Complexity Theory

Seen lots of algorithms in class, lots of solvable problems.

Q: what can't we solve?

Unfortunate answer: almost everything.

Small variation of problem $\Rightarrow$ probably intractable

**Flows:**

- Max $s$-$t$ flow
- Multicommodity flow:
  - $K$ commodities, each w/ own $S_i \rightarrow E_i$
  - $X$ can't even for $K=2$

**Min cut:**

- Smallest $(S, V \setminus S)$ cut
- Max cut:
  - Largest $(S, V \setminus S)$ cut

**Min Spanning tree:**

- Smallest tree connecting all nodes

**Min Steiner tree:**

- Only need to connect subset $S \subseteq V$

**Interval packing:**

- Most disjoint intervals
- On multiple machines
  - Each interval works on $M_k$ machines
  - $X$

Shortest path $\checkmark$

Longest path $X$
Complexity theory studies hard problems

- how hard are they?
- classify into "complexity classes"

Zooms out, for big picture view.

Now: \( n = \) size of input in bits

Only consider decision problems
(binary answer: YES or NO)

Examples:
"Shortest Path" has input \((G, s, t, K)\)
Q: does there exist a path in \(G\)
    of length \( \leq K \)

What is \( n^2 \)?

\[
  n = \begin{align*}
  & E \cdot \lg v \quad \text{\# represent graph [length \leq 2^w]} \\
  & + E \cdot w \quad \text{edge costs w bit words} \\
  & + 2 \lg v \quad s, t \\
  & + (w + \lg v) \quad \text{representation of } K
  
  & = \Theta(E(w + \lg v)) \quad [K \leq V \cdot 2^w]
  
\]

Dijkstra = \( O(E + V \lg v) \) word operations
= \( O(Ew + wV \lg v) \) bit operations
\leq O(n \log n)
"min cut" $\rightarrow$ "\exists$\cup$ of size $\leq k$"
"min spanning tree" $\rightarrow$ "\exists spanning tree of size $\leq k$"
"max cut" $\rightarrow$ "\exists$\cup$ of size $\geq k$"

If you can solve decision problem, can solve optimization ("what is shortest path length") by binary search on $K$, $\lceil O(\lg n) \rceil$ iterations]

$\Rightarrow$ can find solution / "what is the path?"
[remove edges sequentially & see if length changes]
$O(E) \leq O(n)$ times slower

Big Picture: such differences don’t matter
& decision problems are simply to study

$P$: problems solvable in polynomial time
[= now time]

Shortest path $\in P$,

Longest path (\exists path of length $\geq K$)
seems hard to solve—
but can check a solution w/ proof
(if a long path exists, can check it)
(if doesn’t exist, can’t prove it)
NP: Non-deterministic polynomial time

There are problems for which there exist polynomial time verifiers.

"Completeness"

For all YES instances \( x \),

\[ \exists y \in \{0,1\}^\text{poly}(n) \quad \text{s.t.} \quad A(x, y) = \text{YES} \]

"Soundness"

For all NO instances \( x \),

\[ \forall y \in \{0,1\}^\text{poly}(n) \quad A(x, y) = \text{YES} \]

Anything true (can be proven; nothing false can be)

\[ \text{NP} \subseteq \text{P}^\text{co-NP} \]

\[ \text{P} \leq \text{NP}: \quad A(x, y) = A'(x) \]

Many harder problems:

- How many longest paths are there?
- Does white win this chess position?
- Does this program halt?

Million dollar question: is \( \text{P} = \text{NP} \)?

Finding proof harder than checking it?

\( \{ \text{Max-CUT, longest path, multi-commodity flow, steepest tree, integer LP, ...} \} \subseteq \text{NP} \)

Unknown if they lie in \( \text{P} \).

Best unconditional lower bound for any problem: \( 3.011 n \)
But we do know something:
all these problems equally hard: all in P

How to show problem A easier than B
if we don't know how hard either are is:

Reduction: A "reduces to" B
(= "Reduction from A to B")
if: Can solve A using B

Cook Reduction:
given oracle to B
Can solve A using $n^{o(1)}$ time
+ $n^{o(1)}$ calls to B.

Karp Reduction:
$A(x) = B(F(x))$
for polynomial time function $F$.

$A \leq_p B \iff$ Karp reduction $A \rightarrow_B$

We've seen linear time reductions:

Bipartite matching reduces to flow
Fastest solvable game reduces to Shortest path
In algorithms: (unknown, new problem) reduces to (known easy problem) 
\[ \Rightarrow \text{ new problem no harder than old } \]

In complexity: (known, "hard" problem) reduces to (unknown, new one) 
\[ \Rightarrow \text{ new problem no easier than old } \]

**Cook's Theorem** 3 problem, called SAT such that every problem $P \in NP$