HW 2 grades out after lecture

HW2 problem 3a will be made extra credit
b/c we accepted some incorrect answers. - will update soln's with examples.

HW 3 solutions also will be up by noon

HW 4 written due tonight + HW 3 programming

HW 5 + ethics 2 out tonight.

More Linear Programming

CSE 417 Winter 21

Lecture 17

Example Problem

Gardens each get enough soil:

$x_{A,1} + x_{B,1} \geq 4$

$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$

$x_{A,3} + x_{C,3} \geq 1.5$

$x_{B,4} + x_{C,4} \geq 2.5$

Can't overuse a pile:

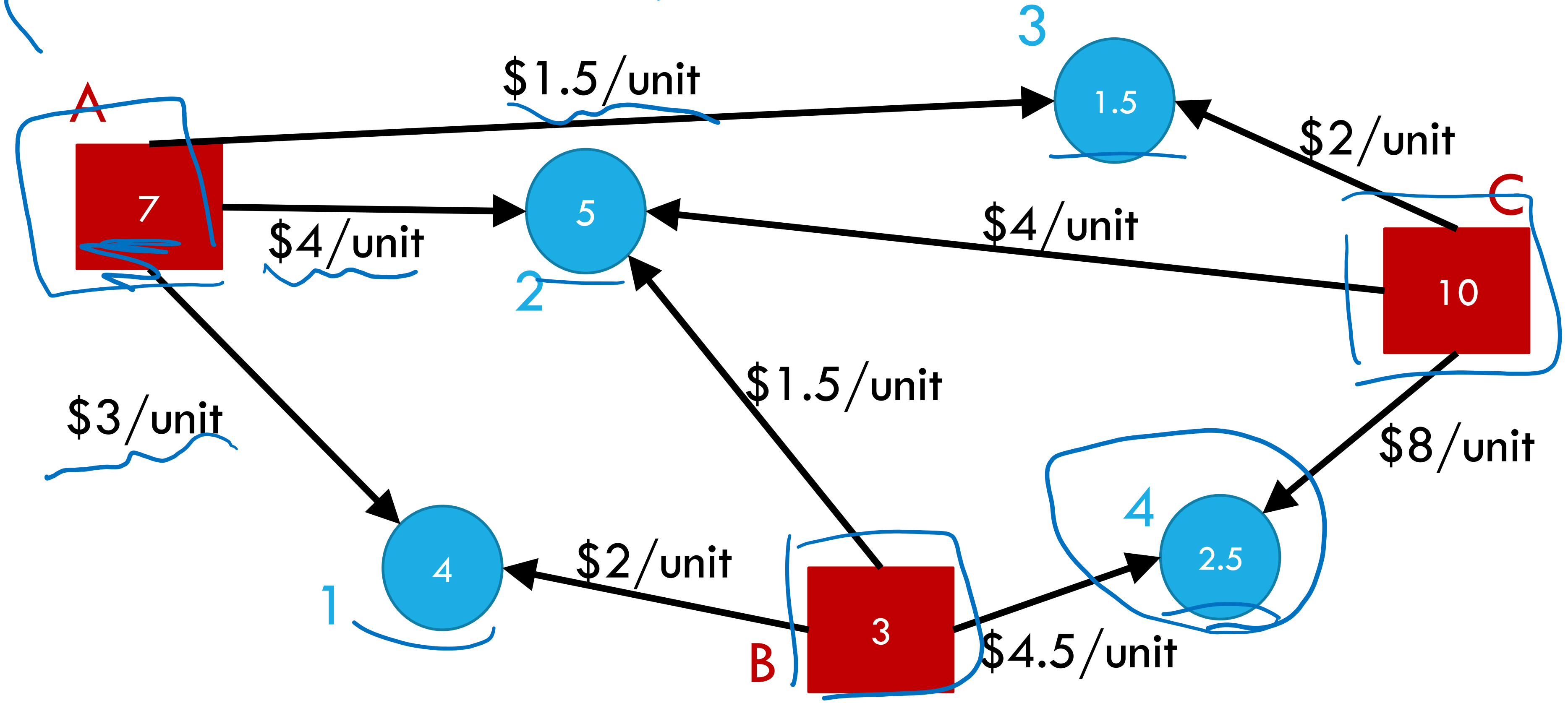
$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$

$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$

$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$

No anti-soil:

$x_{i,j} \geq 0$ for all i, j



Full Definition

Minimize: $(x_{A,1} \cdot 3 + x_{A,2} \cdot 4 + x_{A,3} \cdot 1.5) + (x_{B,1} \cdot 2 + x_{B,2} \cdot 1.5 + x_{B,4} \cdot 4.5) + (x_{C,2} \cdot 4 + x_{C,3} \cdot 2 + x_{C,4} \cdot 8)$

Subject To:

$x_{A,1} + x_{B,1} \geq 4$

$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$

$x_{A,3} + x_{C,3} \geq 1.5$

$x_{B,4} + x_{C,4} \geq 2.5$

$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$

$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$

$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$

$x_{i,j} \geq 0$ for all i, j

A Linear Program

A linear program is defined by:

Real-valued **variables**

Subject to a list of **linear constraints**

A linear constraint is a statement of the form: $\sum a_i x_i \leq c_i$
where a_i are constants, the x_i are variables and c_i is a constant.

Maximizing or minimizing a linear objective function

A linear objective function is a function of the form: $\sum b_i x_i$
where b_i are constants and the x_i are variables.

Linear constraints

Can you write each of these requirements as linear constraint(s)?

Some of these are tricks…

x_i times x_j is at least 5	No way to represent ☹
$5x_i$ is equal to 1	$5x_i \leq 1$ and $-5x_i \leq -1$
$x_i \leq 5$ OR $x_i \geq 7$	No way to represent ☹
x_i is non-negative.	$x_i \geq 0$
x_i is an integer.	No way to represent ☹

Full Definition

Minimize: $(x_{A,1} \cdot 3 + x_{A,2} \cdot 4 + x_{A,3} \cdot 1.5) + (x_{B,1} \cdot 2 + x_{B,2} \cdot 1.5 + x_{B,4} \cdot 4.5) + (x_{C,2} \cdot 4 + x_{C,3} \cdot 2 + x_{C,4} \cdot 8)$

Subject To:

$x_{A,1} + x_{B,1} \geq 4$

$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$

$x_{A,3} + x_{C,3} \geq 1.5$

$x_{B,4} + x_{C,4} \geq 2.5$

$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$

$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$

$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$

$x_{i,j} \geq 0$ for all i, j

What are we looking for?

A solution (or point) is a setting of all the variables

A **feasible point** is a point that satisfies all the constraints.

An **optimal point** is a point that is feasible and has at least as good of an objective value as every other feasible point.

