

# Linear Programming

## LP in canonical form

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.: } & \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i=1, \dots, m \\ & x_j \geq 0, \quad j=1, \dots, n \end{aligned}$$

n variables

$x_1, \dots, x_n$

n+m constraints

## Matrix Notation

$$\begin{array}{l} \max c^T x \\ \text{s.t. } Ax \leq b \\ \quad x \geq 0 \end{array} \quad \begin{aligned} x &= \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} & c &= \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \\ b &= \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} & A &= (a_{ij}) \in \mathbb{R}^{m \times n} \end{aligned}$$

## More general constraints

$$\begin{aligned} \max \quad & c^T x \\ (Ax)_i \leq b_i \quad & \text{for } i \in I \subseteq [1:m] \\ (Ax)_i = b_i \quad & \text{for } i \notin I \\ x_j \geq 0 \quad & \text{for } j \in N \subseteq [n] \end{aligned}$$

Feasible Region: Polytop of  $x \in \mathbb{R}^n$  s.t.  $x$  satisfies all constraints

Vertex: Point  $x$  in the feasible region such that n of the constraints are tight, i.e., satisfied as equalities

Neighbor of a vertex  $x$ : vertex  $x'$  which differs in 1 of the n equality constraints

Value:  $c^T x$

Fact: The max of an LP is attained at a vertex

## Simplex Algorithm

- Start at vertex  $x^*$
- Find neighbor  $y^*$  with maximal value
- If  $\text{value}(y^*) > \text{value}(x^*)$  move to  $y^*$ .
- Repeat

## Running time

- $O(n^3 \cdot nm)$  per step
- Worst case exponentially many steps

Polyomial time alg: Ellipsoid, interior point

## LP-Duality

Goal: Derive best possible upper bound in terms of linear combinations of the constraints

### Primal LP

$$\begin{aligned} & \max c^T x \\ \text{s.t. } & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Upper Bd  $b^T y$   
 if  $A^T y \geq c$   
 $y \geq 0$

### Dual LP

$$\begin{aligned} & \min b^T y \\ \text{s.t. } & A^T y \geq c \\ & y \geq 0 \end{aligned}$$

## More General Duality

### Primal:

$$\begin{aligned} & \max c^T x \\ \text{s.t. } & (Ax)_i \leq b_i \quad i \in I \\ & (Ax)_i = b_i \quad i \notin I \\ & x_j \geq 0 \quad j \in P \end{aligned}$$

### Dual

$$\begin{aligned} & \min b^T y \\ \text{n.t. } & (A^T y)_j \geq c_j \quad j \in P \\ & (A^T y)_j = c_j \quad j \notin P \\ & y_i \geq 0 \quad i \in I \end{aligned}$$

## Thm [Weak Duality]

$$\text{Opt Primal} \leq \text{Opt Dual}$$

## Thm [Strong Duality]

IF primal LP is bounded

$$\text{Opt Primal} = \text{Opt Dual}$$

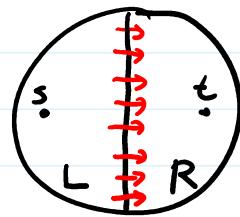
## Min Cut = Max Flow

- Input:
- a directed graph  $G = (V, E)$
  - capacities  $c_e \geq 0, e \in E$
  - a source  $s \in V$ , a sink  $t \in V$

Def  $s-t$  cut: a partition  $V = L \cup R$  s.t.  $s \in L, t \in R$

$$\text{capacity}(L, R) = \sum_{\substack{uv \in E \\ u \in L, v \in R}} c_{uv}$$

$$\text{Min Cut} = \min_{(L, R)} \text{capacity}(L, R)$$



## Maximum Flow Problem

Max size( $f$ ) s.t.  $f$  is a flow from  $s$  to  $t$

$$\text{Max } \sum_{v: s \neq v \in E} f_{sv}$$

$$\text{s.t. } 0 \leq f_e \leq c_e \text{ for all } e \in E$$

$$\sum_{u \neq v \in E} f_{uv} = \sum_{v \neq w \in E} f_{vw} \text{ for all } v \neq t, t$$

$$\text{Max Flow} = \max_{f \text{ is a flow}} \text{size}(f)$$

Thm: If the capacities are integers

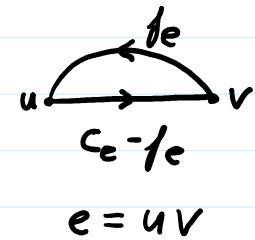
$$\text{MinCut} = \text{Max Flow} = \text{size } f$$

where  $f$  is the output of the Ford-Fulkerson algor.

