Max Flow Round 2

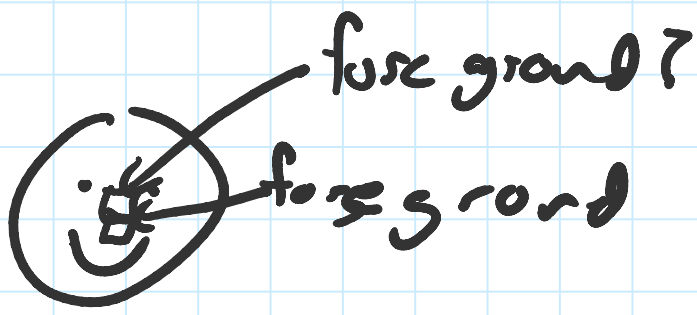
Wednesday, April 11, 2018 5:58 PM

Objs

Image Segmentation

given the probability that a pixel belongs to the background/foreground.

Maximize the probability that a set of pixels belongs to the foreground

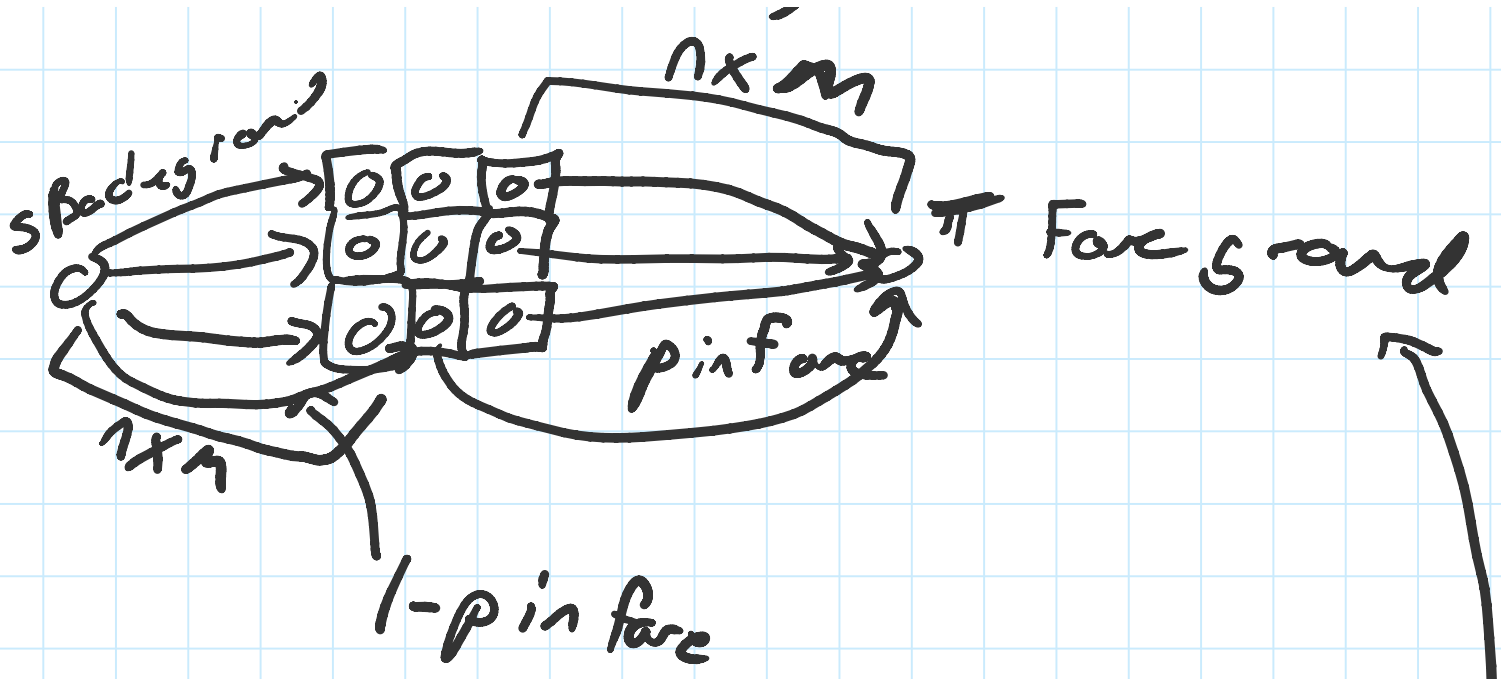


all in foreground

center pixel most likely belongs to foreground

p_{ij} if i and j are adjacent and similar p_{ij} is going larger.

