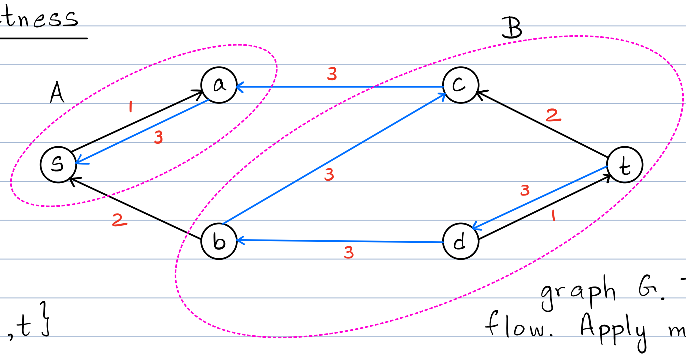
Correctness



$A = \{s, a\}$
$B = \{b, c, d, t\}$

Note, final residual graph identifies a cut (A, B) with capacity 5 in original graph G. This matches the found flow. Apply max-flow min-cut Thm.

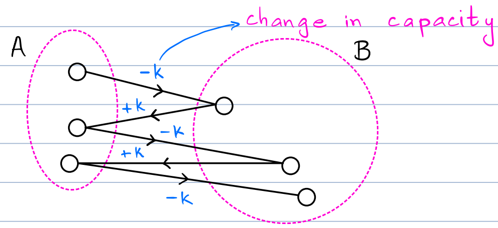
The final residual graph must have s & t disconnected, otherwise could increase flow from s to t, using existing path.

Let A be component containing s & B contains t.
Let c be capacity of the (A, B) cut in original graph.

Claim F-F finds flow of value c.

What happens to residual capacity of (A, B) cut after each iteration?

Assume in an iteration we find a flow k path. Even if the path zig-zags between A & B, the net result is a drop in residual capacity of the cut by exactly k.



So, drop in capacity equals increase in flow. At end of the algorithm, the residual capacity of (A, B) cut is zero, which implies found flow has value c.

Max-flow Min-Cut Thm $\Rightarrow$ max flow has value c.

Max-Flow Min-Cut Theorem

In any graph G, for any two vertices s & t, max flow from s to t equals capacity of min s-t-cut.

Edmonds-Karp Algorithm (Variation 1)

We fix a heurestic to determine which path we choose in step 1 of F-F.

Pick path with largest capacity. How does this influence complexity of algorithm?

Claim 1 In a graph with max s-t flow F, there must exist a path from s-t with capacity at least F/m.

Proof Why do we want to argue this? In F-F we argued in the worst case flow goes up by 1 each iteration, leading to F number of iterations.
How does EKv1 change this? We can give a bound on the largest capacity path.

Assume we remove all edges with capacity less than F/m.
We note that this cannot disconnect s & t, since otherwise we would have identified a cut of value less than F. (number of removed edges must be less than m.) Existing remaining s-t path must have capacity at least F/m.

Claim 2 EKv1 makes at most $O(m \lg F)$ iterations.

Proof At each iteration we add at least 1/m of the flow to the existing solution.

How many times can we remove 1/m of the remaining amount?

i.e., $F\left(1-\frac{1}{m}\right)^{\overset{\text{number of iterations}}{x}} < \underset{\text{less than 1 since we consider integer flow}}{1}$

Using $\left(1-\frac{1}{m}\right)^m \lesssim \frac{1}{e}$, we have $x = m \ln F$.

Edmonds-Karp Algorithm (Variation 2)

Pick shortest path in residual graph.

Claim EKv2 takes $O(mn)$ iterations.

Proof Let d be $\underbrace{\text{distance}}_{\text{number of edges in shortest path}}$ from s to t in current residual graph.

We show that:
a) d never decreases over iterations
b) Every m iterations, d has to increase by at least 1.
(This can happen n times.)

Lay out G according to levels from s via BFS, i.e., nodes at level i are distance i from s. Note t is at level d.

Each iteration will cause atleast one forward edge to be saturated & removed from residual graph & addition of backward edge.

This implies i) d cannot decrease.

ii) as long as d doesn't change, atleast one forward edge is removed.

We can remove forward edges atmost m times. So, within m iterations, either s-t become disconnected $\Rightarrow d = \infty$ OR new s-t path uses a non-forward edge $\Rightarrow d$ goes up by atleast one

Re-apply argument based on new distance.
Distance between s & t can increase at most $O(n)$ times.

So, total iterations are $O(m+n)$.

Linear Programming

Let us start off with a few examples of what a linear program looks like.
Note that a better title is linear optimization.

Example 1

There are 168 hours in a week.
We want to optimize our time between (S)tudy, (P)lay & (E)verything else based on our notion of happiness.

- To survive we need to spend atleast 56 hrs. on E.
- To maintain sanity $P + E \geqslant 70$
- To pass courses need $S \geqslant 60$
- Need more S if don't sleep enough or we P too much, so

The exact structure of the constraints is not up to us, receive them as input $\longleftarrow 2S + E - 3P \geqslant 150$

Can we do this? Yes, e.g., $S = 80$, $P = 20$, $E = 68$
Assume, notion of happiness is $\underbrace{2P+E}_{\text{objective function that we want to maximize.}}$. What solution maximizes this?

