Graph problems with linear programming

Ben Rosenberg

December 19, 2025

Overview

- Graphs and math notation review
- Maximum flow problem
- Shortest path problem
- Integrality

Math notation review

Sets:

- $a \in A$: the value a is in the set A
- $A = \{1, 2, 3\}$: the values 1, 2, and 3 comprise the set A
- $A \subseteq B$: the set A is a subset of (and possibly equal to) the set B
- $A \subset B$: the set A is a subset of (and not equal to) the set B
- $|A|$: the number of elements in the set A
- $A \backslash B$: the set of elements that are in A but not in B (set difference)

Useful sets to know symbols for:

- Natural numbers: $\mathbb{N} = \{0, 1, 2, 3, \dots\}$
  - Sources may differ on whether 0 is included in $\mathbb{N}$, in our class we will say it is included
- Integers: $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$
- Real numbers: $\mathbb{R}$, includes e.g. π, e, $1/2$, -1.234353, 0, etc.

Math notation review (cont.)

Tuples and Cartesian product:

- $T = (t_1, t_2, t_3)$: the values t_1, t_2, t_3 comprise a tuple T (in that order)
- $X \times Y$: if $|X| = n$ and $|Y| = m$, then $X \times Y$ is the set of tuples $(x_1, y_1), (x_1, y_2), \ldots, (x_1, y_m), \ldots, (x_n, y_1), \ldots, (x_n, y_m)$, with $|X \times Y| = nm$

Functions:

- $f : A \to B$: the function f has a domain (set of possible input elements) of A, and a range (set of possible output elements) of B
- $f(1) = 2$: the result of applying f to 1 is 2

Math notation review (cont.)

Quantifiers:

- $\forall x \in X : (p(x))$: for all values x in the set X, the proposition $p(x)$ holds
- $\exists x \in X : (p(x))$: there exists some value x in the set X for which the proposition $p(x)$ holds

Sums and products (examples):

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$: the sum of all numbers from 1 to n is equal to $\frac{n(n+1)}{2}$
- $\prod_{i=1}^{n} i = n!$: the product of all numbers from 1 to n is equal to $n!$

Combining notations:

- $\sum i \quad \forall i \in \{1, 2, \ldots, 10\} = 1 + 2 + \cdots + 10 = 55$
- $\sum x \quad \forall x : x \in \mathbb{N}, x \leq 10, x \equiv 0 \bmod 2 = 30$

Graphs

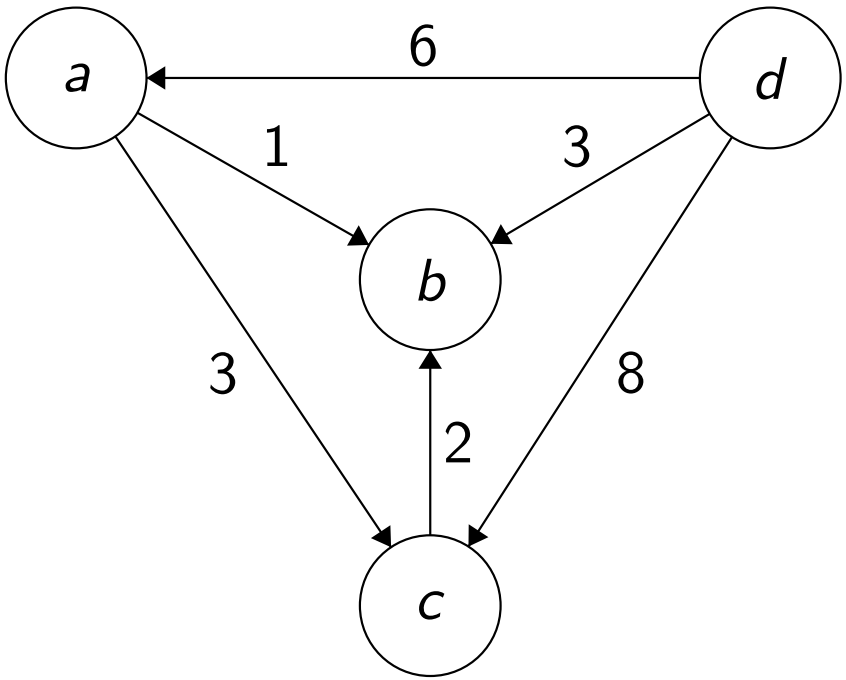
A graph is a 2-tuple $G = (V, E)$ where V is the set of vertices and $E \subseteq V \times V$ is a set of edges.