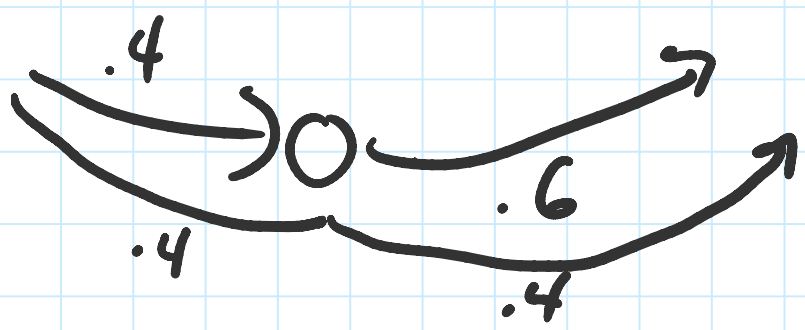
$n \times m$



add pixel to the fore ground

if its edge to sink

is not saturated

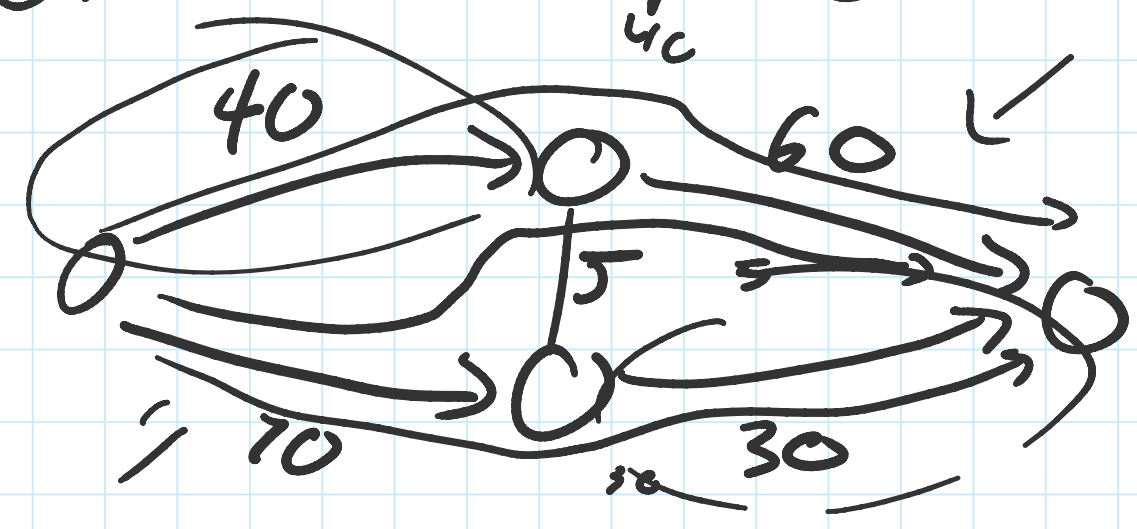


connect each node with

each other using their

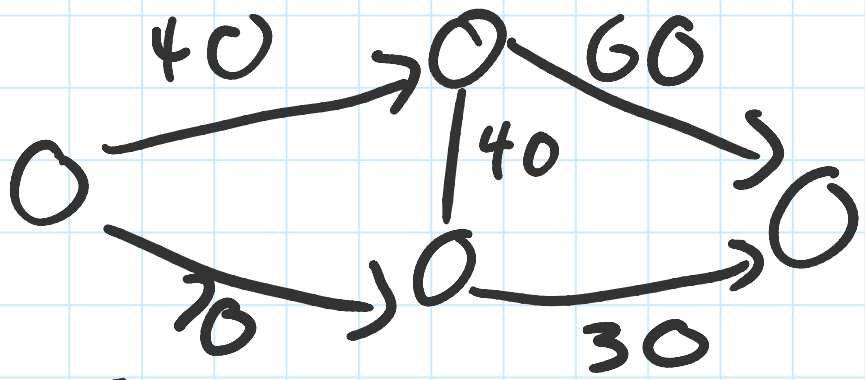
correlated probabilities

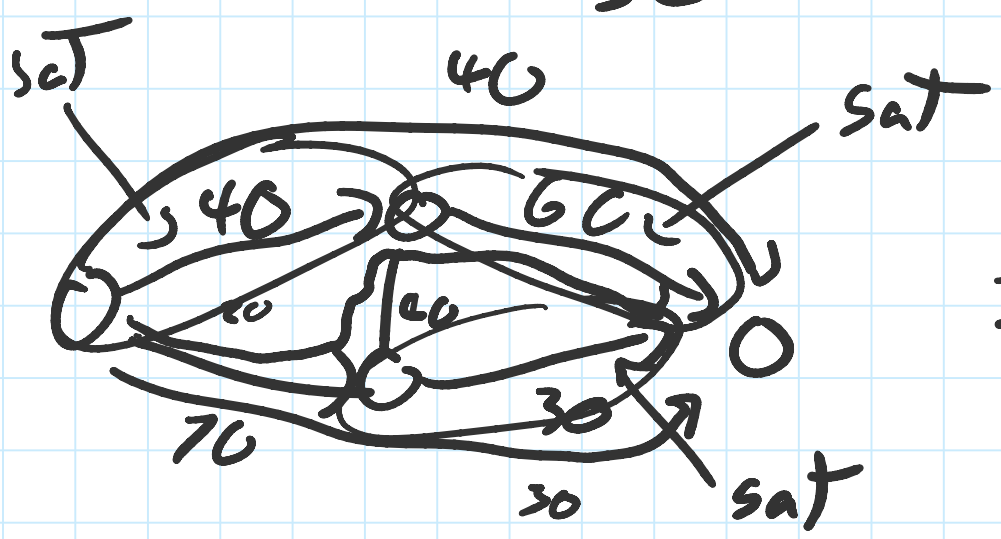
Uncorrelated pixels



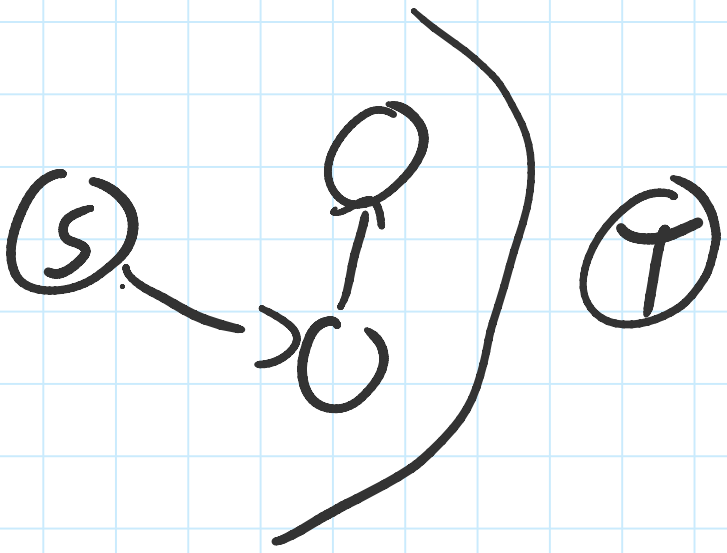
75

Corrected pixels





30+20+40=
90



Basketball elimination

Set of teams

Set of games played

T_1, T_2 T_1

T_2 T_4 T_2

T_4 T_3 T_3

Set of games yet to be played

be played

T_1 T_4

T_3 T_4

T_2 T_3

Find all teams that cannot possibly have the highest number of wins regardless of the result of the remaining games

	T_1 T_2	T_1	
	T_2 T_4	T_2	
	T_4 T_3	T_3	
to play	T_1 T_4	T_4	1-4
	T_3 T_1		1-3
	T_2 T_3		2-3