Example Problem

Gardens each get enough soil:

$$x_{A,1} + x_{B,1} \geq 4$$
$$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$$
$$x_{A,3} + x_{C,3} \geq 1.5$$
$$x_{B,4} + x_{C,4} \geq 2.5$$

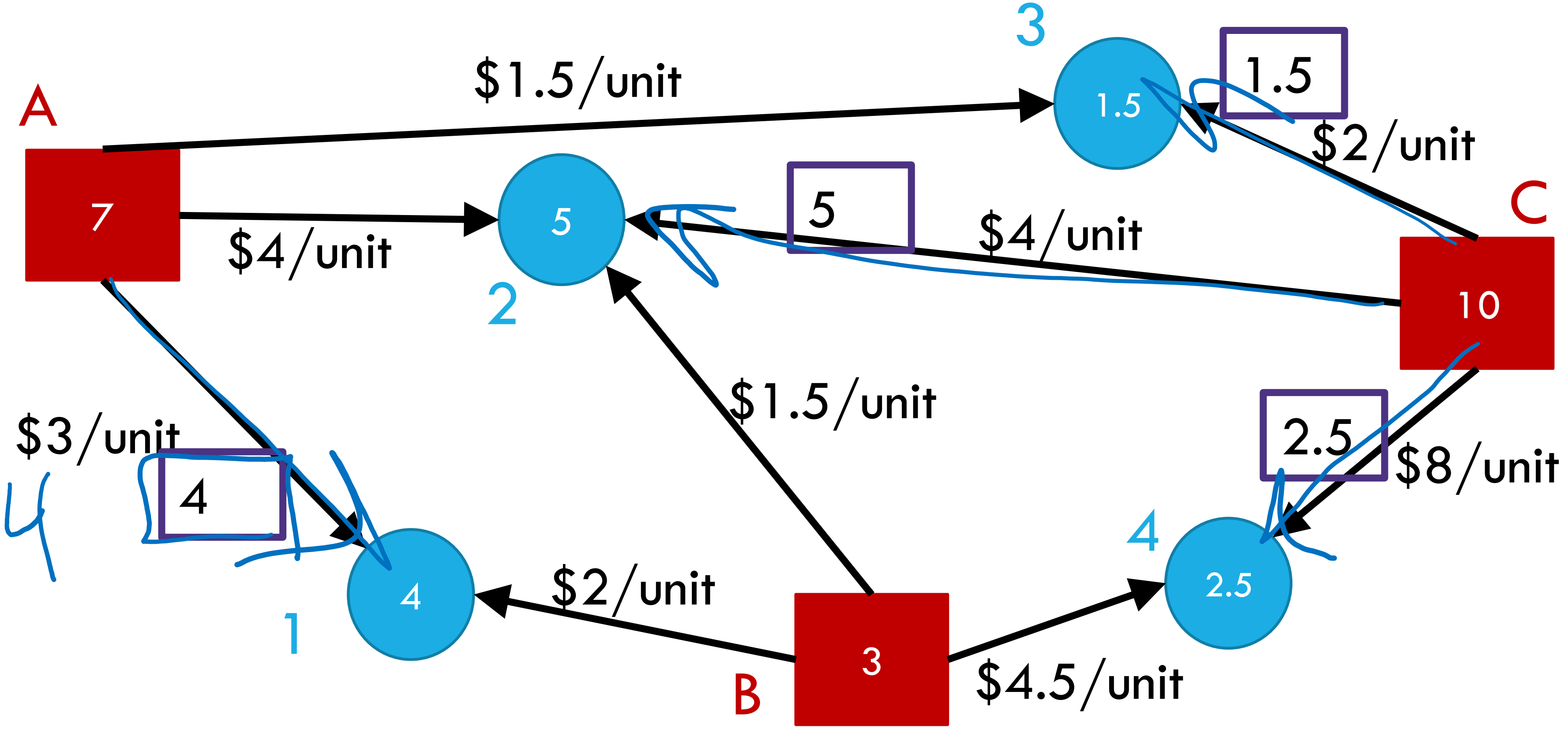
Can't overuse a pile:

$$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$$
$$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$$
$$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$$

No anti-soil:

$x_{i,j} \geq 0$ for all i, j

A feasible point.
Objective: 55



$x_{A,1}=4$

Example Problem

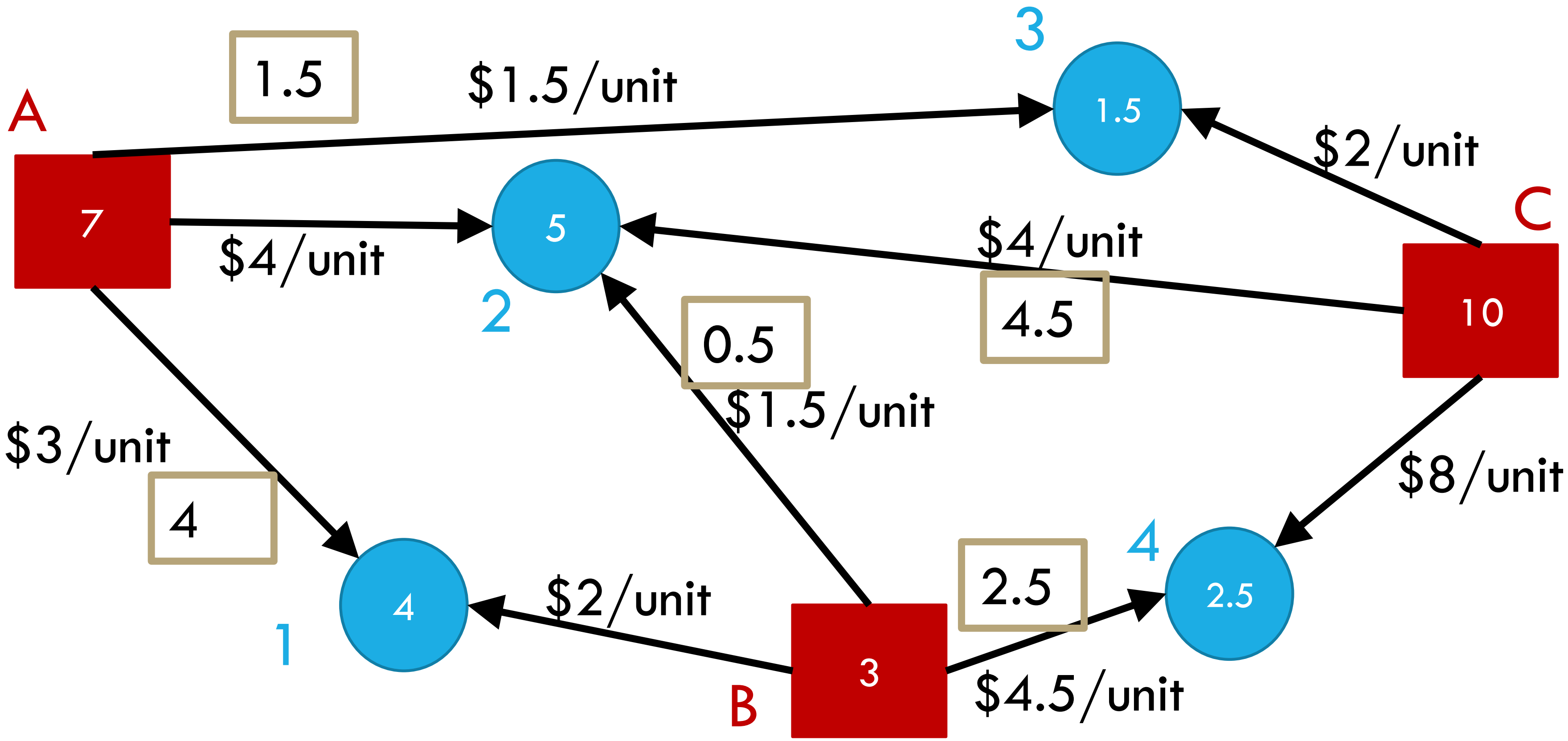
Gardens each get enough soil:
$x_{A,1} + x_{B,1} \geq 4$
$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$
$x_{A,3} + x_{C,3} \geq 1.5$
$x_{B,4} + x_{C,4} \geq 2.5$

Can't overuse a pile:
$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$
$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$
$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$

No anti-soil:
$x_{i,j} \geq 0$ for all i, j

A feasible point.
Objective: 44.25

This is an optimal point. There are others!