We will be considering directed graphs in these examples, so each edge $e \in E$ from node v_i to node v_j is denoted as a 2-tuple (v_i, v_j) (rather than a set $\{v_i, v_j\}$).

We can extend the graph to have additional properties, e.g. capacities or distances on edges. We will typically write a property of an edge as a function $C : E \to \mathbb{N}$ (where C can either mean "capacity" or "cost", depending on the application).

We restrict the cost/capacity mapping to take on nonnegative integer values here. We will see later why this is important.

Graph example



This graph has the following vertices, edges, and costs:

- $V = \{a, b, c, d\}$
- $E = \{(a, b), (a, c), (c, b), (d, a), (d, b), (d, c)\}$
- $C = \{((a, b), 1), ((a, c), 3), ((c, b), 2), ((d, a), 6), ((d, b), 3), ((d, c), 8)\}$

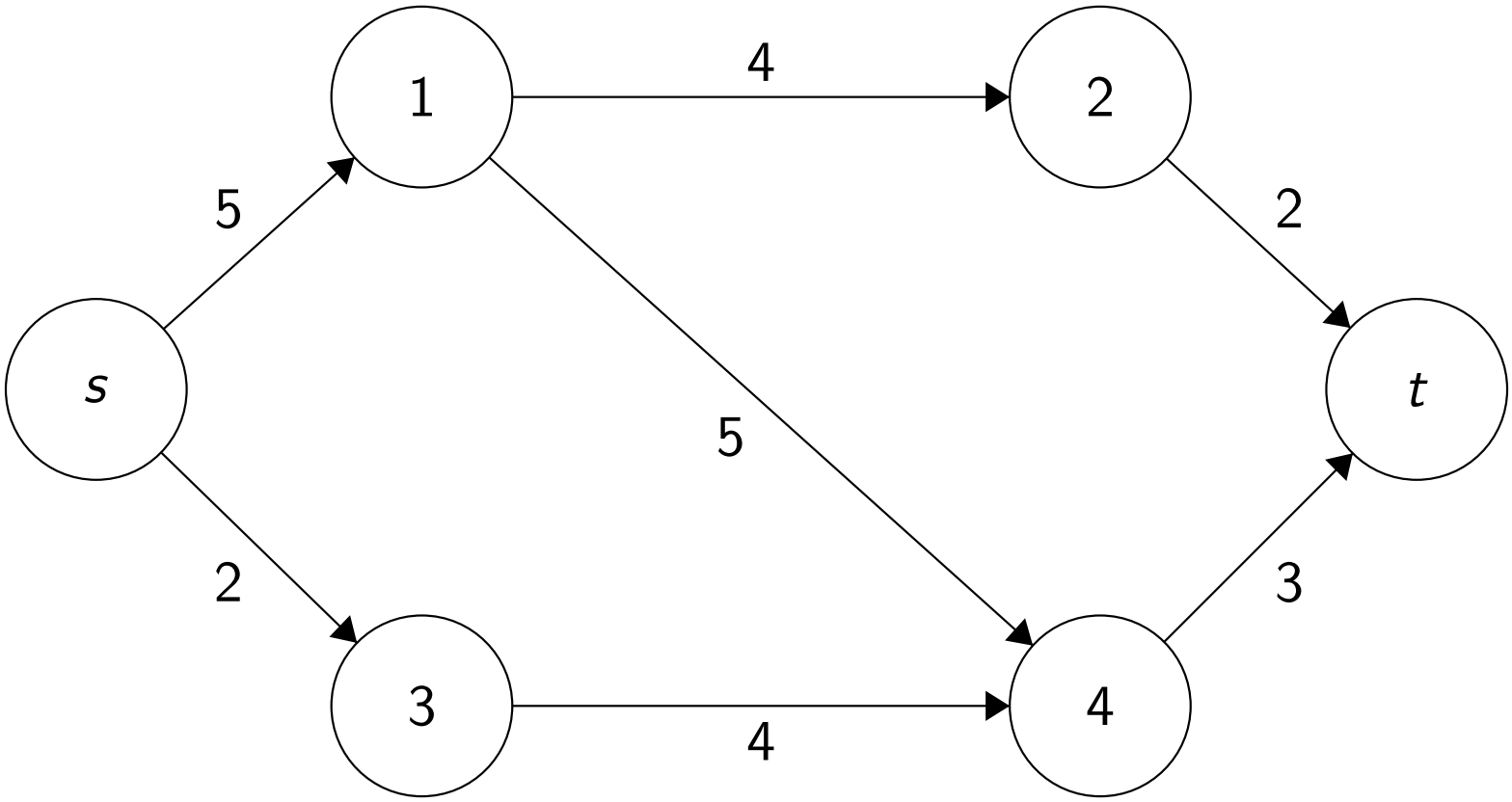
Max flow

The max flow problem considers a directed graph G with two specific vertices $s, t \in V$, so that s has no parents and t has no children.