Consider team 1

can it win enough

can it win enough

Brute force approach

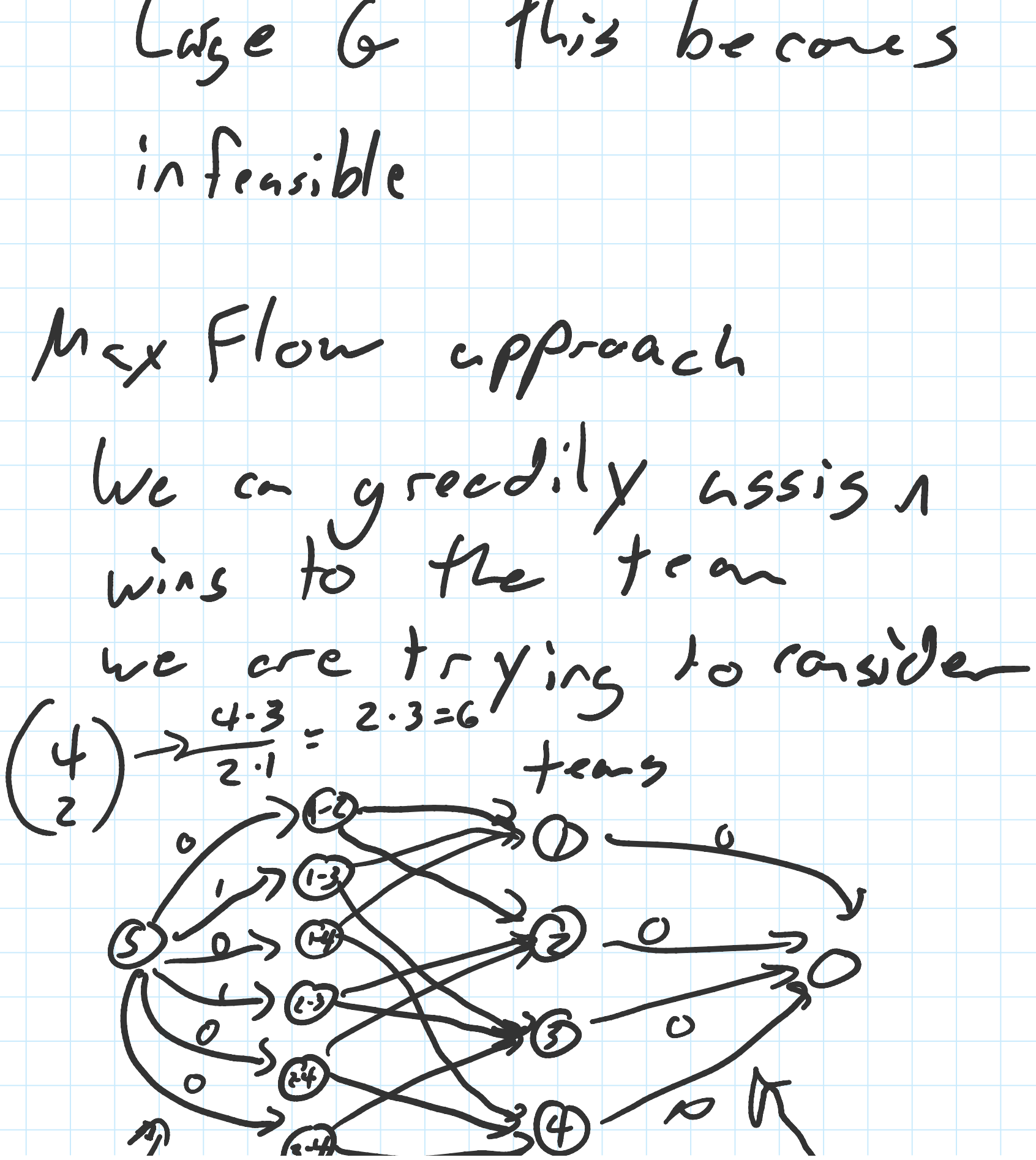
G left to play

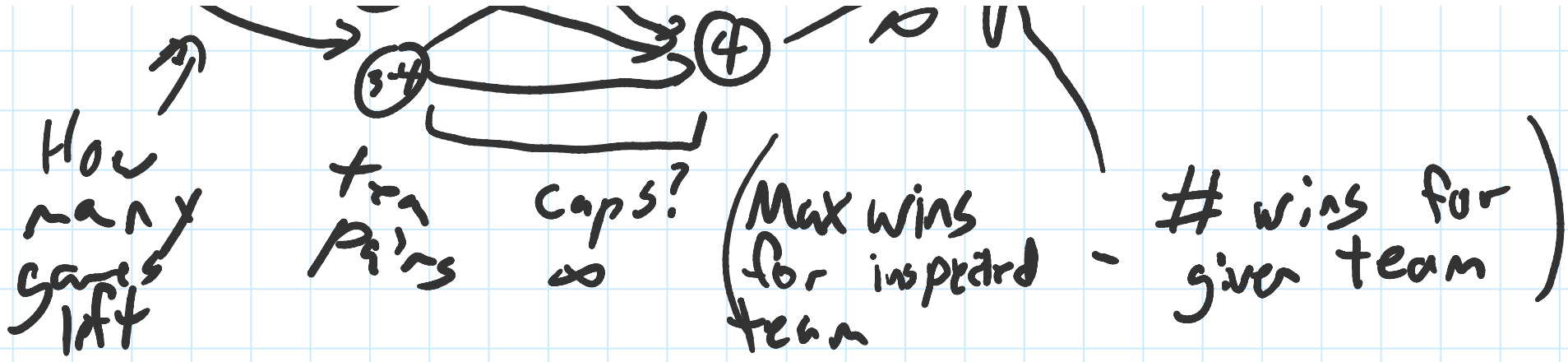
2^G maybe prune bad paths using backtracking

