COMP 360: Algorithm Design

Julian Lore

Last updated: January 8, 2018

Notes from Hatami Hamed's Winter 2018 lectures.

Contents

1 01/08/18

Course webpage. Look at it for more details on the grading scheme, assignments and more.

We are assumed to have some background in the course, so today Hatami will be looking over what we should know for this course.

1.1 Background Knowledge

- Tree
- Graph, $G = (V, E)$ (all questions in assignments and exams will be written formally, so you should know what the letters mean)
- DFS, BFS

- Basic algorithm techniques: Greedy algorithms, dynamic programming, divide and conquer, recursion
- Running time analysis (Big-O notation)
- It's important that you should be able to read math, like precise and formal notation.

1.2 Sample Problems

You should be able to read and understand these problems. The problems are available here on the course webpage.

Example 1 S is a set of positive integers.

$$A = \sum_{x \in S} x^2$$
$$B = \sum_{\substack{x \in S, \\ x^2 \in S}} x$$

Let $S = \{1, 2, 3, 4, 5\}$. What are A and B?

$$A = 1^2 + 2^2 + 4^2 + 5^2 = 1 + 4 + 9 + 16 + 25 = 46$$
$$B = 1 + 2 = 3$$

For B, the number must be in S and its square must also be in S.

Example 2 M is an $n \times n$ matrix. M_{ij} denotes ij-entry of M. The total sum of the entries of M is 100.

$$\sum_{i=1}^{n} \sum_{j \in \{1,\dots,n\} \setminus \{i\}} \sum_{r=1}^{n} M_{ir} = ?$$
$$= \sum_{i=1}^{n} \sum_{r=1}^{n} (n-1) M_{ir} = (n-1)100$$

Since we are summing the inner entry $n-1$ times (the second summation).
Binary expansion/representation.

Example 3 How many digits are in the binary expansion of n?

$$\text{Ex.}n = 5 \implies n = \underbrace{101}_{\text{binary}}$$

$$\lceil \log_2 n \rceil \text{ is the answer.}$$

Example 4

$$\sum_{n=0}^{k} 2^n =? = 2^{k+1} - 1$$

In binary, this is $\underbrace{1111\ldots1}_{\text{binary}}$. Note that this is a geometry sum and that you should be able to calculate these.

Example 5 $S = (a_1, a_2, \ldots, a_n)$ a sequence of integers. E is the set of even numbers in $\{1, \ldots, n\}$.

$A = \sum_{i \in E} a_i$

Example:

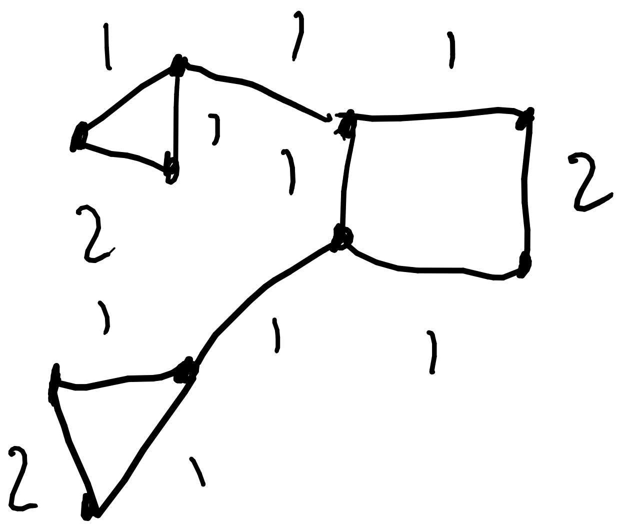
$S = \{1, \underline{3}, 2, \underline{5}, 4\}$
$A =? = \sum_{i \in \{2,4\} a_i} = a_2 + a_4 = 3 + 5 = 8$

Example 6 $G = (V, E)$ an undirected graph. Suppose to every edge uv a number C_{uv} is assigned. What does the following statement mean?

$$\exists c \forall u \in v \sum_{uv \in E} c_{uv} = c$$

There exists some number c, such that for every vertex we chose, the sum of all edges containing this vertex is the same for all vertices.

Example



In this case, $c = 3$.

Example 7 $G = (V, E)$ undirected graph degree of every vertex is 10. Suppose to every vertex $v \in V$ a positive integer a_v is assigned.
If $\sum_{v \in V} a_v = 5$ then what is $\sum_{u \in v} \sum_{\substack{w \in V: \\ uw \in E}} a_w = ? = \sum_{w \in V} 10 a_w = 10 \times 5 = 50$. Each a_w appears in the sum 10 times since the degree of each vertex is 10.