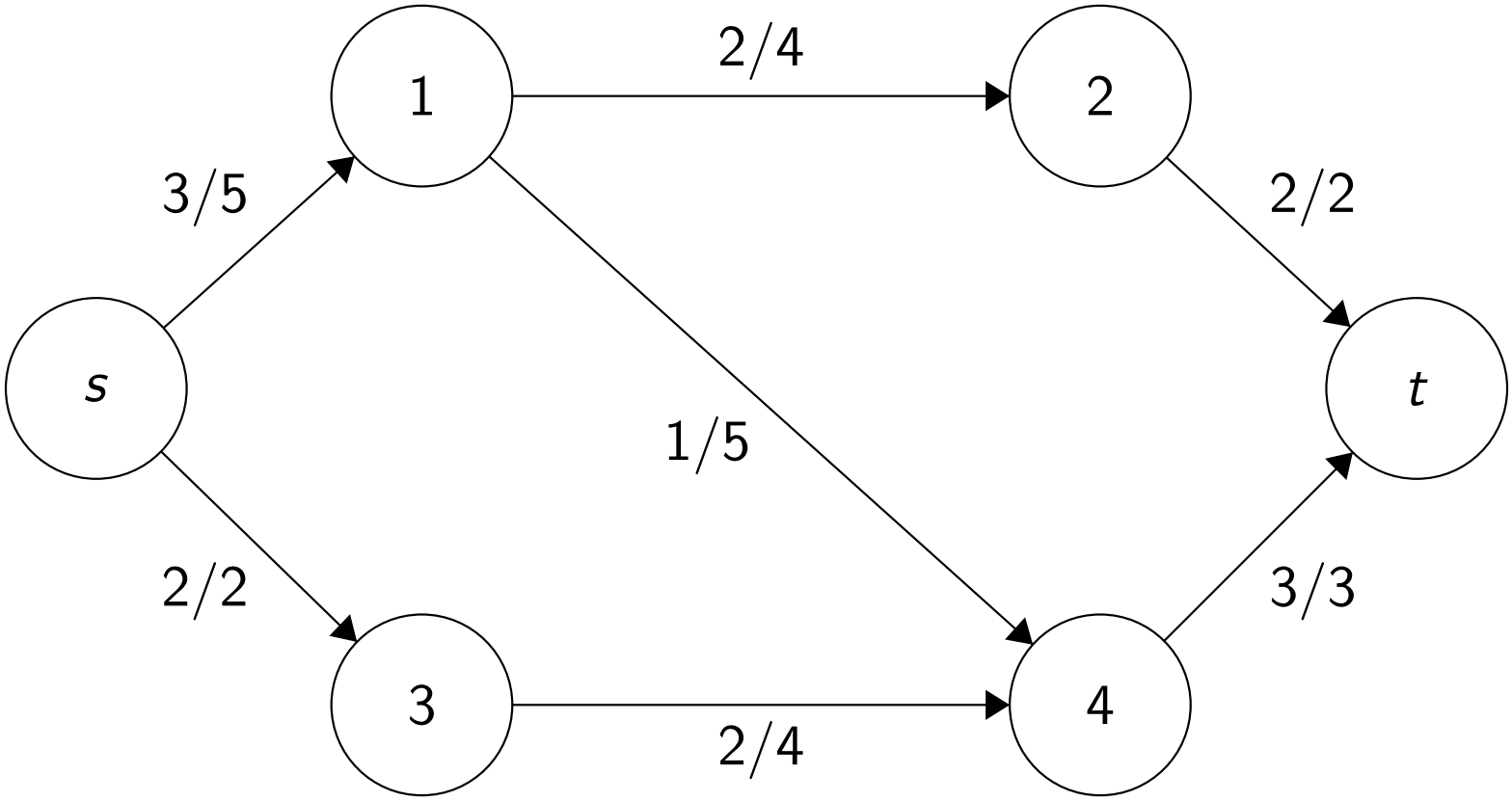
Additionally, G has a capacity on each of its edges, which we can denote by C as described earlier. The capacity of an edge is the maximum amount of "flow" that can be sent through that edge.

The objective is to maximize the flow that can be sent from s to t.

Max flow example



Max flow example (one possible solution)



Formalizing max flow

How can we formalize "sending flow" from s to t?