Large G this becomes infeasible

Max Flow approach

We can greedily assign wins to the team we are trying to consider

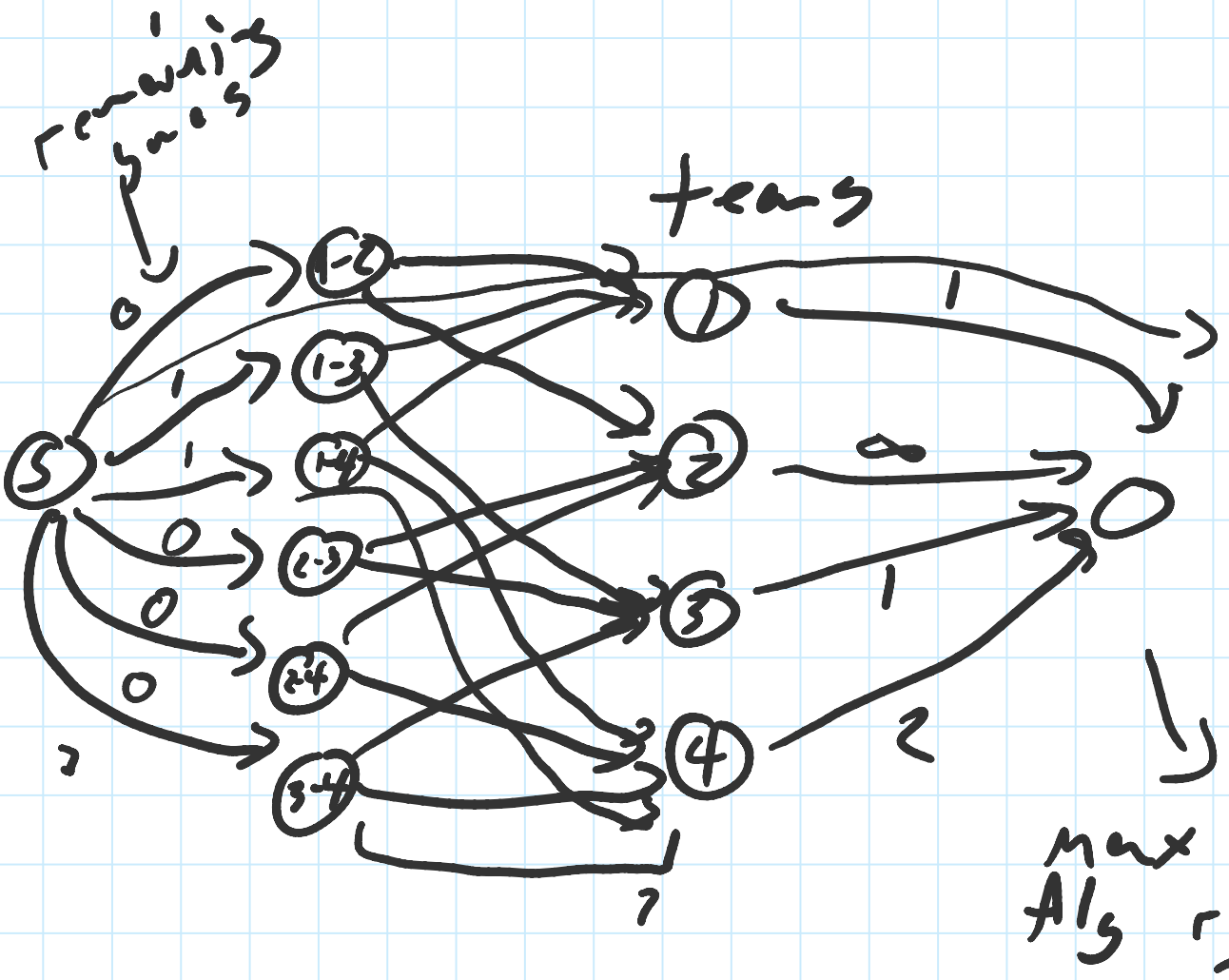




Example with team 4

assume 4 wins game 1-4

What flow do we need such that 4 is chosen is not eliminated



$T_1 = 1$

$T_2 = 1$ (+1) ← target + each wins remaining games

$T_3 = 1$

T_{4-n}

$T_4 = 0$