1.3 Topics Covered

The following are the topics we will be covering in this course:

- Network flows (More of like a practice topic for what we'll be seeing in the course, will use the algorithm to solve this problem for seemingly unrelated problems. We'll be doing this a lot in the course, called reduction, where we reduce solving one problem to another problem.)
- Linear Programming (Bunch of constraints and want to optimize a linear function). This will be one of the most important concepts we learn in this course.
- Midterm
- Linear Programming again

- NP-Completeness (no good algorithms for problems that seem very basic, useful skill to have even if you aren't a theoretician)
- Approximation algorithms (settling for the next best thing for NP-Complete problems, might be able to find an algorithm that approximates things, not exactly optimal, but some sort of factor of how good the approximation is; lots of research happening in this area, better and better approximations). Will use a lot of linear programming here.
- Randomized algorithms (randomness can actually help us; probability theory/knowledge of random variables may help a little bit here, but this is the last stretch of the course and not very essential)

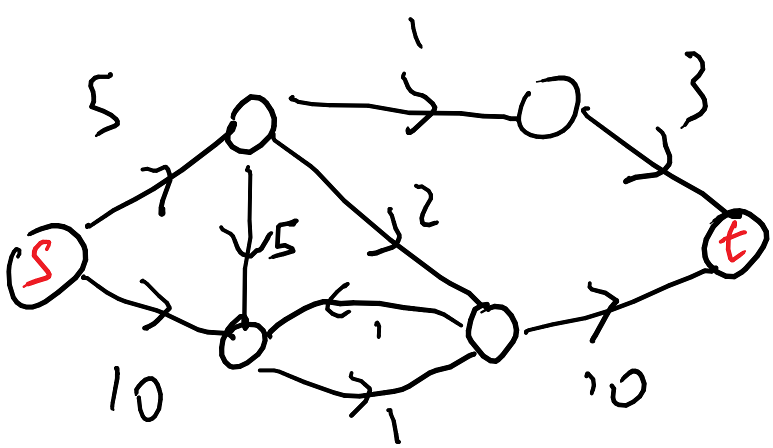
1.4 Network Flows

Max Flow Problem

Very important, used in things like game theory. Def: A flow is a directed graph $G = (V, E)$ such that:

1. Every edge e has a capacity $c_e \geq 0$.
2. There is a source $s \in V$.
3. There is a sink $t \in V$ such that $t \neq s$.

Example



Remark : For the sake of convenience we make the following assumptions.

1. No edge enters the source.

2. No edge leaves the sink.

3. All capacities are integers.

4. There is at least one edge incident to every vertex.

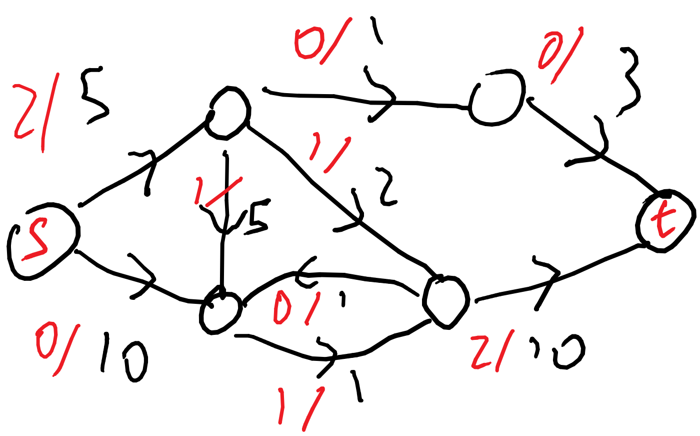
Def: [flow] A flow is a function $f : E \to \mathbb{R}^+$ such that: (Note that :$\mathbb{R}^+ = \{X \in \mathbb{R} | x \geq 0\}$)

(i) [capacity] $\forall e \in E, 0 \leq f(e) \leq c_e$ (flow cannot be negative nor can it exceed capacity)

(ii) [conservation] For every node u other than source and sink the amount of flow that goes into $u =$ the amount of flow that leaves u. Formally:

$$\forall u \in V \setminus \{s,t\} \underbrace{\sum_{vu \in E} f(uv)}_{f^{\text{in}}(u)} = \underbrace{\sum_{uw \in E} f(uw)}_{f^{\text{out}}(u)}$$

Example



Def: $Val(f) = \sum_{su \in E} f(su) = f^{\text{out}} s$

Max Flow Problem: Given a flow network find a flow with largest possible value.