Example Problem

Gardens each get enough soil:

$x_{A,1} + x_{B,1} \geq 4$
$x_{A,2} + x_{B,2} + x_{C,2} \geq 5$
$x_{A,3} + x_{C,3} \geq 1.5$
$x_{B,4} + x_{C,4} \geq 2.5$

Can't overuse a pile:

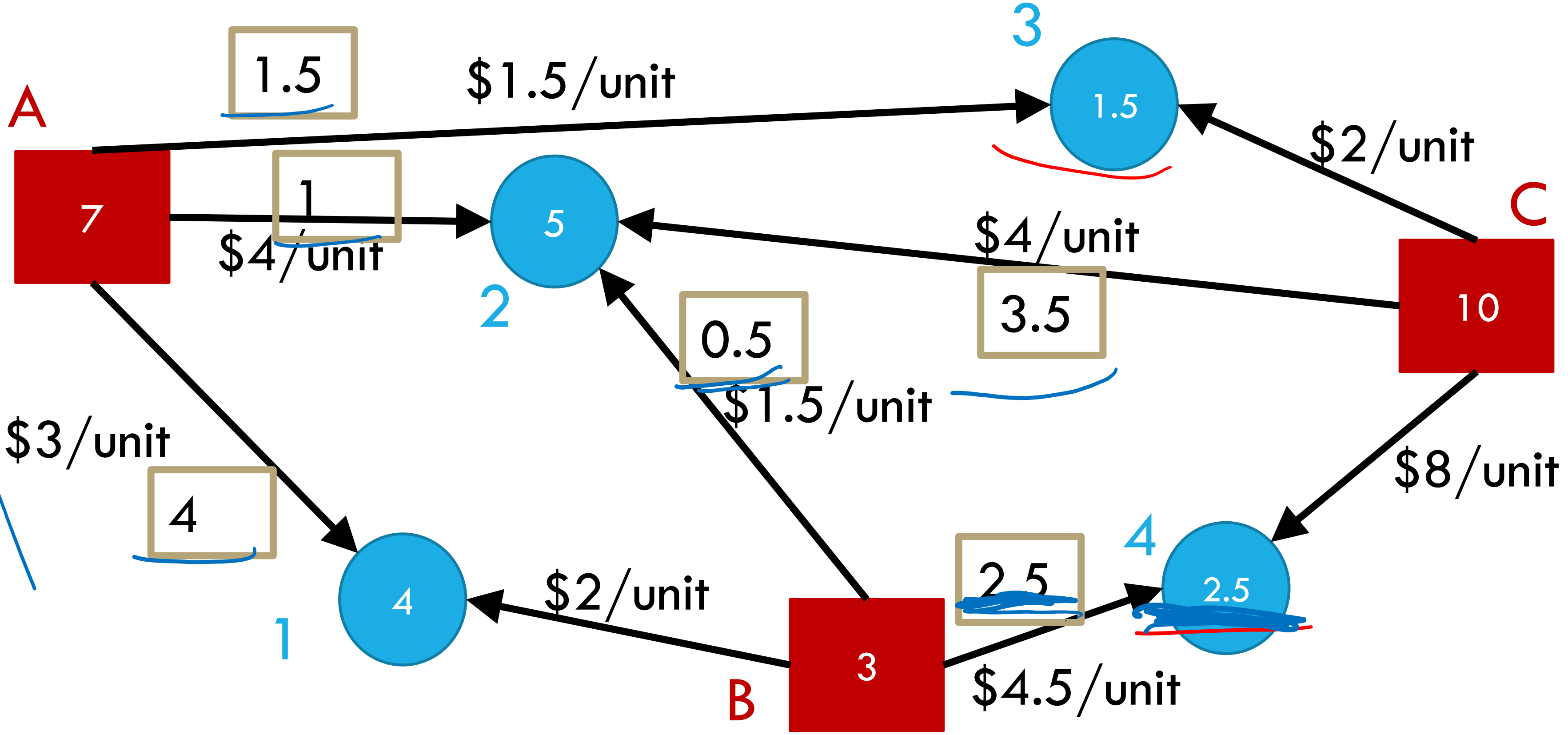
$x_{A,1} + x_{A,2} + x_{A,3} \leq 7$
$x_{B,1} + x_{B,2} + x_{B,4} \leq 3$
$x_{C,2} + x_{C,3} + x_{C,4} \leq 10$

No anti-soil:

$x_{i,j} \geq 0$ for all i, j

A feasible point.
Objective: 44.25

Here's another optimal point!!



Solving LPs

For this class, we're only going to think about library functions to solve linear programs (i.e. we won't teach you how any of the algorithms work)

The most famous is the **Simplex Method** – can be quite slow (exponential time) in the worst case. But rarely hits worst-case behavior.

Very fast in practice. Idea: jump from extreme point to extreme point.

The **Ellipsoid Method** was the first theoretically polynomial time algorithm $O(n^6)$ where n is the number of bit needed to describe the LP (usually ≈ the number of constraints)

Interior Point Methods are faster theoretically, and starting to catch up practically. $O(n^{2.373})$ theoretically

Another Question

Change the problem

Instead of infinitely divisible dirt…

What if instead we're moving whole unit things (the dirt is in bags we can't open or we're moving bikes or plants or anything else that can't be split)

Or if we're assigning people to shifts (can have 1/3 of a person on a shift)

Well, the constraints will change (your "demand" and "supplies" will be integers)

Non-Integrality

Some linear programs only have optimal solutions that have some (or all) variables as non-integers (even with only integers in the objective function and constraints).

For dirt or water or anything arbitrarily divisible, no big deal!

For cell phones or bicycles…only possibly a big deal!

- In practice: if the optimal thing to manufacture 999,999.8 widgets per day, rounding up or down probably isn't going to make a huge difference in your profits.
- But sometimes rounding isn't ok…

What do you do if you need integers?

Integer Programs are linear programs where you can mark some variables as needing to be integers.

In practice – often still solvable (Excel also has a solver for these problems). But no longer guaranteed to be efficient.

Lots of theory has been done for when the optimum will be all integers. (MATH 407 or MATH 514)

But sometimes you get a fractional solution…what can you do?

Extra Practice

You have 20 pounds of gold and 40 pounds of silver.

You can turn 2 pounds of silver and 3 pound of gold into a (really heavy) necklace that can be sold for $\$10$.