### Residual Graph

Given a directed graph  $G = (V, E)$  and a flow  $f$ , the residual graph  $G_f$  is obtained by

- replacing the capacities  $c_e$  by  $c_e - f_e$
- adding backedges with capacities  $f_e$



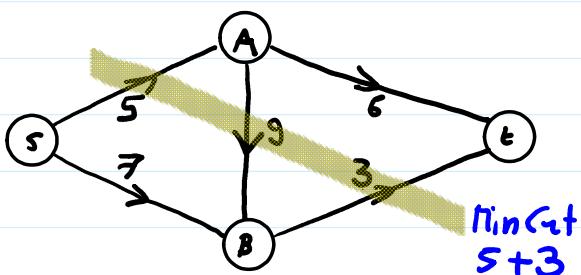
### Ford-Fulkerson Algorithm

- 1) Find not yet saturated path  $P$  from  $s$  to  $t$  in  $G_f$
- 2) Send additional flow  $\Delta f(P) = \min_{e \in P} c_e(G_f)$  along  $P$
- 3) Repeat until  $\Delta f(P) = 0$  for all paths from  $s$  to  $t$  in  $G_f$

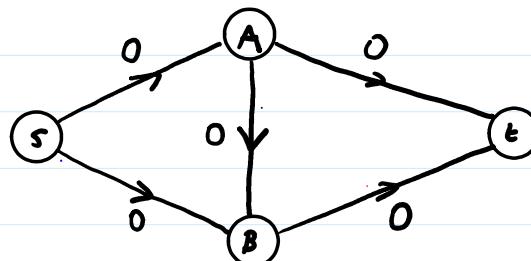
Rem: If all capacities are integers, this outputs an integer flow, and Run Time =  $O(\text{Max Flow} \cdot (m+n))$

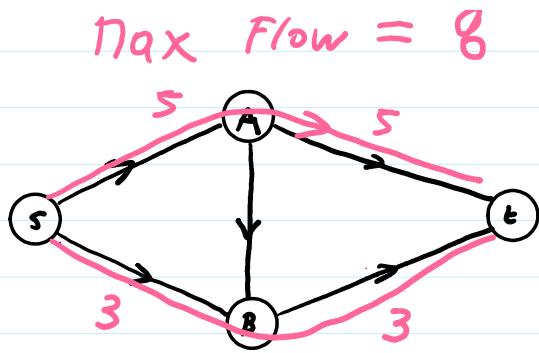
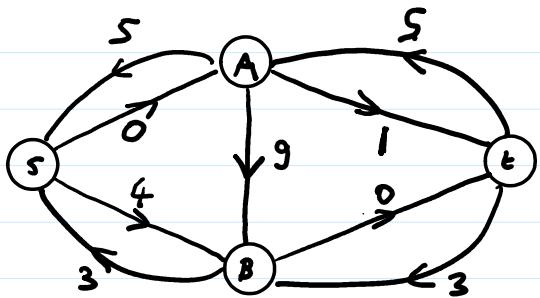
### Example:

#### Residual Graph

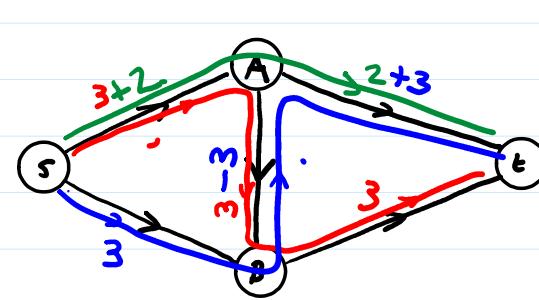
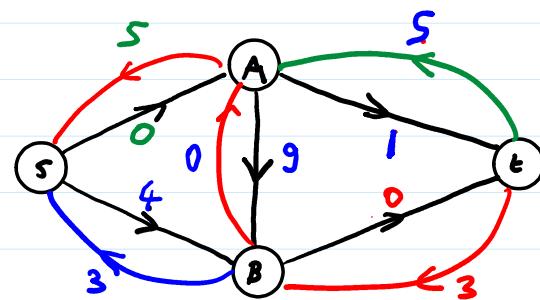
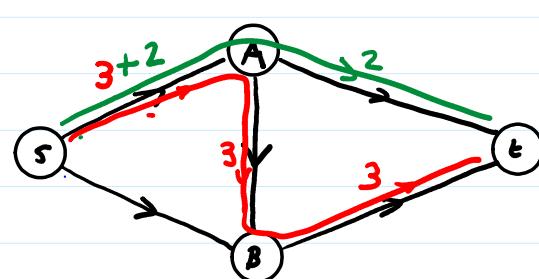
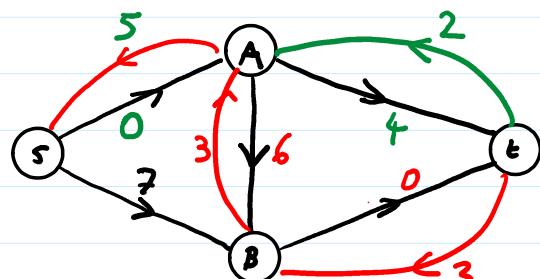
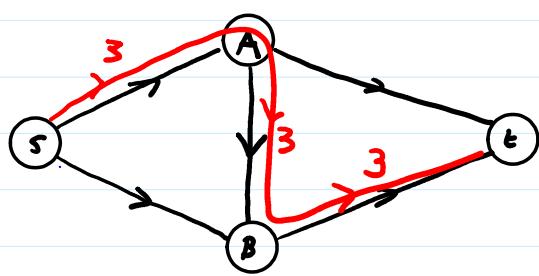
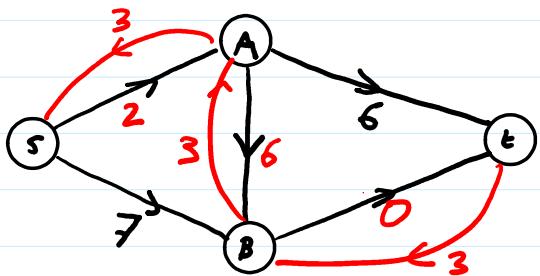


#### Flow





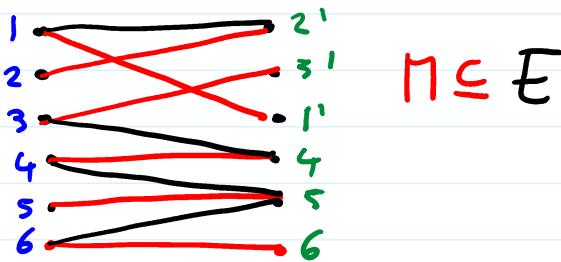
Intermediate steps



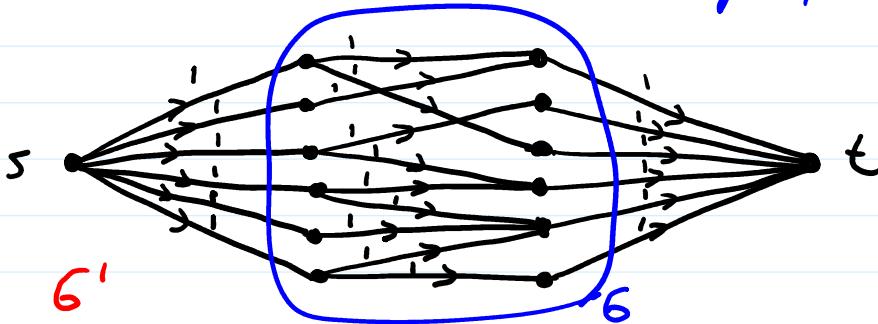
Bipartite Perfect Matching

Input: Bipartite graph  $G = (L, R, E)$   $|L| = |R| = n$

Output: A perfect matching  $M$  from  $U$  to  $V$



Solution via Max-Flow on new graph  $G'$



Thm:  $G$  has perfect matching  $\Leftrightarrow G'$  has integer flow with size  $f(G') = n$

$\Rightarrow$  Run Ford-Fulkerson on  $G'$  to find matching, if it exists

## Two Player - Zero Sum Games

Input: Payoff Matrix  $M$

Row Player: picks row  $r$   
Col. Player: picks col  $c$  } Payoff  $\begin{cases} M[r, c] \\ -M[r, c] \end{cases}$