To check for elimination
Verify that the remaining
gains is not equal to
the flow.

Make a flow for each
team and check the
flow.

Project Assignments
task management

have a list of tasks
each task has a cost
and a reward!

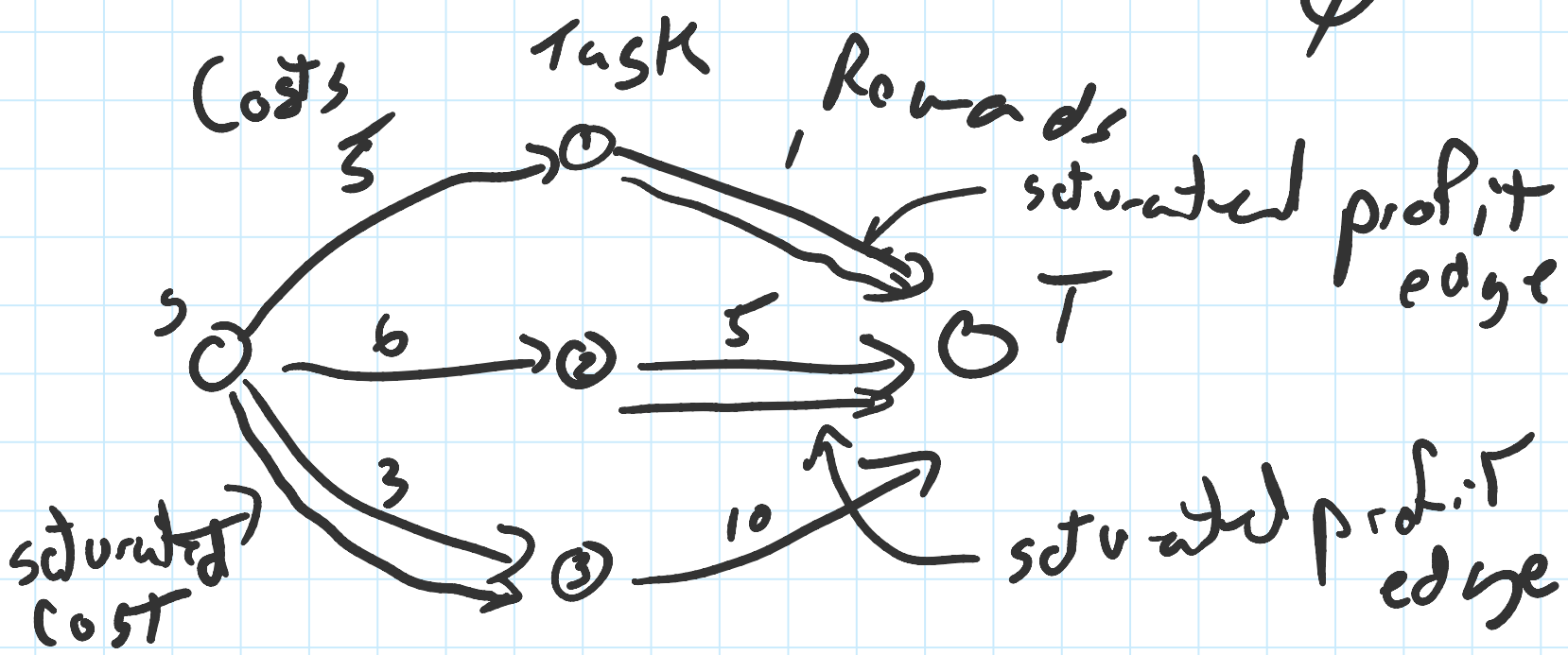
We can perform each
task once we want
to maximize the $\sum$ reward -
$\sum$ cost

of selected tasks!

