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 $S_i \rightarrow T_i$
  - $\times$ can't even for $K=2$

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

Min spanning tree:
- Smallest tree connecting all nodes $\checkmark$

Min Steiner tree:
- Only need to connect subset $S \subseteq V$ $\times$

Interval packing:
- Most disjoint intervals $\checkmark$

on multiple machines
- Each interval works on $\mathcal{M} \subseteq \mathcal{K}$ $\times$

Shortest path $\checkmark$

Longest path $\times$
Complexity theory studies hard problems

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

Zooms out, for big picture view.

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

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

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

what is \(n^2\)?

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

Dijkstra = \(O(E + V \log V)\) \text{ word operations}
= \(O(Ew + wV \log v)\) \text{ bit operations}
\leq O(n \log n)
"min cut" \rightarrow "\exists \text{ cut of size } \leq k"

"min spanning tree" \rightarrow "\exists \text{ spanning tree of size } \leq k"

"max cut" \rightarrow "\exists \text{ cut of size } \geq k"

If you can solve decision problem, can solve optimization ("what is shortest path length")
by binary search on k.
[\mathcal{O}(\log n) \text{ iterations}]

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

Big picture: such differences don’t matter & decision problems are simpler to study.

P: problems solvable in polynomial time
[= \text{ now time}]

Shortest path & P.

Longest path (\exists \text{ path of length } \geq k \notin P)
seems hard to solve—
but can check a solution with proof
(if a long path exists, can check it)
(if doesn’t exist, can’t prove it)
NP: Non-deterministic polynomial time
problems for which ∃ poly time verifier A.

"Completeness"
∀ YES instances X
(\text{input } x \text{ s.t. answer } = \text{YES})
∃ proof y ∈ \{0, 1\}^{poly(n)}
s.t. A(x, y) = \text{YES}

"Soundness"
∀ NO instances X
∃ proof y ∈ \{0, 1\}^{poly(n)}
s.t. A(x, y) = \text{YES}

[anything true can be proven; nothing false can be]

\[ \text{NP} \] still means easy (ish)
\[ P \subseteq \text{NP}: A(x, y) = A'(x). \]

\exists much harder problems:
how many longest paths are there?
Does white win this chess position?
Does this program halt?

Million dollar Q: is \( P = \text{NP} \)?
finding proof harder than checking it?

\{\text{Max-Cut, longest path, multi commodity flow, steiner tree, integer LP, ...} \} \subseteq \text{NP}

Unknown if they lie in \( P \).
Best unconditional lower bound for any problem:
\[ 3.011n \]
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 one is?

Reductions \( A \) ”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 \( f \).

\[ A \prec_p B \iff \text{Karp reduction } A \rightarrow B \]

We’ve seen linear time reductions:

Bipartite matching reduces to flow
Fastest sol into 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 problem)
\[ \Rightarrow \text{new problem no easier than old} \]

**Cook's Theorem**

3 problem, called SAT

such that \( \text{SAT} \in \text{NP} \) and:

- If \( \text{SAT} \in \text{P} \),
- then \( \text{P} = \text{NP} \)

NP-hard

\( \text{NP} \supseteq \text{P} \)

equiv.: Every \( q \in \text{NP} \) can be solved in a polynomial reduction to SAT

\[ \text{Circuit SAT} \quad \text{Boolean circuit satisfiability} \]

SAT

\[ (a \lor b) \lor (c \lor d) \lor (e \lor (f \lor g)) \]

3-SAT

\[ (a \lor b \lor c) \lor (d \lor e \lor f) \lor (g \lor h \lor i) \]

rest of Karp's 21 problems