Think about:

- Decision variables
- Constraints
- Objective function

Formalizing max flow (cont.)

- Decision variables:
  - The amount of flow being sent on each edge
- Constraints:
  - Capacities of each edge must be obeyed
  - For vertices besides s and t, the amount of flow into a node must equal the amount of flow out of a node
- Objective function:
  - Maximize the amount of flow from s to t (can choose to maximize either the amount of flow either being sent from s or the amount of flow being sent into t)

Formalizing max flow: Decision variables

We need a decision variable for the amount of flow on each edge.

This means that we need a decision variable for each edge:

$$x_{i,j} : (i,j) \in E$$

We can't send negative flow:

$$x_{i,j} \geq 0 \qquad \forall (i,j) \in E$$

Formalizing max flow: Constraints

The capacities of each edge must be obeyed:

$$x_{i,j} \leq C((i,j)) \qquad \forall i,j : (i,j) \in E$$

For vertices besides s and t, flow in must equal flow out:

$$\sum_{i \in V} x_{i,k} = \sum_{j \in V} x_{k,j} \qquad \forall k \in V \backslash \{s,t\}, \forall i,j : (i,k) \in E, (k,j) \in E$$

Formalizing max flow: Objective function

Maximize the amount of flow from s to t (WLOG, choose to maximize the amount of flow leaving s):

$$\max \quad \sum_{j \in V} x_{s,j} \qquad \forall s, j : (s, j) \in E$$

Formalizing max flow: Final LP

Everything together:

$$
\begin{aligned}
\max \quad & \sum_{j \in V} x_{s,j} && \forall s, j : (s,j) \in E \\
s.t. \quad & x_{i,j} \leq C((i,j)) && \forall i, j : (i,j) \in E \\
& \sum_{i \in V} x_{i,k} = \sum_{j \in V} x_{k,j} && \forall k \in V \backslash \{s,t\}, \forall i, j : (i,k) \in E, (k,j) \in E \\
& x_{i,j} \geq 0 && \forall (i,j) \in E
\end{aligned}
$$

We can then (as we will see later) write a program that solves the above LP to determine the optimal values of $x_{i,j}$.