	rock	paper	sciss.
rock	0	-1	1
paper	1	0	-1
sciss.	-1	1	0

Pure Strategy: a single row / column

Mixed Strategy: probability distribution over pure strategies

Row players average score = - col player average score

$$\text{Score}(p, q) = \sum_{r,c} p_r q_c \pi[r, c]$$

when row plays mixed strat.  $p$  and col plays mixed strat.  $q$

### 1) Row plays first

1. Row player announces mixed strategy  $p$

2. Col player responds w/ mixed strategy  $q$

Row player choose strategy, anticipating best response

$$\max_p \min_q \text{Score}(p, q) = \max_p \min_c \sum_r p_r M[r, c]$$

$\uparrow$  pure strategy

### 2) Col plays first

$$\min_q \max_p \text{Score}(p, q) = \min_q \max_r \sum_c q_c M[r, c]$$

$\uparrow$  pure strategy

### Min Max Theorem

$$\max_p \min_q \text{Score}(p, q) = \min_q \max_p \text{Score}(p, q)$$

### Example

Row player first  
Col.  $\rightarrow$  2<sup>nd</sup>

	1	2
1	3	-1
2	-2	1

Col. player first  
Row  $\rightarrow$  2<sup>nd</sup>

$$\max_p \min_q \text{Score}(p, q)$$

solution of LP

$$\min_q \max_p \text{Score}(p, q)$$

solution of dual LP

$$\begin{aligned} \max \quad & x \\ \text{s.t.} \quad & x \leq 3p_1 - 2p_2 \\ & x \leq -p_1 + p_2 \\ & p_1 + p_2 = 1 \\ & p_1, p_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \min \quad & y \\ \text{n.th.} \quad & y \geq 3q_1 - q_2 \\ & y \geq -2q_1 + q_2 \\ & q_1 + q_2 = 1 \\ & q_1, q_2 \geq 0 \end{aligned}$$

Strong LP Duality:  $\max \min = \min \max$

## DYNAMIC PROGRAMMING

### General Recipe

Problem P



- Try to reduce it to a slightly smaller subproblem

$$\text{Irregular shape} = f(\text{Irregular shape with hole})$$

- Introduce sequence of smaller and smaller problems
- Calculate bottom up

Formally:

- 1) Define subproblems
- 2) Derive recursion relation
- 3) Determine dependencies / calculation order

## 7) Travelling Salesman Problem (TSP) $O(n^2 2^n)$

Given:  $n$  cities, distances  $d_{ij}$   $i \neq j$

Goal: Find path of minimal length,

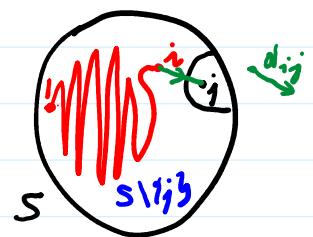
starting at 1, ending at 1, visiting every city once

Subproblem: For every  $S \subseteq \{1, \dots, n\}$  containing 1,  $\forall j \in S, j \neq 1$

$C(S, j) =$  Length of shortest path from 1 to  $j \in S$   
visiting every  $i \in S$  once

Recurrence:

$$C(S, j) = \min_{\substack{i \in S \\ i \neq 1, j}} C(S \setminus \{j\}, i) + d_{ij}$$



Calculation Orders: All sets  $S$  of size 2,  
all sets  $S$  of size 3, ...

Initialization:

$$C(\emptyset, 1) = \infty \quad \text{if } |\emptyset| > 1 \quad \text{and} \quad C(\{1\}, 1) = 0$$

Algorithm:

$$C(\{1\}, 1) = 0$$

For  $k = 2, \dots, n$

For all  $S \subseteq \{1, \dots, n\}$ ,  $1 \in S$ ,  $|S| = k$

$$C(S, 1) = \infty$$

For all  $j \in S \setminus \{1\}$

$$C(S, j) = \min_{i \in S, i \neq j} C(S \setminus \{j\}, i) + d_{ij}$$

$$\text{Output } \min_{j \in \{1, \dots, n\}} C(\{1, \dots, n\} \setminus \{j\}, j) + d_{j1}$$

1) Longest path in a DAG  $O(n+m)$

Subproblem  $L(v) = \text{length of longest path ending in } v$

Recursion

$$L(v) = \begin{cases} \max_{u: u \in E} (L(u) + 1) & \text{if exist } u \text{ with } u \in E \\ 0 & \text{otherwise} \end{cases}$$

Calculation order topological sort

2) Longest Increasing Subsequence  $O(n^2)$

Subproblem:  $L(i) = \text{length of longest increasing subsequence ending in } a_i$

3) Edit Distance  $O(nm)$

Subproblem:  $E(x[1:i], y[1:j]) = \text{edit distance between prefixes}$

4) Knapsack (capacity  $W$ , items with weights  $w_1, \dots, w_n$  & values  $v_1, \dots, v_n$ )

4a) Knapsack with Replacement  $O(n|W|)$

Subproblem:

$K(C) = \max \text{ total value with capacity } C$   $C=0, 1, \dots, W$

4b) Knapsack w/o replacement  $O(n|W|)$

Subproblem

$K(C, k) = \text{Optimum with total weight } \leq C$   
 while only using items in  $\{1, \dots, k\}$        $k=0, 1, 2, \dots$

