes, goods in trucks on roads
- Flow: a different set of edge weights
 - flow does not exceed capacity in any edge
 - flow at every vertex satisfies equilibrium
- Goal: find the maximum flow from s to t



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

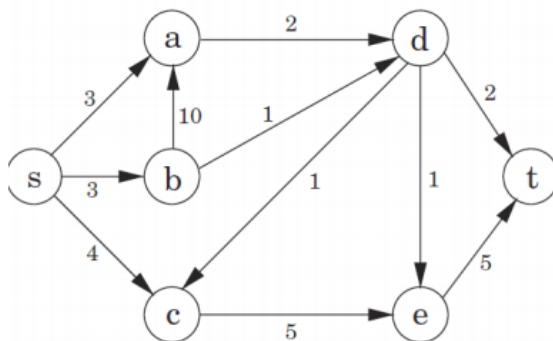
LP Formulation of the problem:

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)



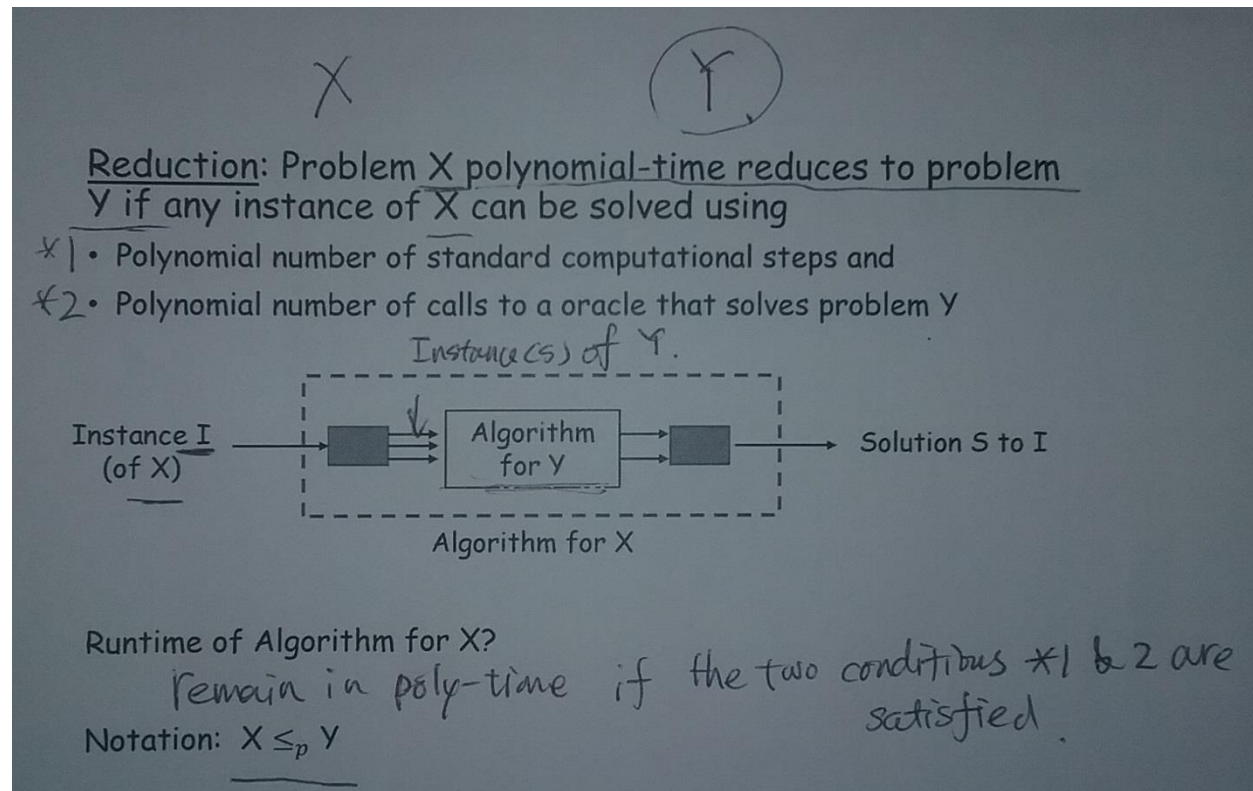
- How to calculate the total flow? Maximize the sum of the flow that is coming out of S or going into T.
- Objective: Maximize $X_{sa} + X_{sb} + X_{sc}$
- Objective, constraints, and variables. (all linear)

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? $X \leq Y$
- 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:

The use of poly-time reductions

X is no harder than Y .