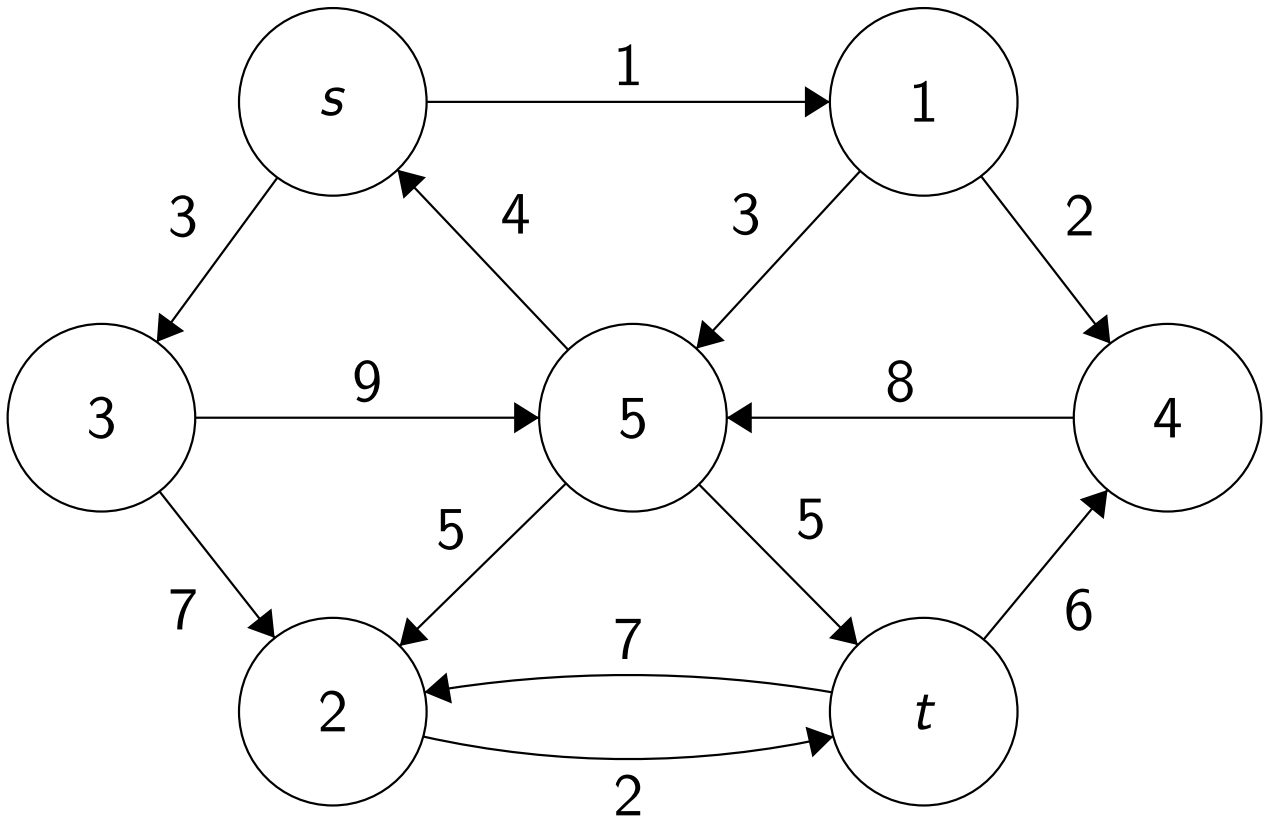
Shortest path

The shortest path problem considers a directed graph G with two specific vertices $s, t \in V$. (Unlike max flow, s and t may have parents and/or children.)

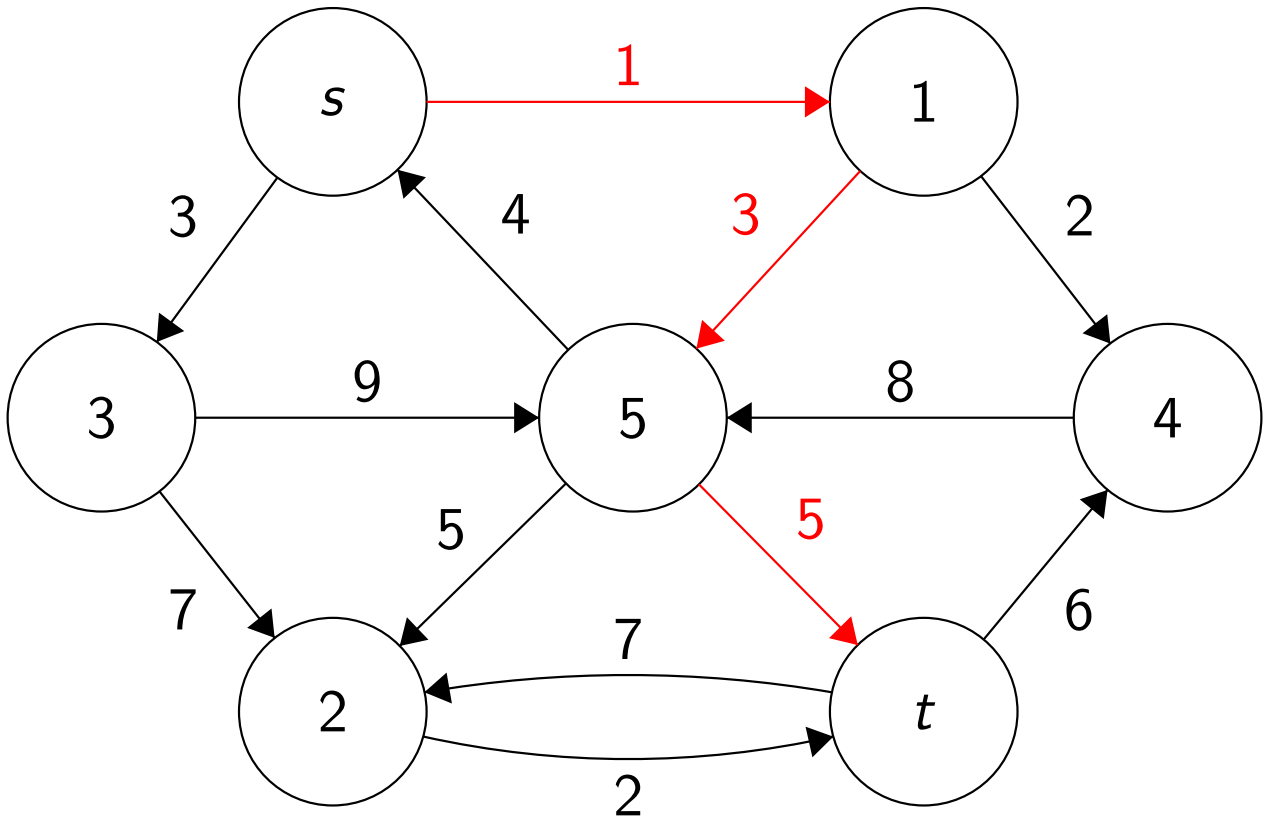
Additionally, G has a cost on each of its edges, which we denote by C. The cost of an edge is the distance from the start vertex to the end vertex of that edge.

