Computational Complexity

Homework

- Homework 8
  - Help after lecture

Before we start

- Any questions?

The Turing Machine

- Motivating idea
  - Build a theoretical a "human computer"
  - Likened to a human with a paper and pencil that can solve problems in an algorithmic way
  - The theoretical machine provides a means to determine:
    - If an algorithm or procedure exists for a given problem
    - What that algorithm or procedure looks like
    - How long would it take to run this algorithm or procedure.

The Church-Turing Thesis (1936)

- Any algorithmic procedure that can be carried out by a human or group of humans can be carried out by some Turing Machine"
  - Equating algorithm with running on a TM
  - Turing Machine is still a valid computational model for most modern computers.

Decision Problem

- Let's formalize this a bit
  - A decision problem is a problem that has a yes/no answer
  - Example:
    - Is a given string x a palindrome (Is $x \in$ pal?)
    - Is a given context free language empty?

Decision Problem

- Running a decision problem on a TM.
  - The problem must first be encoded
  - Example:
    - Is a given string x a palindrome (Is $x \in$ pal?)
      - x is an instance of the probkem
    - Is a given context free language empty?
      - Instance of a problem is a CFG…must be encoded.

What makes a good algorithm?

- Suppose we know that a decision problem is decidable, how do we know that there is a good algorithm that solves the problem?
  - Consider running time on a TM
  - Actually, consider the complexity of a TM that can solve the problem.

Complexity

- Complexity refers to the rate at which the storage or time grows as a function of the input to the algorithm
  - T(n) = time complexity (amount of time an algorithm will take based on input)
  - S(n) = space complexity (amount of space an algorithm will take based on input)
- For the rest of this lecture, we'll only consider T(n) though the discussion can also be applied to S(n)

Units

- What does T(n) return?
  - A "basic operation" can be defined as a move a TM makes in accepting / rejecting a string
  - T(n) = number of TM moves

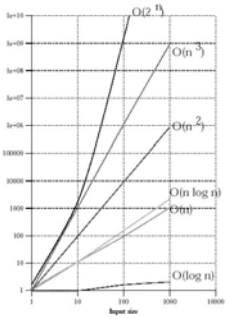
Asymptotic Analysis

- Based on the idea that as the input to an algorithm gets large
  - The complexity will become proportional to a known function.
  - Notations:
    - O (Big-O) – upper bounds on the complexity
    - Θ (Big-Theta) – tight bounds on the complexity

Asymptotic Analysis

- Big-O (order of)
  - $T(n) = O(f(n))$ if and only if there are constants c_0 and n_0 such that
    - $T(n) \leq c_0 f(n)$ for all $n \geq n_0$
  - What this really means
    - Eventually, when the size of the input gets large enough, the upper bounds on the runtime will be proportional to the function f(n).

Algorithm Efficiencies



Are you a good witch?

- So what should be the cutoff between a "good" algorithm and a "bad" algorithm?
  - In the 1960s, Jack Edmonds proposed:
    - A "good" algorithm is one whose running time is a polynomial function of the size of the input
    - Other algorithms are "bad"
  - This definition was adopted:
    - A problem is called tractable if there exists a "good" (polynomial time) algorithm that solves it.
    - A problem is called intractable otherwise.

Is this a valid cutoff?

Time complexity function	Size n: 10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

The class P

- The class P contains all decision problems that are decidable by an "algorithm" that runs in polynomial time.
  - Does this define a class of languages?
  - Yes…
    - The set of all problems whose encodings of "yes instances" (a language) is recognized by a TM M
    - M recognizes the above language in Polynomial Time.

The class P

- If we take the Church-Turing Thesis to be true:
  - P is robust:
    - The problems contained in P remain in P even if we change our basic model of computation.
      - A basic TM
      - A Sun, a Mac, even a PC running Windows XP!

The class P

- Are there actually problems in P?
  - Most everyday algorithms (e.g. sorting, searching) are in P
- Are there problems not in P:
  - Yes, we know that Self-Accepting is not in P since it isn't even solvable
- Are there decidable problems not in P?
  - Let's take a look…

Satisfiability (SAT)

- INSTANCE
  - A Logical expression containing
    - variables x_i
    - logical connectors &, |, and !
    - In conjuction normal form (C_1 & C_2 & C_3 … & C_n)
- PROBLEM
  - Is there an assignment of truth values to each of the variables such that the expression will evaluate to true.

Satisfiability (SAT)

- Example of an instance
  - (x_1 | x_2 | x_3) & (x_4 | !x_5) & !x_6 & (x_7 | x_8)
  - One Solution:
    - x_1 = true x_5 = false
    - x_2 = false x_6 = false
    - x_3 = false x_7 = true
    - x_4 = false x_8 = false

Satisfiability (SAT)

- Naïve algorithm to solve
  - Systematically consider all combinations of assignments of True and False values for all variables and test the expression
  - In the worst case, for n variables
    - $T(n) = O(2^n)$

Satisfiability (SAT)

- Can we do better?
  - No known polynomial algorithm
    - Either one doesn't exist
    - One exists and we haven't found it yet.

Non-deterministic TM

- Let's reconsider the NDTM
  - Same as the ordinary TM except:
    - The transition function will return a set of triplets
      - (q, x, D)
    - For each state / symbol combination, 0 or more transitions can be defined.
    - The machine can "choose" which transition to take.

