Complexity Theory

- Study *inherent* complexity of computing a function.
- I.e., What is the minimum time that the best algorithm that solves a given computational problem takes?
- Notice order of quantification:
  - Problem is a mathematical function $f$.
  - Say, algorithm $A$ computes function $f$ correctly on all inputs.
  - Let $A$ have worst-case running time $T_A(n)$ on inputs of length $n$.
  - Over all such algorithms, take the one with the "smallest" asymptotic running time, and let this one run in time $T(n)$.
  - Complexity theory tries to find out this $T(n)$ for every (or at least, every interesting) function $f$.

Complexity theory (contd.)

- Two main directions:
- Prove upper bounds on $T(n)$. What does this mean? Give an algorithm for $f$ and analyze its running time. Did this all semester long.
- Prove lower bounds on $T(n)$. How? (Not good enough to say some $A$ takes at least $T_L(n)$ time!)
- Need to do this by deep mathematics!

Some lower bound techniques

- Restrict model of computation and analyze information complexity. E.g., Sorting takes $\Omega(n \log n)$ time in the comparison based model.
- More generally: Always need to formalize model of computation to prove lower bounds.
- Another approach: Diagonalization = Very subtle contradiction. Results of this type include:
  - There is no finite time algorithm that can determine if a given computer program will eventually terminate on a given input.
  - There is no $O(n)$ time algorithm that can determine if a given computer program will terminate in $O(n^2)$ time on an input of length $n$.
- What else? Not much really! Lots of interesting problems that we are unable to give tight bounds!
Some example problems

Satisfiability problems (logic):

• 2SAT:
  Input: $n$ Boolean variables $x_1, \ldots, x_n$ with $m$ clauses $x_1 \lor x_2$, $x_1 \lor \neg x_2$, $\neg x_1 \lor x_3$...
  Goal: Find "true/false" assignments of variables such that every clause is satisfied (i.e., in every pair above has one term set to "true").

• 3SAT: Same as above, except clauses may contain 3 terms.

State of affairs: 2SAT has an $O(n + m)$ algorithm, while 3SAT is not known to have a $O(n^c)$ algorithm for any $c$.

Example problems (contd.)

Example problems (contd.)

Graph coloring problems (logic):

• 2Col:
  Input: Undirected graph $G$. Goal: Color vertices "Red/Blue" such that adjacent vertices are colored differently (if this is possible).

• 3Col: Same as above, except can use 3 colors.

State of affairs: 2Col has an $O(V + E)$ algorithm, while 3Col is not known to have a $O(E^c)$ algorithm for any $c$.

Example problems (contd.)

Tours of graphs:

• Euler Tour:
  Input: Undirected graph $G$. Goal: Find a closed walk of the graph which traverses every edge exactly once (if possible).

• Hamiltonian Tour:
  Same as above, except must visit every vertex exactly one.

State of affairs: Euler tour an $O(E^3)$ algorithm, while Hamiltonian Tour is not known to have a $O(E^c)$ algorithm for any $c$.

Example problems (contd.)

• Travelling Salesperson Problem (TSP):
  Input: Undirected graph with lengths on edges.
  Output: Smallest tour that visits every vertex at least once.

• Clique:
  Input: Undirected graph.
  Output: Largest subset $C \subseteq V$ such that every pair of vertices in $C$ are adjacent to each other.

Status: Neither problem known to have polytime solution.

Defn: $P$ is the class of problems solvable in polynomial time.
Summary

- Bunch of interesting problems.
- Would love to show they are in P.
- Would be satisfied if we can prove they are not in P.
- But can do neither! So what to do?
- Show they are equivalent!
- Polynomial time reductions (or transformations)!

Polynomial Reductions

A reduction from problem A to problem B is a transformation that allows an algorithm that solves problem B to be used (as a subroutine) to solve problem A.

Def: A (polytime) reduction from A to B is a pair of polynomial time computable functions f and g such that f maps inputs to A into inputs of B and g maps solutions of B into solutions of A with the following requirement:

If y is a solution to f(x), then g(y) is a solution to x.

Terminology: A reduces to B if such a reduction exists.

Example Hamiltonian Tour reduces to TSP.

f: Takes input graph G for Ham. Tour and converts into input for TSP. Use same graph with lengths = 1.

g: If TSP has solution of length V, then the tour is the Hamiltonian Tour (so, in this case g(y) = y) but if the TSP solution has of length greater than V, then G has no Ham. Tour (so, g(y) = NULL).

Equivalence of problems

Defn: Problem A is (polynomial-time) equivalent to Problem B (A ≡p B), if

\[ A \leq_p B \quad (A \text{ reduces to } B) \]

and \[ B \leq_p A \]

Main outcome of “NP-completeness theory”

Thm:

3SAT \equiv_p 3COL \equiv_p Ham. Tour \equiv_p TSP \equiv_p Clique.
**Easy and Hard Problems**

**Defn:** $P$ (for Poly-time solvable) is the class of polynomial time solvable problems (i.e., has an $O(n^c)$ algorithm for some constant $c$.)

(Informally, $P$ is the class of "easy" problems.)

NP will be defined as to contain many hard problems (including all those on the previous page).

NP stands for "Non-deterministic polynomial time". (and NOT "Not Polytime").

**NP**

Class of Boolean computational problems, i.e., problems in which the output is one of TRUE/FALSE.

**Defn:** A Verifier $V$ is an algorithm that takes two inputs $x$ (the real input) and $y$ (a purported proof) and runs in polynomial time and outputs either TRUE/FALSE.

**Defn:** A Boolean computational problem $A$ is in NP if there exists a verifier $V$ (and constant $c$) such that the following hold:

- $A(x) = TRUE \Rightarrow$ there is some string $y$ of length $O(|x|^c)$ such that $V(x,y) = TRUE$.
- $A(x) = FALSE \Rightarrow$ for every string $y$ $V(x,y) = FALSE$.

(In other words, for TRUE $x$'s there is a proof $y$ that convinces the verifier. For FALSE $x$'s no proof convinces the verifier.)

**NP Examples**

The following problems are all in NP:

- 3SAT: Is the input 3SAT problem satisfiable?
- 3COL: Is the input graph 3-colorable?
- Ham. Tour: Does the input graph have a Ham. Tour?
- TSP: Does the input graph have a TSP tour of length $\leq L$?
- Clique: Does the input graph have a Clique of size $\geq k$?

(Note the Boolean reformulations of TSP and Clique!)

In the next lecture we'll define NP-completeness and describe the methods used to show the equivalence of all the above problems.