You can also turn 9 pounds of silver and 1 pound of gold into a (really fancy) shield that can be sold for $\$15$.

How many of each should you make to maximize your profit? (fractional values are ok for this problem

Extra Practice

You have 20 pounds of gold and 40 pounds of silver.

You can turn 2 pounds of silver and 3 pound of gold into a (really heavy) necklace that can be sold for $\$10$.

You can also turn 9 pounds of silver and 1 pound of gold into a (really fancy) shield that can be sold for $\$15$.

How many of each should you make to maximize your profit?

Max $10N + 15S$

Subject to

$2N + 9S \leq 40$

$3N + S \leq 20$

Plugging into an LP solver would give $N = 5.6$ and $S = 3.2$
(we'll give resources for solvers next lecture)

An Example Where things go wrong

Independent Set

A set S of vertices is an independent set if for all u, v in S, (u, v) is not an edge.
(i.e. every edge has at most one endpoint in the set)

Sound familiar?

This isn't a vertex cover – that was at least on endpoint per edge.

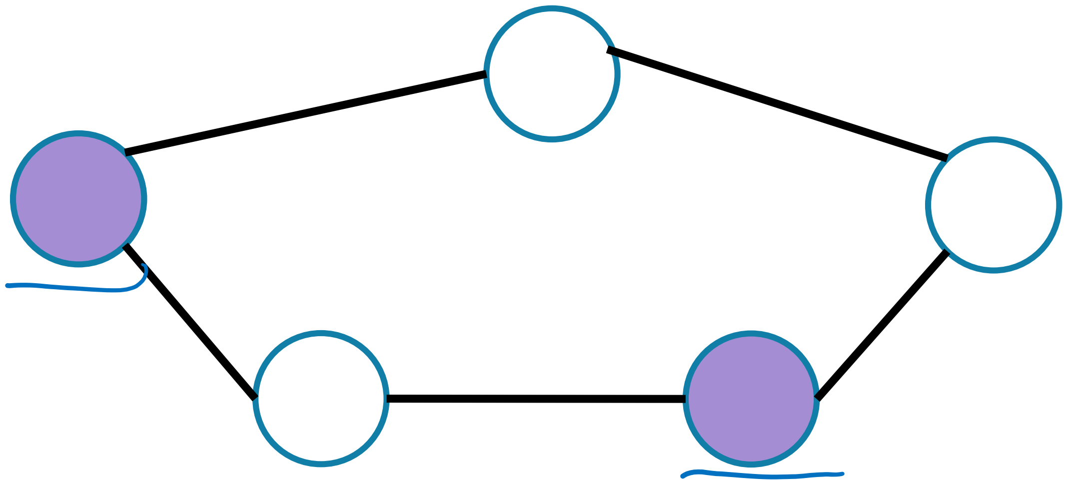
A 2-coloring divides vertices into two independent sets (i.e. all the "red" vertices are an independent set, all the "blue" vertices are another independent set).

Independent Set

Independent Set

A set S of vertices is an independent set if for all u, v in S, (u, v) is not an edge.
(i.e. every edge has at most one endpoint in the set)

An independent set of size 2.

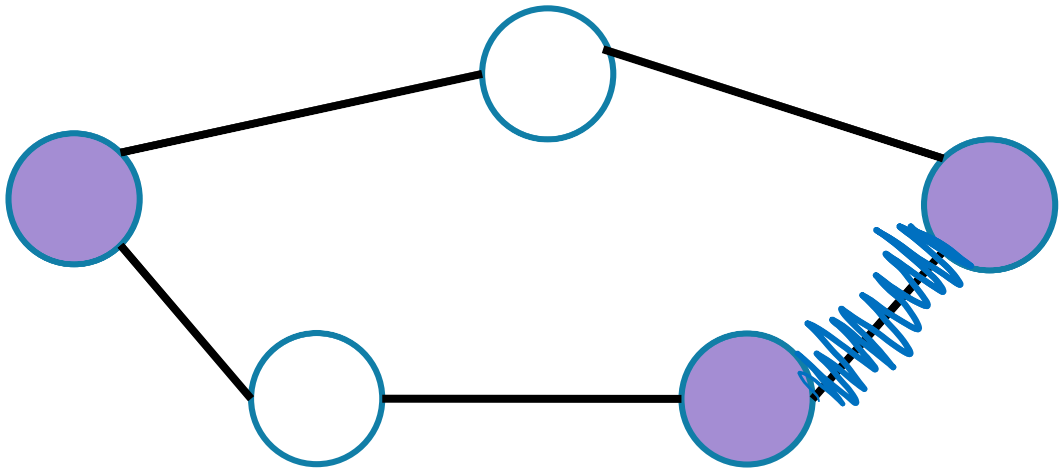


Independent Set

Independent Set

A set S of vertices is an independent set if for all u, v in S, (u, v) is not an edge.
(i.e. every edge has at most one endpoint in the set)

Not an independent set.

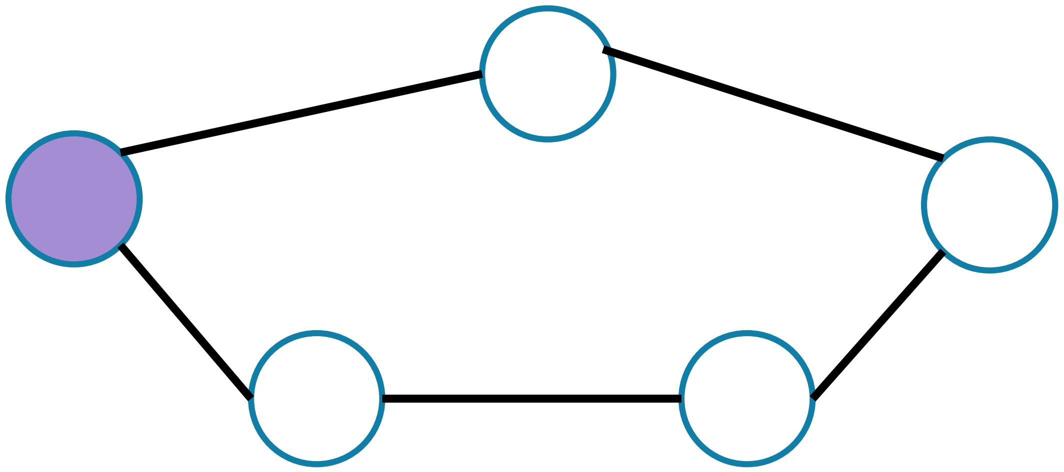


Independent Set

Independent Set

A set S of vertices is an independent set if for all u, v in S, (u, v) is not an edge.
(i.e. every edge has at most one endpoint in the set)

An independent set of size 1.



Write an LP for independent set

What do you want your variables to be?

How do you ensure that you don't have two adjacent vertices in the set?

Write an LP for independent set

What do you want your variables to be?

How do you ensure that you don't have two adjacent vertices in the set?

Have a variable for each vertex x_u -- have it **indicate** whether you include u or not (i.e. make it a Boolean)

Constraints?

