This Week

• Beyond Theory of Computation:
  – Computational Complexity
  – Reading: Sipser Chapter 7
  – Not covered in final exam

• Revisit familiar topics
  – Diagonalization
  – Reductions

• NP-completeness
Computability vs Complexity

• Theory of computation
  – What computational problems can be solved algorithmically?  
  – Are there undecidable problems? Yes!

• Computational Complexity
  – What problems admit efficient algorithms?  
  – Are there decidable problems with no efficient solution? Yes!
Last Time

• We used diagonalization to show that there are decidable problems that require at least \( \exp(n) \) time to solve, where \( n \) is the size of the input.

• Remarks
  - Same proof shows that are decidable problems requiring \( f(n) \) time for arbitrary large \( f(n) \).
  - Similar construction shows that are problems requiring \( f(n) \) memory to solve
Today: Reductions

- Computability theory:
  - Reducing A to B (A<B): “if B is decidable then A is decidable”
  - Method to compare hardness of two problems: A is not harder than B

- Complexity Theory:
  - Efficient Reductions: $A \leq_p B$
    “if B can be solved efficiently then A can be solved efficiently”
  - Contrapositive:
    “if A has no efficient solution, then B has no efficient solution”
Theory of Efficient Computation

- A problem can be solved in polynomial time if it admits an algorithm running in time $T(n) = O(n^c)$ for some constant $c$.
- $P$: the class of all decision problems that admit a polynomial time solution.
Polynomial Time vs Efficiency

• Efficient algorithms may run in time $O(n)$
• Efficient algorithms may run in time $O(n^2)$
• Should an algorithm running in time $O(n^{100})$ be considered efficient?
• Why identifying efficient computation with the theoretical notion of polynomial time?
Why P is important

- Problems with efficient solutions are in P
- Natural problems in P typically admit efficient solutions (running in, say, at most $O(n^3)$)
- P is the smallest class
  - Containing linear time algorithms $T=O(n)$
  - Closed under program composition
Why $P$ is important

- Problems with efficient solutions are in $P$.
- Natural problems in $P$ typically admit efficient solutions (running in, say, at most $O(n^3)$).
- $P$ is the smallest class:
  - Containing linear time algorithms $T = O(n)$
  - Closed under program composition

If $M$ has running time $O(n)$ and $M'$ makes $O(n)$ calls to $M$, what's the total running time of $M'$?

A) $O(n) + O(n) = O(2n)$
B) $O(n)^{O(n)} = O(n^n)$
C) $O(n)*O(n) = O(n^2)$
D) I don’t know
Why P is important

- Problems with efficient solutions are in P
- Natural problems in P typically admit efficient solutions (running in, say, at most $O(n^3)$)
- P is the smallest class
  - Containing linear time algorithms $T=O(n)$
  - Closed under program composition

If M has running time $O(n^3)$ and M’ makes $O(n^4)$ calls to M, what’s the total running time of M’?

A) $O(n^4)$
B) $O(n^7)$
C) $O(n^{12})$
D) I don’t know
Why P is important

• Smallest class including efficient algorithms and closed under program composition

• Invariant under different computational models
  − Any k-tape TM M with running time $O(n)$ can be converted into an equivalent 1-tape TM $M'$ with running time $O(n^2)$
  − Extended Church-Turing Thesis: any reasonable model of computation is polynomially equivalent to the TM, i.e., one can efficiently convert between models with at most a polynomial slow down
Polynomial Time Reductions

- Assume $L(M) = B$
- Let $M'$ be a program using $M$ as a subroutine s.t.
  - $L(M') = A$
  - $M'$ makes polynomially many calls to $M$, and performs a polynomial amount of local computation
- If $M$ decides $B$ in polynomial time, then $M'$ decides $A$ in polynomial time
- We say that $A \prec_p B$
Non-determinism

• For any Non-deterministic TM (NTM) $M$, there is a deterministic TM $M'$ such that $L(M) = L(M')$
  - $M'(x)$ tries all possible computational paths of $M(x)$
  - $M'(x)$ accepts if an computational path is accepting

• What about running time?
  - Assume $M(x)$ runs in time $T$, and at each step, $M$ can choose (non-deterministically) between two different transitions
  - How many different computational paths are there?
Non-determinism

• For any Non-deterministic TM (NTM) M, there is a deterministic TM M’ such that L(M)=L(M’)
  – M’(x) tries all possible computational paths of M(x)
  – M’(x) accepts if an computational path is accepting

• What about running time?
  – Assume M(x) runs in time T, and at each step, M can choose (non-deterministically) between two different transitions
  – How many different computational paths are there?

A) $2^T$
B) $T^2$
C) $2^T$
D) I don’t know
Simulating non-determinism

- The natural way to simulate NTM Time(T) computation by a deterministic TM takes Time(exp(T))

- Is there a more efficient way to turn NTM into TM?

- Perhaps no: no method has been discovered despite many efforts

- Still, no proof that NTM cannot be efficiently turned into deterministic TM
P vs NP

- P: class of problems solvable in polynomial time by a (deterministic) TM
- NP: class of problems solvable in polynomial time by a NTM
- Efficient simulating NTM by TM $\leftrightarrow$ Showing $P=NP$
Why is NP important?

- We don’t know how to build NTM

- Assume $P \neq NP$:
  - Extended Church-Turing Thesis $\rightarrow$ NTM is not a reasonable model

- Still, NP captures an interesting class of problems:
  - Problems whose solution, once found, can be efficiently checked

- Given a NTM $M$ and input $x$
  - We don’t know how to efficiently find an accepting computation of $M(x)$
  - Given $C[0], C[1], \ldots$, we can efficiently check $C$ is an accepting computation
NP completeness

• Cook-Levin Theorem: there is a problem B in NP such that for any problem A in NP it is the case that $A \leq_{p} B$

• Implication:
  – If B is in P, then A is in P
  – P=NP

• Any such B is called “NP-complete”
NP-complete problems

• Many problems of practical interest are NP-complete:
  – SAT: Determine if a boolean formula is satisfiable
  – CLIQUE: Find the largest clique in a graph
  – HAM: is there a tour that visits all nodes in a graph exactly once?
  – Many more computational problems from computational biology, optimization, economics, mathematics, ...

• If any of these problems is solvable in polynomial time, then they all are!
Next Time

• Friday’s class: Review

• Final exams:
  - Section A: Monday December 5, 8-11am, CENTR 109
  - Section C: Wednesday December 7, 8-11am, CENTR 109
  - Bring Photo ID, pens
  - Note card allowed
  - New seating map
  - No review session outside class; office hours instead