Defn Given n variables $x_1, \dots, x_n$
m linear inequalities / equalities
and a linear objective function

Goal Find x_i's such that constraints are satisfied & we maximize / minimize objective function

A feasibility problem has no objective function.
In our example, we have

Variables: S, P, E
Objective function: $2P + E$
Constraints:
$$S + P + E = 168$$
$$E \geqslant 56$$
$$S \geqslant 60$$
$$2S + E - 3P \geqslant 150$$
$$P + E \geqslant 70$$
$$\underbrace{P, S, E \geqslant 0}$$
positivity constraints applied given the interpretation of variables as representing time.

Example 2

Four manufacturing plants make cars. Each works differently in terms of labor required, materials & pollution produced per car.

	Labor	Material	Pollution
Plant 1	2	3	15
2	3	4	10
3	4	5	9
4	5	6	7

- Need to produce atleast 400 cars at plant 3
- We have 3300 hrs of labor & 4000 material available
- Allowed to produce 12,000 units of pollution
- Want to maximize number of cars produced

Variables: x_1, x_2, x_3, x_4 where x_i is number of cars built at plant i

Objective function: $\max \ x_1 + x_2 + x_3 + x_4$

Constraints: $\forall i,\ x_i \geqslant 0$

$x_3 \geqslant 400$

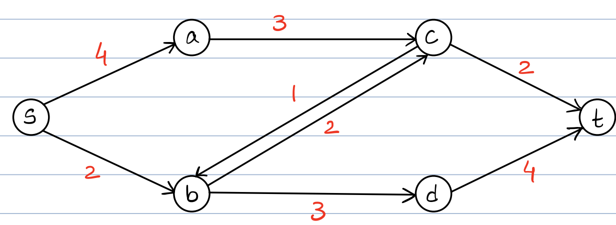
$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$

$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$

$15x_1 + 10x_2 + 9x_3 + 7x_4 \leq 12000$

No guarantee that solution is integral.

Max Flow as a Linear Program



Variables: f_{uv}, flow on each edge uv

Objective function: $\max \sum_{u \in V} f_{ut}$

Constraints: For all (u,v) $\quad 0 \leq f_{uv} \leq c_{uv}$ (capacity constraint)

For all $v \notin \{s,t\}$ $\quad \sum_u f_{uv} = \sum_u f_{vu}$ (flow conservation)

e.g., for the above graph we have, $\max \ (f_{ct} + f_{dt})$

s.t. $0 \leq f_{sa} \leq 4$, $0 \leq f_{ac} \leq 3$, etc.

$f_{sa} = f_{ac}$, $f_{sb} + f_{cb} = f_{bc} + f_{bd}$, etc.

Two Player Zero Sum Games

Defn A game is an interaction between players with shared interests.

Each player has a set of choices from which they choose one.
Combined choices determine payoffs.

Example

Payoff matrix between a goalkeeper (column) & shooter (row).

		Player 2 Goalie	
		Left	Right
Player 1 — Shooter	Left	$(-1,+1)$	$(+1,-1)$
	Right	$(+1,-1)$	$(-1,+1)$

$\longrightarrow$ (r, c) — r: row player pay off, c: column player pay off

The game is zero-sum if $r + c = 0$. Assume the players are rational & want to maximize payoff. A general (mixed) player strategy is a probability distribution over the choices.

Row player expected payoff $V_R(p, q) = \sum_{i,j} \Pr(\text{row plays } i, \text{ col plays } j)\, R_{ij}$

$p \longrightarrow$ row player strategy

$q \longrightarrow$ col. player strategy

$$= \underbrace{\sum_{i,j} p_i\, q_j\, R_{ij} = -V_C(p,q)}_{\text{for zero-sum game}} \quad \text{Col. player payoff}$$

Row player wants to find p^* that maximizes expected payoff, for all choices of opponent's strategy q. So, strategy that maximizes minimum payoff is

$$\max_p \min_q V_R(p,q)$$ can think of it as a lower bound

Guaranteed payoff to row player, no matter what column player does.

Similarly, col. player wants to find q^* that maximizes expected payoff, for all choices of opponent's strategy p. So, strategy that maximizes minimum payoff is

$$\max_q \min_p V_C(p,q) = \max_q \min_p \left(-V_R(p,q)\right)$$ for zero-sum games

$$= -\min_q \max_p V_R(p,q)$$

$\Rightarrow$ upper bound on row player is $\min_q \max_p V_R(p,q)$. Col. player can guarantee row player does not get more regardless of actions.

Want to find p^*. Observe that we can assume column player has a deterministic strategy. Why? So,

$$lb = \max_p \min_j \sum_i p_i R_{ij}$$

$$ub = \min_q \max_i \sum_j q_j R_{ij}$$

Min Max Optimality

Given a finite two player zero sum game,

$$lb = \max_p \min_j \sum_i p_i R_{ij} = \min_q \max_i \sum_j q_j R_{ij} = ub$$ (Called value of the game)

LP Example

Zero-Sum game, where $R = \begin{pmatrix} 20 & -10 & 5 \\ 5 & 10 & -10 \\ -5 & 0 & 10 \end{pmatrix}$ ($n \times n$ matrix)

Variables are n probabilities $p_1, \ldots, p_n$, $p_i \geq 0$, $\sum_i p_i = 1$.

Want to maximize worst case (minimum over all columns our opponent can play).

Add a new variable v, representing min & put in constraints that our payoff has to be atleast v for every column choice & then objective is to maximize v.

$$\begin{aligned} &\max \ v \\ \text{s.t. } & p_i \geq 0, \ \sum_i p_i = 1 \\ & 20p_1 + 5p_2 - 5p_3 \geq v \\ & -10p_1 + 10p_2 \geq v \\ & 5p_1 - 10p_2 + 10p_3 \geq v \end{aligned}$$