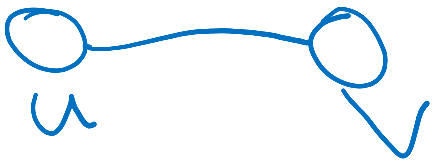
Write an LP for independent set

What do you want your variables to be?

How do you ensure that you don't have two adjacent vertices in the set?

Have a variable for each vertex x_u -- have it **indicate** whether you include u or not (i.e. make it a Boolean)

Constraints?



If (u, v) is an edge then $x_u + x_v \leq 1$

If those are Booleans, this definitely works. If they aren't, well it still kinda makes sense at least.

Write an LP for independent set

What do you want your variables to be?

How do you ensure that you don't have two adjacent vertices in the set?

Have a variable for each vertex x_u -- have it **indicate** whether you include u or not (i.e. make it a Boolean)

Constraints?

If (u, v) is an edge then $x_u + x_v \leq 1$

$0 \leq x_u \leq 1$ for all u (closest we can get to saying we have a Boolean)

Objective? Max $\sum x_u$

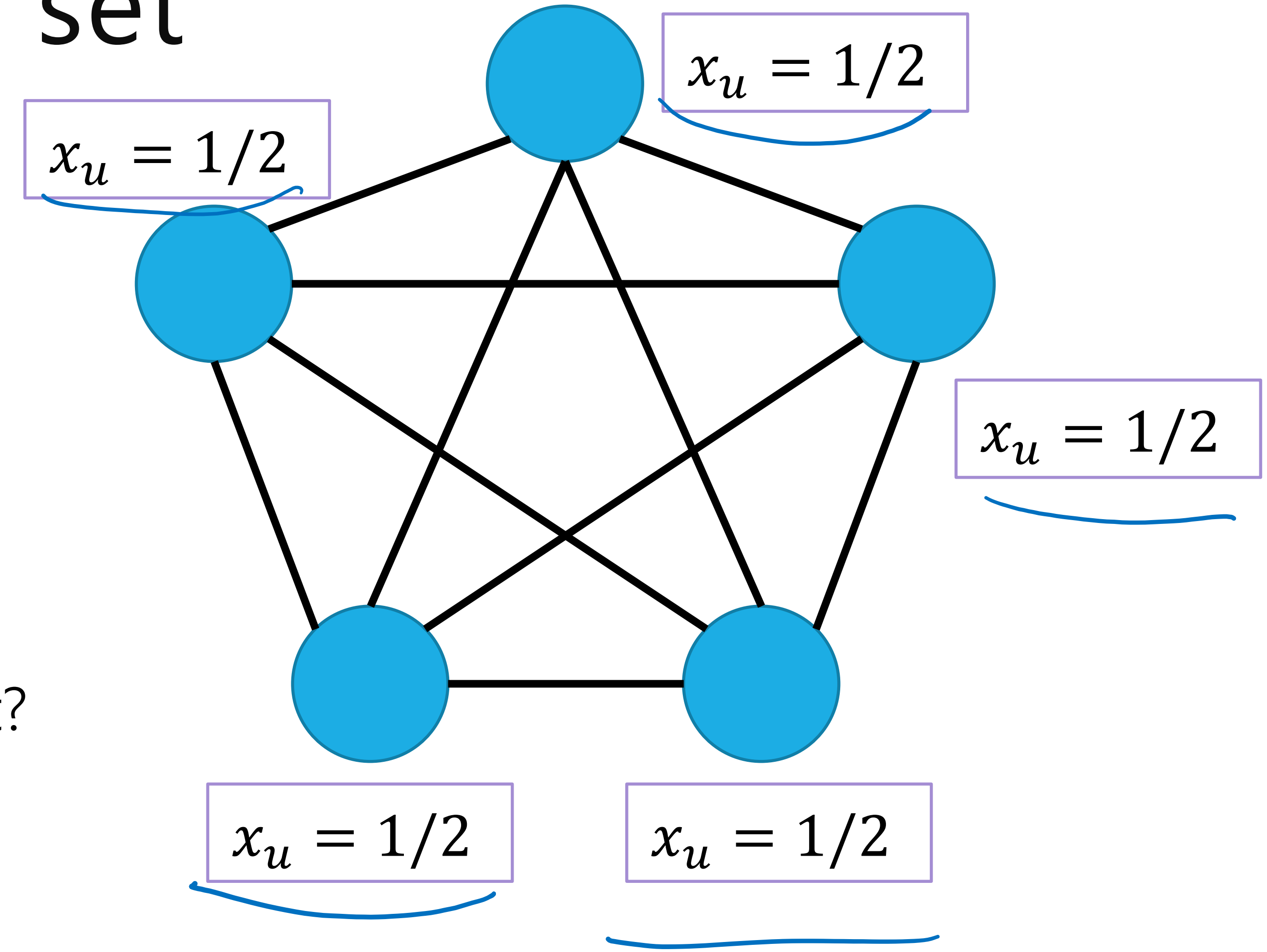
LP for Independent set

Max: $\sum x_u$

Subject to:

$x_u + x_v \leq 1$ for all $(u, v) \in E$

$0 \leq x_u \leq 1$ for all u



How big is the biggest independent set?

1

What does the LP find

2.5 – there's no "real" independent set this corresponds to.

LP for Independent set

Max: $\sum x_u$

Subject to:

$x_u + x_v \leq 1$ for all $(u, v) \in E$

$0 \leq x_u \leq 1$ for all u

For this problem an LP is not a useful tool.
If you get a fractional solution, there's no known process to turn it into an integral one without ending up far from the real best.

A nicer example

Sometimes we can round fractional solutions into integral ones.

Minimum Weight Vertex Cover

We've seen how to solve the problem with DP on trees.

Let's try it now with linear programming.

> A set S of vertices is a vertex cover if for every edge (u, v), u is in S, v is in S or both are in S.

"at least one endpoint"

Vertex Cover LP

Fill out the poll everywhere for Activity Credit!
Go to pollev.com/cse417 and login with your UW identity

Write an LP for finding the minimum weight vertex cover

A set S of vertices is a vertex cover if for every edge (u, v), u is in S, v is in S or both are in S.

What are your variables, then how do you constrain them?

Let $w(u)$ be the weight for a vertex u. You can treat $w(u)$ as a constant.

Vertex Cover LP

Minimize $\sum w(u) \cdot x_u$

Subject to:

$x_u + x_v \geq 1$ for all $(u, v) \in E$

$0 \leq x_u \leq 1$ for all u.