### 5) Single Source shortest Path $O(nm)$

**Subproblems**

$\text{dist}(v, k) = \text{length of shortest path from } s \rightarrow v \text{ using } \leq k \text{ edges}$

Gives modified version of Bellman Ford

### 6) All Pairs shortest path (Floyd Warshall Algorithm) $O(n^3)$

**Subproblem**

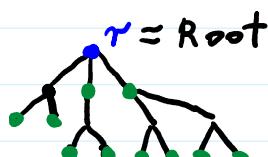
$$V = \{1, 2, \dots, n\}$$

$\text{dist}(i, j; k)$  uses only vertices in  $\{1, 2, \dots, k\}$   
 as intermediate vertices

### 8) Maximal Independent Set for Trees $O(n)$

Def: Given  $G = (V, E)$ , an independent set is a set  $I \subseteq V$  s.t. no pair of vertices  $\{x, y\} \subseteq I$  is an edge in  $E$

Define Subproblem:

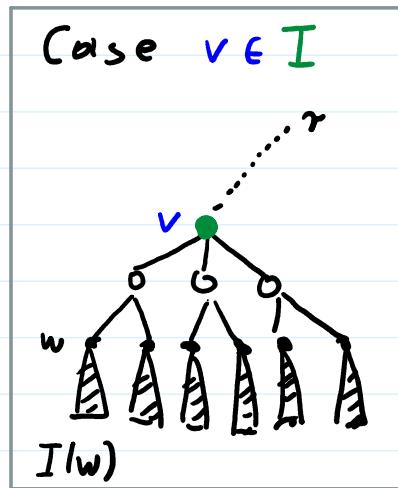
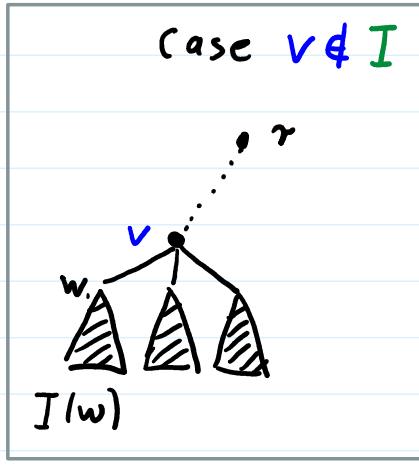


Subproblem:

$T_v = \text{subtree under } v$

$I(v) = \text{size of largest independent set } I \text{ in } T_v$

## Recursion



Base Case:  $v$  is a leaf

$$I(v) = 1$$

Calculation Order: Leaves to root

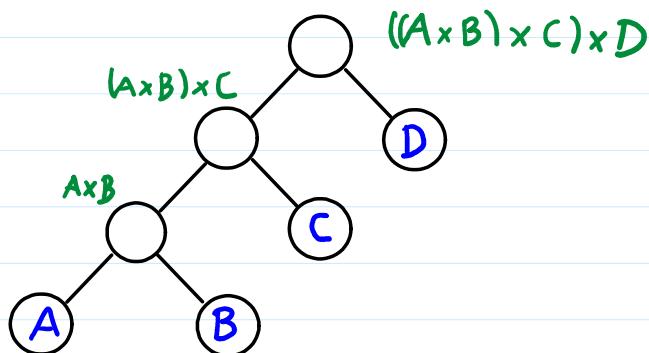
## g) Chain Matrix Multiplication

Input: Matrices  $A_1, \dots, A_n$ , where  $A_i$  is  $m_i \times m_i$ , ...

$A_n$  is  $m_{n_1} \times m_{n_2}$

Question: Best order to calculate

Binary tree representation:



## Subproblem

$$A_i \times A_{i+1} \times \cdots \times A_j$$

$C(i, j)$  run time for optimal way to put parentheses

## Recursion:

$$(A_1 \times \cdots \times A_h) \times (A_{h+1} \times \cdots \times A_j) .$$

Cost:

$$C(i, h) + C(h+1, j) + m_{i-1} m_h m_j$$

Best way

$$C(i, j) = \min_{i \leq k \leq j} \{C(i, h) + C(h+1) + m_{i-1} m_h m_j\}$$

## Order of Calculations

$$s = |i-j| = 0, 1, 2, \dots$$

## Algorithm

For  $i=1, \dots, n$      $C(i, i) = 0$

For  $s=1, \dots, n-1$

    For  $i=1, \dots, n-s$

        set  $j = i+s$

$$C(i, j) = \min_{i \leq k \leq j} (C(i, h) + C(h+1) + m_{i-1} m_h m_j)$$

Return  $C(1, n)$