The objective is to determine the shortest path that can be sent from s to t.

Shortest path example



Shortest path example (solution)



Shortest path LP intuition

Intuition: We can build off of our max flow LP.

- Send one unit of flow through the graph
- Have a decision variable for each edge indicating whether flow is being sent through it (1) or not (0)
- Flow conservation: s will send one unit of flow and t will receive one unit of flow, all other vertices will net to 0
- Since each edge's decision variable will indicate whether it is used or not, we can directly minimize the product of that decision variable with the edge's cost to minimize the length of the path (and therefore get the shortest path as a result)

Shortest path: Decision variables

Similarly to the max flow LP, we have a decision variable for each edge:

$$x_{i,j} \qquad \forall i, j : (i, j) \in E$$

These will need to be either 0 or 1, depending on whether the edge is used in the path:

$$0 \leq x_{i,j} \leq 1 \qquad \forall i, j : (i, j) \in E$$

We can't specify that a variable must be an integer, so in theory $x_{i,j}$ could take on a value of (e.g.) 0.5, but as we'll see later that won't be an issue for this problem.

Shortest path: Constraints

Flow conservation: s will send one unit of flow and t will receive one unit of flow, all other vertices will net to 0:

$$\sum_{j \in V} x_{s,j} - \sum_{i \in V} x_{i,s} = 1$$

$$\sum_{j \in V} x_{k,j} - \sum_{i \in V} x_{i,k} = 0 \quad \forall k \notin \{s, t\}$$

$$\sum_{j \in V} x_{t,j} - \sum_{i \in V} x_{i,t} = -1$$

Shortest path: Objective function

Minimize the total cost of the chosen edges:

$$\min \quad \sum_{(i,j)\in E} x_{i,j} \cdot C((i,j))$$

Shortest path: Final LP

Everything together:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} x_{i,j} \cdot C((i,j)) \\
s.t. \quad & \sum_{j\in V} x_{s,j} - \sum_{i\in V} x_{i,s} = 1 \\
& \sum_{j\in V} x_{k,j} - \sum_{i\in V} x_{i,k} = 0 \quad \forall k \notin \{s,t\} \\
& \sum_{j\in V} x_{t,j} - \sum_{i\in V} x_{i,t} = -1 \\
& 0 \leq x_{i,j} \leq 1 \qquad \forall i,j : (i,j) \in E
\end{aligned}
$$

Shortest path: Integrality

One thing we didn't consider:

- We have no constraints on $x_{i,j}$, except that it's between 0 and 1
- Therefore, we could get some solution where $0 < x_{i,j} < 1$, e.g. $x_{i,j} = 0.5$ (which would not make sense)

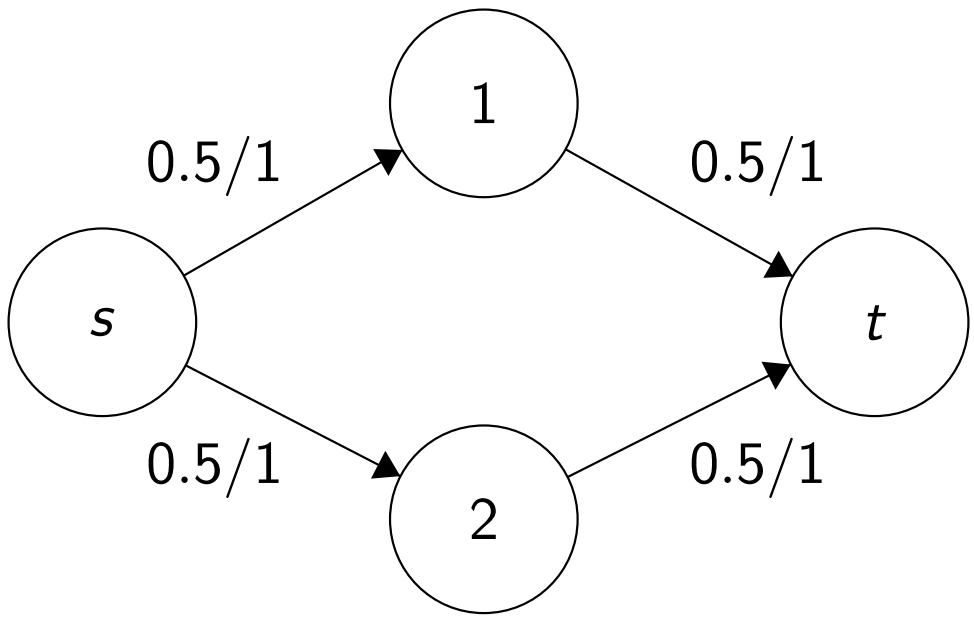
Thankfully, we can show that the optimal solution will never have this result.