Integrality

We need an integral solution

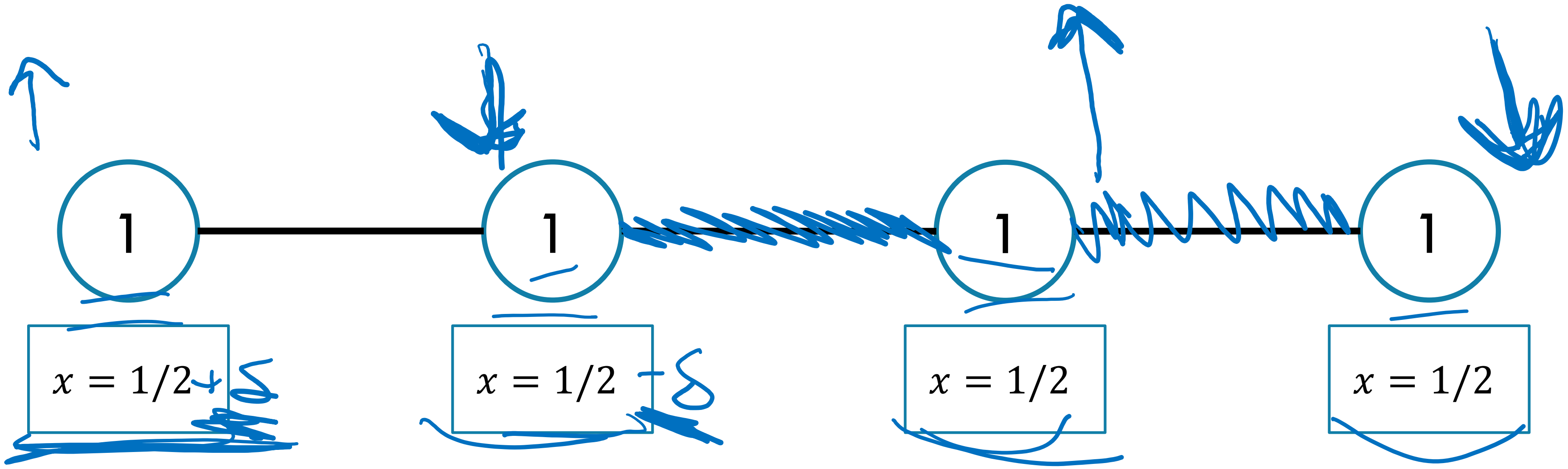
Having $\frac{1}{3}$ of u in the set doesn't make sense.

How do we make the variables integral?

What do we do

Let's try an example first

Suppose your LP gave you this solution on this graph. How would you round it (i.e. convert to a valid vertex cover)?



What do we do

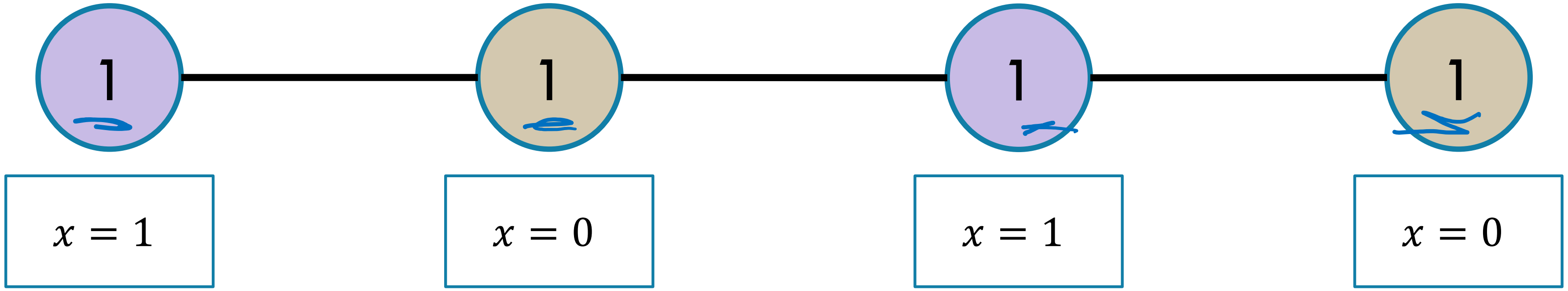
Increase x for the purple vertices, and decrease x for the gold vertices.

(at the same time at the same rate)

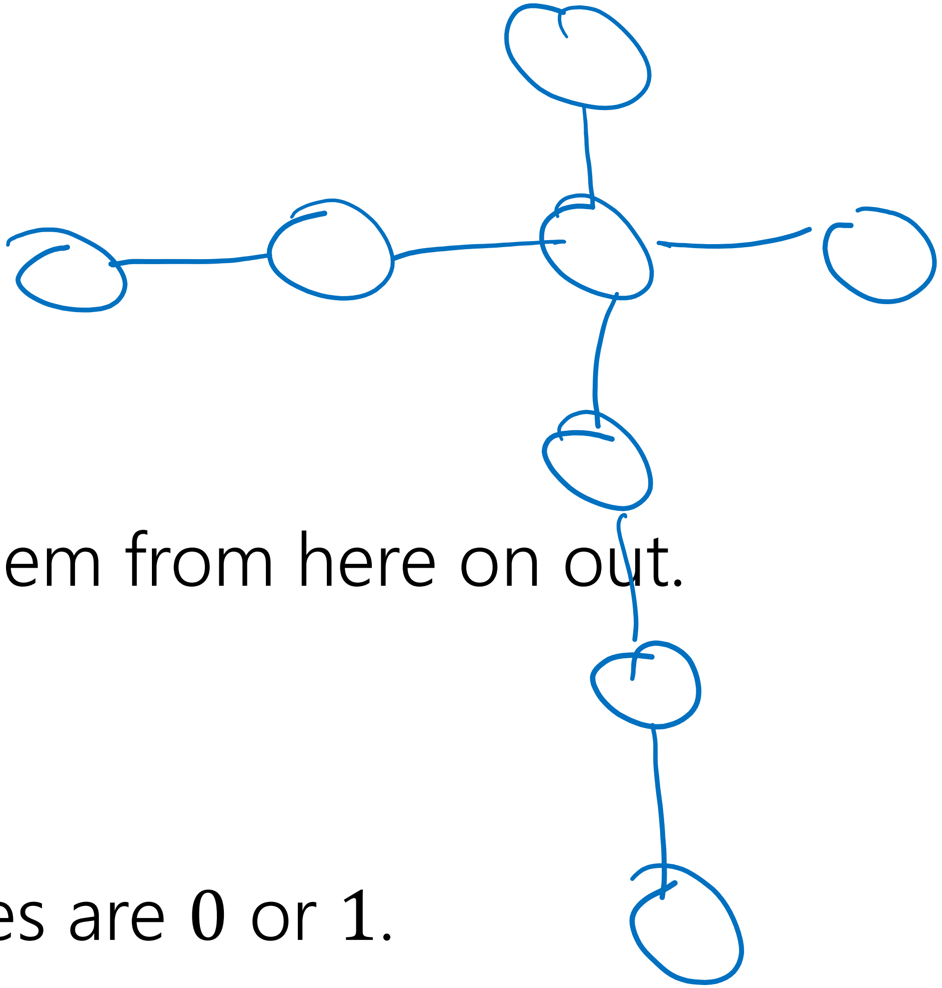
Every edge (in our example) has a purple and gold endpoint, so every constraint is still satisfied.

The objective (in our example) increases and decreases at the same rate.

So we still have an optimal (minimum) vertex cover



What do we do



Those vertices are done now!

They're integers – no need to change them from here on out.

Ignore all the vertices where the variables are 0 or 1.

How do we handle the ones that are left…what if they're not a nice simple path?

In General

If we have more than a path, we have to be careful when changing values.

If we decrease the value at u, we need to be sure **every** edge incident to u has its other vertex increase.

Otherwise an edge might be uncovered (we might not have a valid solution to the LP anymore).

So every neighbor of a gold vertex must be purple.

