CSE 417: Algorithms and Computational Complexity

Winter 2001

Lecture 21

Instructor: Paul Beame

1

Computational Complexity

- We've been interested in solving problems by using efficient algorithms.
- Algorithm run-times we've liked:
  - $O(n)$, $O(n \log n)$, $O(n^2)$, $O(nm)$, $O(n^3)$, $O(n^{2.81})$
  - Bounded by a polynomial in # of bits in input
- Ones we haven't: $O(2^n)$, $O((1.618)^n)$

2

Polynomial versus exponential

- We'll say any algorithm whose run-time is
  - polynomial is good
  - bigger than polynomial is bad
- Note:
  - n^{100} is bigger than $(1.001)^n$ for most practical values of n but usually such run-times don't show up
  - There are algorithms that have run-times like $O(2^{n/22})$ and these may be useful for small input sizes.

3

Decision problems

- Computational complexity usually analyzed using decision problems
  - answer is just 1 or 0 (yes or no).
- Why?
  - much simpler to deal with
  - can just encode each bit of a problem that has a longer answer as a decision problem
  - certain definitions such as NP only make sense in terms of decision problems

4

Computational Complexity

- Classify problems according to the amount of computational resources used by the best algorithms that solve them
- Recall:
  - worst-case running time of an algorithm
    - **max** # steps algorithm takes on any input of size n
- Define:
  - $TIME(f(n))$ to be the set of all decision problems solved by algorithms having worst-case running time $O(f(n))$

5

Polynomial time

- Define P (polynomial-time) to be
  - the set of all decision problems solvable by algorithms whose worst-case running time is bounded by some polynomial in the input size.
- $P = \bigcup_{k\geq 0} TIME(n^k)$

6

Beyond P?

- There are many natural, practical problems for which we don't know any polynomial-time algorithms
- e.g. decisionTSP:
  - Given a weighted graph G and an integer k, does there exist a tour that visits all vertices in G having total weight at most k?

7

Solving TSP given a solution to decisionTSP

- Use binary search and several calls to decisionTSP to figure out what the exact total weight of the shortest tour is.
  - Upper and lower bounds to start are n times largest and smallest weights of edges, respectively
  - Call W the weight of the shortest tour.
- Now figure out which edges are in the tour
  - For each edge e in the graph in turn, remove e and see if there is a tour of weight at most W using decisionTSP
    - if not then e must be in the tour so put it back

8

More examples

- Independent-Set:
  - Given a graph $G=(V,E)$ and an integer k, is there a subset U of V with $|U| \geq k$ such that no two vertices in U are joined by an edge.
- Clique:
  - Given a graph $G=(V,E)$ and an integer k, is there a subset U of V with $|U| \geq k$ such that every pair of vertices in U is joined by an edge.

9

Satisfiability

- Boolean variables $x_1,\ldots,x_n$
  - taking values in $\{0,1\}$. 0=false, 1=true
- Literals
  - x_i or $\neg x_i$ for $i=1,\ldots,n$
- Clause
  - a logical OR of one or more literals
  - e.g. $(x_1 \vee \neg x_3 \vee x_7 \vee x_{12})$
- CNF formula
  - a logical AND of a bunch of clauses

10

Satisfiability

- CNF formula example
  - $(x_1 \vee \neg x_3 \vee x_7 \vee x_{12}) \wedge (x_2 \vee \neg x_4 \vee x_7 \vee x_5)$
- If there is some assignment of 0's and 1's to the variables that makes it true then we say the formula is satisfiable
  - the one above is, the following isn't
  - $x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \neg x_3$
- Satisfiability: Given a CNF formula F, is it satisfiable?

11

Common property of these hard problems

- There is a special piece of information, a short hint or proof, that allows you to efficiently verify (in polynomial-time) that the YES answer is correct. This hint might be very hard to find
- e.g.
  - DecisionTSP: the tour itself,
  - Independent-Set, Clique: the set U
  - Satisfiability: an assignment that makes F true.

12

The complexity class NP

- NP consists of all decision problems where one can verify the YES answers efficiently (in polynomial time) given a short (polynomial-size) hint.
- The only obvious algorithm for most of these problems is brute force:
  - try all possible hints and check each one to see if it works.
  - Exponential time.

13

Unlike undecidability

- Nobody knows if all these problems in NP can all be done in polynomial time, i.e. does P=NP?
  - one of the most important open questions in all of science.
  - huge practical implications
- Every problem in P is in NP
  - one doesn't even need a hint for problems in P so just ignore any hint you are given

14