take (Reward > Cost)

Some tasks must be performed before others

	Cost	reward	Require
T_1	5	1	$\emptyset$
T_2	6	5	$\emptyset$
T_3	3	10	$\emptyset$

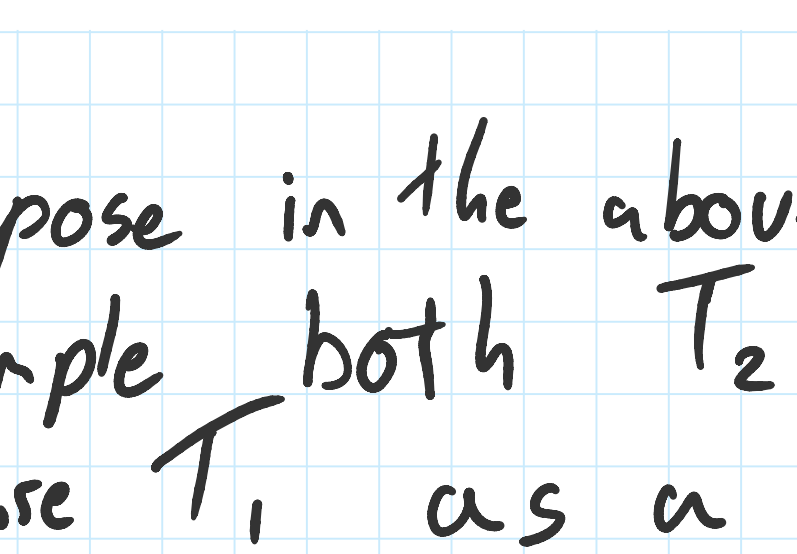


edge

We connect as incoming edges the task or costs associated the the current task

To see if task 3 on some task 1's cost

Suppose in the above example both T_2 and T_3 require T_1 as a pre req. We could say T_2 and T_3 have the additional cost of T_1