…does that sound familiar?

In General…

2-color the graph (call the vertices "purple" or "gold")

Increase all the purple vertices by some value δ

And decrease all the gold vertices by the same value δ

Choose δ so that we set at least one variable to 0 or 1 (but don't move any variables outside the $[0,1]$ range allowed.

Those vertices that just got set to 0 or 1 can be deleted. Start over with the remaining graph.

In General…

But wait!

What if we're increasing the objective value? (i.e. what if there's more weight on the purple vertices than gold).

We won't increase:

If we were, then switch purple and gold. Then we'd be **decreasing** the objective…but we were at the minimum!

So we're really getting a minimum vertex cover.

Running Time

Regardless of which LP solver we're using, n or fewer BFSs is going to be less than the LP solver (in big-O terms)

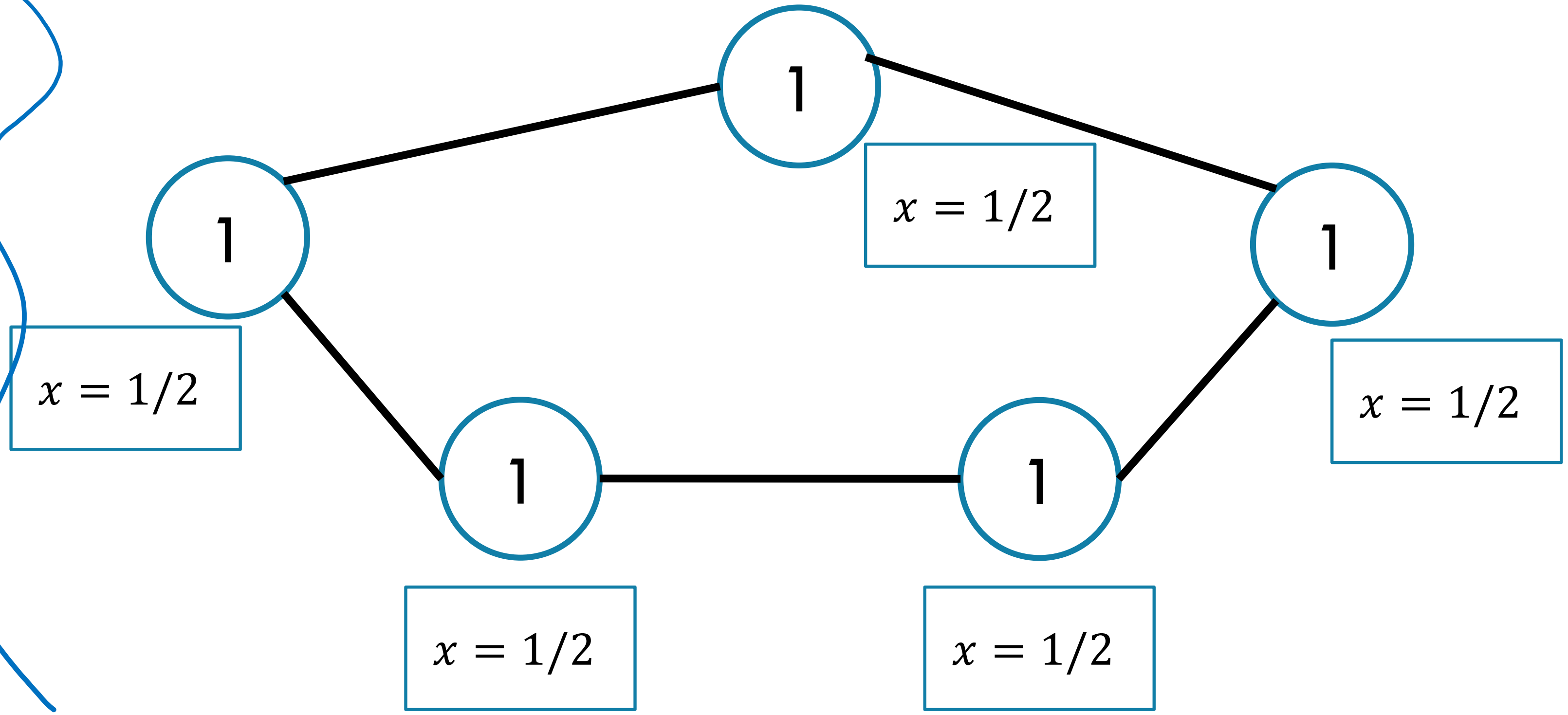
We won't ask you to precisely analyze running times of LPs (depends a lot on which library you're using, whether you have more variables or constraints, whether your constraints have lots of variables,…)

We will check that it's polynomial time: if you have polynomially many variables and constraints, then it's polynomial time.

Non-Bipartite

We needed the graph to be bipartite to be able to 2-color it.

What if our original graph isn't bipartite?



The LP finds a fractional vertex cover of weight 2.5

There's no "real"/integral VC of weight 2.5. – lightest is weight 3.

There's a "gap" between integral and fractional solutions.

Summary

With dynamic programming, we could find the minimum weight vertex cover on trees.

With linear programming, we can find the minimum weight vertex cover on any bipartite graph (trees and other graphs!).

But LPs don’t always give you a fractional solution that’s helpful for finding an integral one. On non-bipartite graphs, we’ll need to do something else. (More in a few weeks).

Side Note for Those with deep LP knowledge

If you know what a “basic feasible solution” of an LP is…

The “basic feasible solutions” of the vertex cover LP has the property that every element is either 0, ½ or 1.

It turns out on bipartite graphs, a basic feasible solution will only have variables set to $0,1.$ i.e. a basic feasible solution is automatically integral.

So if your LP solver finds one of those by default (like the simplex algorithm), you won’t need the rounding step for bipartite graphs.

But this is still a useful exercise for prepping for more advanced rounding ideas in a few weeks.