The proof (sketch) we will do holds for the max flow problem as well - all flow values will take on integer values (provided the capacities are integer).

Shortest path: Integrality (cont.)

Remember we previously restricted the capacities/costs to be in $\mathbb{N}$. Here (in the shortest path problem), the capacities are all 1, but again this holds for the max flow problem too.

Suppose that we have the following type of shortest path (or max flow) solution, which is undesirable because of the non-integral values for $x_{i,j}$:



Shortest path: Integrality (cont.)

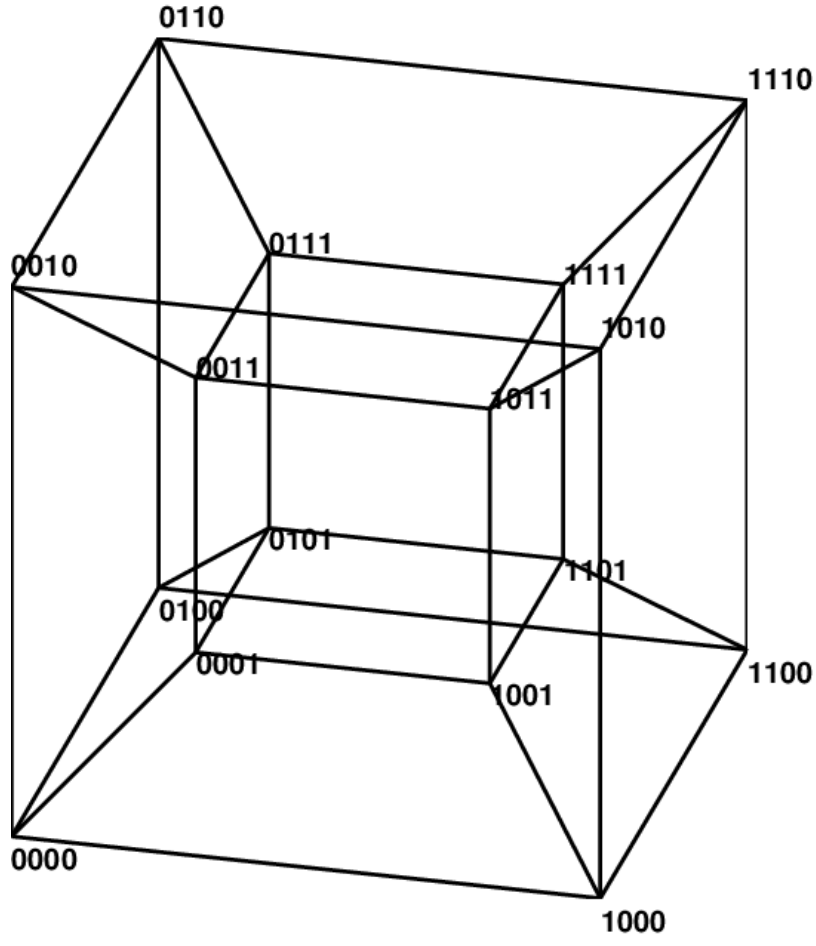
We can think about the extreme points in the LP feasible region. Technically there are 4 variables here $(x_{s,1}, x_{s,2}, x_{1,t}, x_{2,t})$, so this is in 4 dimensions, but we can consider the corner points without thinking too hard.

There are 4 variables here, and each can take on values between 0 and 1. Thus, this will be a hypercube with dimension 4. The extreme points will contain all the permutations of the decision variables being 0 or 1:

- $(x_{s,1}, x_{s,2}, x_{1,t}, x_{2,t}) = (0, 0, 0, 0)$
- $(x_{s,1}, x_{s,2}, x_{1,t}, x_{2,t}) = (0, 0, 0, 1)$
- $(x_{s,1}, x_{s,2}, x_{1,t}, x_{2,t}) = (0, 0, 1, 0)$
- $(x_{s,1}, x_{s,2}, x_{1,t}, x_{2,t}) = (0, 0, 1, 1)$
- ...

Shortest path: Integrality (cont.)