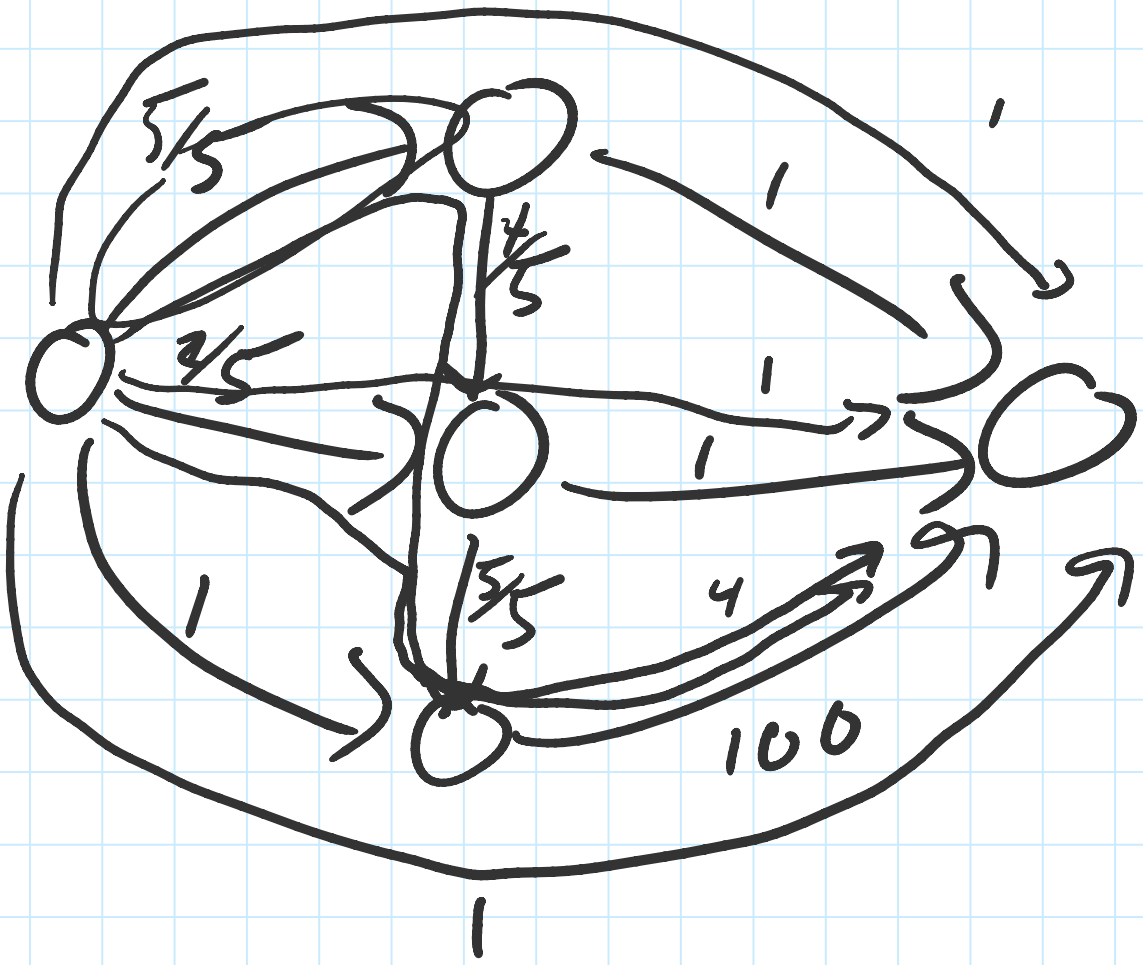


The flow becomes 14. Answer :> 16 - 14 = 2.

but we over pay for T_1. T_3 covers the cost and T_1 "pays" for it aswell.

Connect nodes directly to tasks that require the original node.

	Cost	Rewards	Req.
T_1	5	1	$\emptyset$
T_2	5	1	$\{T_1\}$
T_3	1	100	$\{T_2\}$



Finding the Answer

$162 - 11 = 91$ vs. $102 - 7 = 94$

Actual answer

total profit

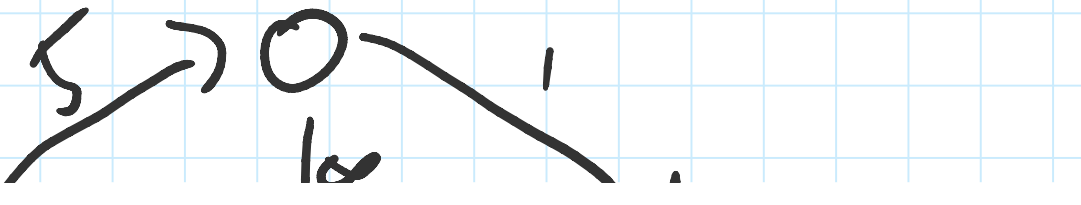
flow

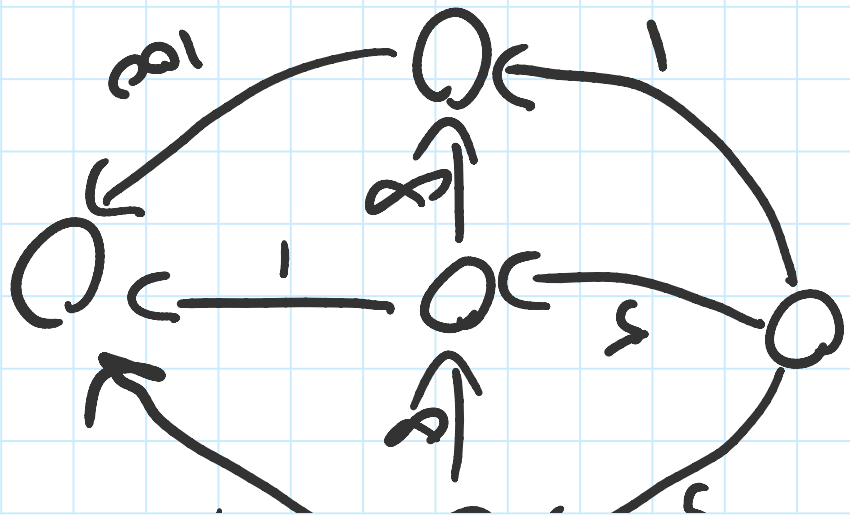
from flow and wrong

We under "paid"

Don't connect internal edge with just cost. Use sum of all pos eg. cost or ∞

Actual graph should be




∞
1
1
∞
1
1
∞