$X \leq_P Y$

- Design algorithms.
 - By reducing X to Y , we can design algorithms for solving X
- Establish intractability.
 - If $X \leq_P Y$ and X cannot be solved in polynomial time
 - Then Y cannot be solved in Polynomial time either
- Establish equivalence
 - If $X \leq_P Y$ and $Y \leq_P X$ X and Y are equivalent and X can be solved in polynomial time if and only if Y can be

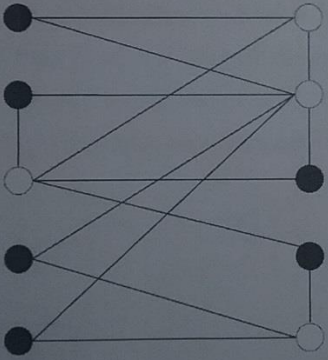
Handwritten diagram: A box labeled 'X' is connected by an arrow to a box labeled 'Y' with 'P' and 'T' inside. An arrow from 'Y' points to a box labeled 'Solution of X'.

- 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**. then the runtime will remain polynomial if:
 1. • Polynomial number of standard computational steps and
 2. • Polynomial number of calls to an oracle that solves problem Y

Independent Set:

Independent Set

Given a graph $G = (V, E)$, an independent set is a set $S \subseteq V$ such that for each edge in E , at most one of its end points is in S .



It is easy to find small independent set. The question is can we find big ones?

An independent set problem:
Given a graph G and an integer k , does G have an independent set of size $\geq k$?

Does this graph has an independent set of size ≥ 7 ?

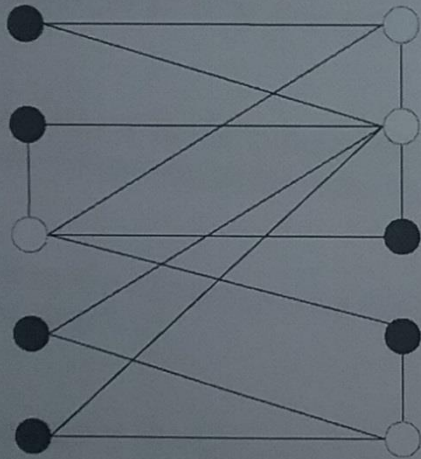
Does this graph has an independent set of size ≥ 6 ?

- 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:

Vertex cover

Given a graph $G=(V,E)$, a vertex cover is a set of vertices $S \subseteq V$ such that for each edge in E , at least one of its end point is in S



It is easy to find large vertex cover. The question is can we find small ones?

A vertex cover problem:
Given a graph G and an integer k , does G have a vertex cover of size $\leq k$?

Is there a vertex cover of size ≤ 4 ?

Is there a vertex cover of size ≤ 3 ?

- 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 Cover of Independent Set will give a solver to the other.**

Set Cover:

Set Cover

- Given a set U of elements, a collection S of subsets of U , and an integer k , are there $\leq k$ of these subsets whose union is equal to U ?

Example:

$$U = \{1, 2, 3, 4, 5, 6, 7\}$$

$$S_a = \{3, 7\}$$

$$S_b = \{2, 4\}$$

$$S_c = \{3, 4, 5, 6\}$$

$$S_d = \{5\}$$

$$S_e = \{1\}$$

$$S_f = \{1, 2, 6, 7\}$$

$$K = 2$$

- **Given a Degree with Requirements $U = \{1, 2, \dots, 7\}$**
- 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