CSE 373 Analysis of Algorithms
Fall 2016
Instructor: Prof. Sael Lee

LEC19: INTRODUCTION TO NP-COMPLETENESS

Lecture slide courtesy of Prof. Steven Skiena

REPORTING TO THE BOSS

- Suppose you fail to find a fast algorithm. What can you tell your boss?
  - "I guess I'm too dumb. . . " (dangerous confession)
  - "There is no fast algorithm!" (lower bound proof)
  - "I can't solve it, but no one else in the world can, either. . . " (NP-completeness reduction)

THE THEORY OF NP-COMPLETENESS

- Several times this semester we have encountered problems for which we couldn't find efficient algorithms, such as thetraveling salesman problem.
- We also couldn't prove exponential-time lower bounds for these problems.
- By the early 1970s, literally hundreds of problems were stuck in this limbo.
- The theory of NP-Completeness, developed by Stephen Cook and Richard Karp, provided the tools to show that all of these problems were really the same problem.

THE MAIN IDEA

Suppose I gave you the following algorithm to solve the *bandersnatch* problem:

- Bandersnatch(G)
  - Convert G to an instance of the Bo-billy problem Y .
  - Call the subroutine Bo-billy on Y to solve this instance.
  - Return the answer of Bo-billy(Y) as the answer to G.

Such a translation from instances of one type of problem to instances of another type such that answers are preserved is called a ***reduction***.

WHAT DOES THIS IMPLY?

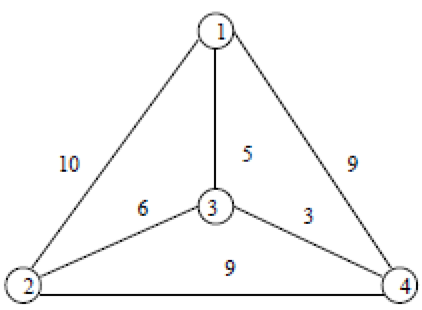
- Now suppose my reduction translates G to Y in *O(P(n)):*
  - 1. If my Bo-billy subroutine ran in *O(P'(n))* I can solve the Bandersnatch problem in *O(P(n) + P'(n'))*
  - 2. If I know that *Ω(P'(n))* is a lower-bound to compute Bandersnatch, then *Ω(P'(n) - P(n'))* must be a lowerbound to compute Bo-billy.
- The second argument is the idea we use to prove problems hard!

WHAT IS A PROBLEM?

- *A problem* is a general question, with parameters for the input and conditions on what is a satisfactory answer or solution.
  - Example: The Traveling Salesman
  - Problem: Given a weighted graph G, what tour $\{v_1, v_2, \ldots, v_n\}$ minimizes $\sum_{i=1}^{n-1} d[v_i, v_{i+1}] + d[v_n, v_1]$.

WHAT IS AN INSTANCE?

- An **instance** is a problem with the input parameters specified.
- TSP instance: $d[v_1; d_2] = 10$, $d[v_1; d_3] = 5$, $d[v_1; d_4] = 9$, $d[v_2; d_3] = 6$, $d[v_2; d_4] = 9$, $d[v_3; d_4] = 3$



- Solution: $\{v_1; v_2; v_3; v_4\}$ cost= 27

INPUT ENCODINGS

- Note that there are many possible ways to encode the input graph:
  - adjacency matrices, edge lists, etc.
- All reasonable encodings will be within polynomial size of each other.
- The fact that we can ignore minor differences in encoding is important.
- We are concerned with the difference between algorithms which are polynomial and exponential in the size of the input.

DECISION PROBLEMS

- A problem with answers restricted to **yes** and **no** is called a ***decision problem***.
- Most interesting optimization problems can be phrased as decision problems which capture the essence of the computation.
- For convenience, from now on we will talk *only* about decision problems.

THE TRAVELING SALESMAN DECISION PROBLEM

- Given a weighted graph G and integer k, does there exist a traveling salesman tour with (cost <= k)?
- Using binary search and the decision version of the problem we can find the optimal TSP solution.

REDUCTIONS

- Reducing (tranforming) one algorithm problem A to another problem B is an argument that if you can figure out how to solve B then you can solve A.
- We showed that many algorithm problems are reducible to sorting (e.g. element uniqueness, mode, etc.).
- A computer scientist and an engineer wanted some tea. . .

SATISFIABILITY

- Consider the following logic problem:
- Instance: A set V of variables and a set of clauses C over V .
- Question: Does there exist a satisfying truth assignment for C?
- Example 1: $V = v_1, v_2$ and $C = \{\{v_1, \overline{v_2}\}, \{\overline{v_1}, v_2\}\}$
- A clause is satisfied when at least one literal in it is TRUE. C is satisfied when $v_1 = v_2$ =TRUE.

NOT SATISFIABLE

- Example 2: $V = v_1, v_2$

$$C = \{\{v_1, v_2\}, \{v_1, \overline{v_2}\}, \{\overline{v_1}\}\}$$

- Although you try, and you try, and you try and you try, you can get no satisfaction.
- There is no satisfying assignment since v_1 must be FALSE (third clause), so v_2 must be FALSE (second clause), but then the first clause is not satisfiable!