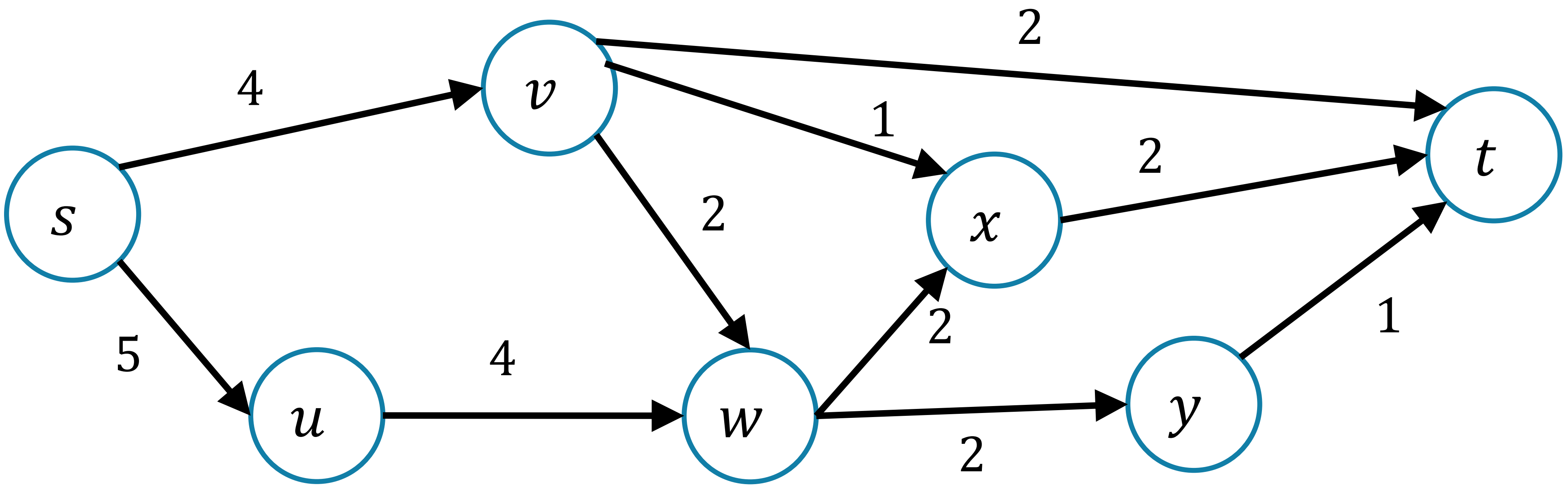
Flows

Max Flow

We have a directed graph G, a source vertex s and a target vertex t.

We have some thing (water or data packets) we have to send from s to t.

Every edge has a capacity, it can only handle so many.

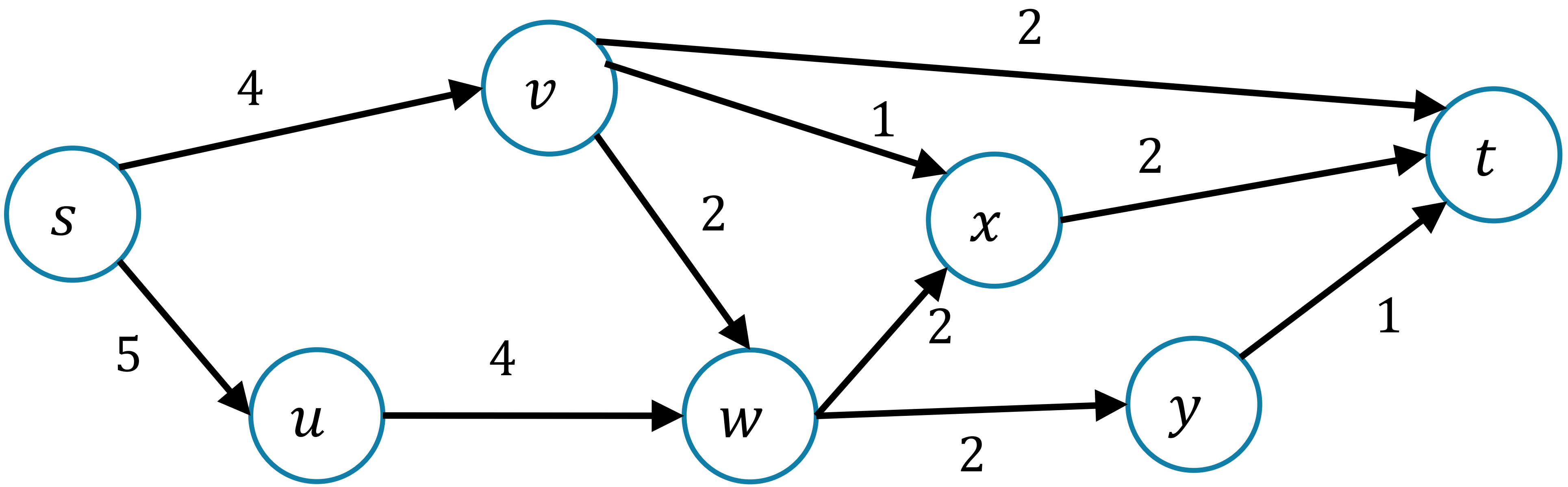


Flows

A **flow** moves units of water from s **to** t.

Water can only be created at s and only disappear at t.

And you cannot move more water than the capacity on any edge.

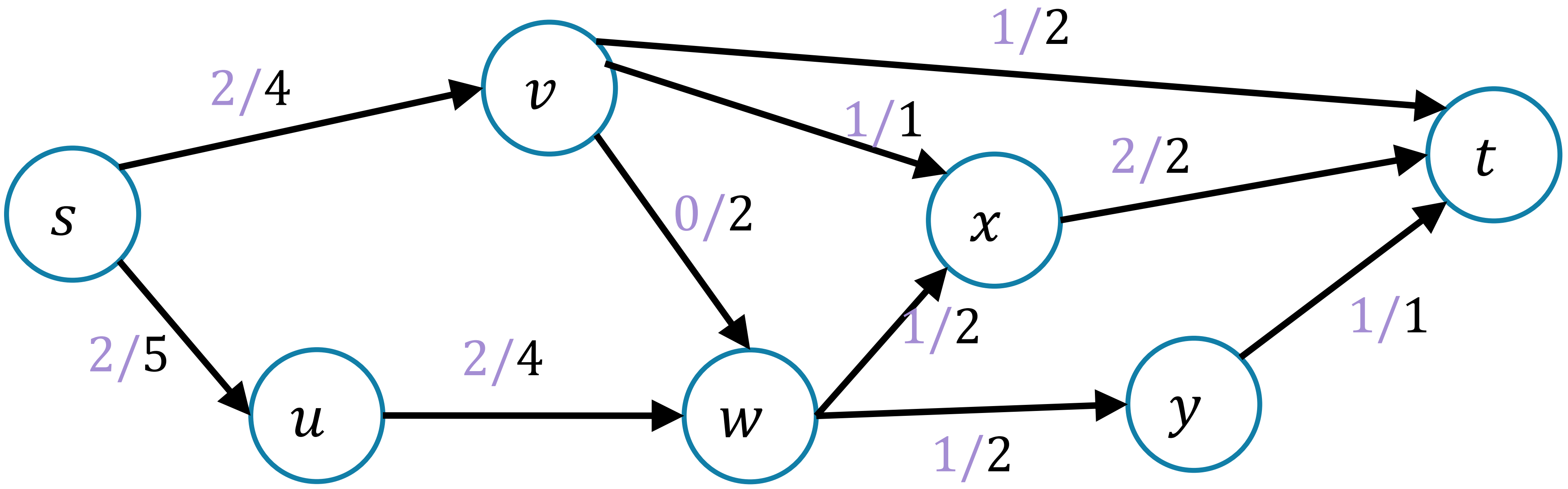


Flows

A **flow** moves units of water from s to t.

Water can only be created at s and only disappear at t.

And you cannot move more water than the capacity on any edge.



Write an LP for finding the biggest flow

Let $c(e)$ be the capacity on edge e

Write an LP for finding the biggest flow

Let $c(e)$ be the capacity on edge e

Max $\sum_{e:e \text{ enters } t} x_e$

Subject to:

$\sum_{e:e \text{ enters } u} x_e = \sum_{e:e \text{ leaves } u} x_e$ for every u other than s and t

$0 \leq x_e \leq c(e)$ for every edge e

Next Time

How to solve this problem without linear programming (an algorithm that runs directly on the graph)