When we solve an LP, the objective function is being pushed to a vertex of the feasible region, or a line (hyperplane) on its border.
Since the variables are non-integral, the objective plane must parallel with one of the borders.
There is an infinite family of solutions where we send an amount of flow y through the top path and $1 - y$ through the bottom half.



However, since the LP solving method (like simplex) requires that we end up at a corner point, the LP solution will always be integral (one of the points on the pictured hypercube).

Integrality: extension from shortest path to max flow

The same logic holds for max flow:

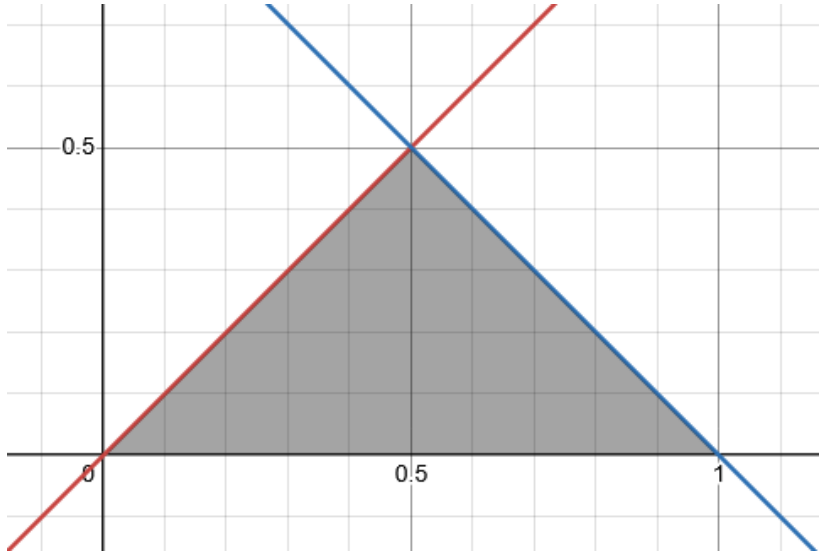
- If all the capacities are integer (which they are, since C's range is $\mathbb{N}$), all of the extreme points for the polytope will be integer, since the decision variables are sufficiently independent of one another
- Thus, the solution to the max flow LP will always have integer decision variables

Caveat on integrality

Important note: It is **not always the case** that having all integer constraints will result in an all-integer solution to an LP.

Consider this LP and corresponding feasible region:

$$\begin{aligned} \max \quad & x_2 \\ s.t. \quad & x_1 - x_2 \geq 0 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$



Here, the only optimal solution is $x_1 = x_2 = 0.5$.

Integrality criteria: TU A and integer b

In order for the **integrality property** to hold, the LP must obey certain properties.

Specifically, b must be integer, and the constraint matrix A must be "totally unimodular" (TU). Total unimodularity basically it means that (as discussed) all the extreme points in the feasible region of the LP have all-integer values.

Criteria for TU (A must satisfy all 3):

1. All entries are 0, 1, or -1
2. Each column contains at most two nonzero entries
3. You can partition the rows into two sets (R_1 and R_2) such that:
   - If a column has two entries of the same sign, one row is in R_1 and the other is in R_2
   - If a column has two entries of opposite signs, both rows are in the same set (R_1 or R_2)

We won't focus on the partitioning part, but the idea is that the -1's and $+1$'s "cancel out" to get -1, 0, or 1.

Integrality of shortest path and max flow (w.r.t. new criteria)

Both shortest path and max flow have a TU constraint matrix A and vector b.

For shortest path, we have the following structure for A and b:

	$x(s,i)$	$x(i,k)$	$x(k,t)$	$\ldots$	b
Source Node s	$+1$	0	0	$\ldots$	1
Intermediate Node i	-1	$+1$	0	$\ldots$	0
Intermediate Node k	0	-1	$+1$	$\ldots$	0
Sink Node t	0	0	-1	$\ldots$	-1

We can incorporate the capacity constraints as well (by concatenating the above matrix with $[I_n \mid b_C]$, where b_C is the vector of capacities) and have the result be totally unimodular as well, but the mechanics of that are out of scope.

Summary

- Max flow can be modeled intuitively as an LP
- Using the same ideas behind max flow's model, we can model shortest path as an LP
- Max flow (and by extension shortest path) have totally unimodular constraint matrices A and integer constraint vectors b, so their LPs have the integrality property
- The integrality property essentially states that the extreme points of the feasible region of an LP will have all integer values for all decision variables
- By the integrality property, the values of the decision variables for an optimal LP solution will all be integer, ensuring that our shortest path implementation works without the "0.5 units of flow" issue