Non-deterministic TM

- If we map all paths that can possibly be taken:
  - Number of paths will be exponential
  - Example:
    - Suppose at each of n states, for each character, there are 2 possible moves.
    - Number of paths 2^n

Satisfiability (SAT)

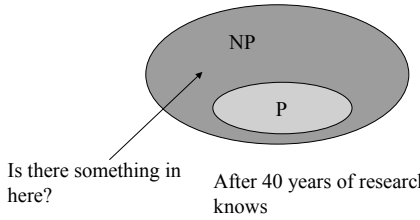
- Running SAT on a non-deterministic TM.
  - Step 1: "Guess" a set of boolean assignments to each variable (this is the non-deterministic part – 2^n different choices)
  - Step 2: Evaluate the truth of the entire expression using the guessed assignment (this part is deterministic)
  - Can certainly do each step in polynomial time

The class NP

- The class NP contains all decision problems that are decidable by a non-deterministic "algorithm" that runs in polynomial time.
- Represents algorithms that run in exponential time
  - Does this define a class of languages? -- Yes…
    - The set of all problems whose encodings of "yes instances" (a language) is recognized by a NDTM M
    - M recognizes the above language in Polynomial Time.
- Are there problems in NP
  - Well, for one SAT is

The class NP

- Clearly P is a subset of NP



Reducing one language to another

- Worked well for deciability…Let's use it here.
- Basic idea
  - Take an encoding of one problem
  - Convert it to another problem we know to be in P or NP
  - Conversion must be done in polynomial time!

Reducing one language to another

- Formally (for complexity)
  - Let L_1 and L_2 be languages over Σ_1 and Σ_2
  - We say L_1 is polynomial time reducible to L_2 ($L_1 \leq_p L_2$) if
    - There exists a Turing computable function
    - f: $\Sigma_1^* \rightarrow \Sigma_2^*$ such that
    - $x \in L_1$ iff $f(x) \in L_2$
    - f can be computed in polynomial time.

Reducing one language to another

- Informally (for complexity)
  - We can take any encoded instance of one problem
    - Use a TM to compute a corresponding encoded instance of another problem.
      - In polynomial time
    - If this other problem has a TM that recognizes the set of "yes encodings" in polynomial time (P), we can run that TM to solve the first problem.
    - If this other problem has a NDTM that recognizes the set of "yes encodings" in polynomial time (NP), we can run that TM to solve the first problem.

The class NP-complete

- The hardest of the problems in class NP are in the class of NP-complete problems:
- A language L is in NP if
  - $L \in NP$
  - For every other language $L_1 \in NP$
    - $L_1 \leq_p L$
- All NP-complete problems are "equally" difficult.

The class NP-complete

- Implications
  - Hardest problem
  - It's probably not worth looking for a solution for an NP-complete problem
  - How do we show a problem to be NP-complete
    - Reduction
    - We need a NP-complete problem to kick things off.

The class NP-complete

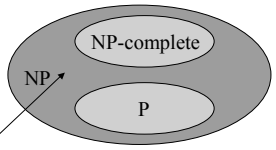
- Guess what?
  - SAT can be shown to be NP-complete
  - Cook's Theorem

NP-Complete and P

- If any a polynomial algorithm is found for any NP-complete problem
  - A polynomial algorithm can be found for all NP-complete problems (via polynomial reduction).
- In other words
  - For any $L \in$ NP-Complete,
    - If $L \in P$ then
    - P = NP
- Finding such a language is still an open problem.

NP-Complete and P

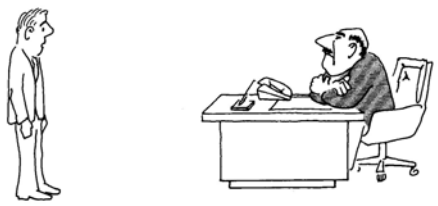
- The world of NP



Is there something in here?

After 40 years of research…nobody knows

Practical considerations



"I can't find an efficient algorithm, I guess I'm just too dumb."

Practical considerations

"I can't find an efficient algorithm, because no such algorithm is possible!"

Practical considerations

"I can't find an efficient algorithm, but neither can all these famous people."

Some NP-Complete Problems

- Traveling Salesman
  - INSTANCE
    - A finite set of "cities" C = { c_1, c_2, …, c_n }
    - A distance function d: $C^2 \rightarrow N$
      - d(c_i, c_j) = distance between c_i and c_j
    - Upper bounds B
  - PROBLEM
    - Is there a "tour" of all cities such that the distance traveled will be less than B?

Some NP-Complete Problems

- Traveling Salesman
  - PROBLEM
    - Said another way, is there an ordering of the cities $<c_{i1}, c_{i2}, \ldots, c_{in}>$ such that

[redacted]

Some NP-Complete Problems

- NDFA inequivalence
  - INSTANCE
    - Two nondeterministic finite automata A_1 and A_2 having the same input language Σ.
  - PROBLEM
    - Do A_1 and A_2 recognize different languages?
  - NOTE
    - Can be done in polynomial time if A_1 and A_2 are deterministic.

Some NP-Complete Problems

- Inequivalence of Programs with Assignments
  - INSTANCE
    - Finite set of X variables
    - Two programs P_1 and P_2
      - Each a sequence of the form $x_0 = (x1 == x_2) ? x_3 : x_4$
    - A value set V which defines possible values for each variable.

Some NP-Complete Problems

- Inequivalence of Programs with Assignments
  - PROBLEM
    - Is there an initial assignment of a value from V for each variable such that the two programs will yield different final results?

Some NP-Complete Problems

- Crossword Puzzle Construction
  - INSTANCE
    - A finite set $W \subseteq \Sigma^*$ of words
    - An n by n matrix A consisting of 1's and 0's
  - PROBLEM
    - Can an n by n crossword puzzle be built up including all words from W and blank squares consisting of the "0" cells in A?

Some NP-Complete Problems

- You want more?
  - "Computers and Intractability" by Michael R. Garey and David S. Johnson.

Dealing with NP-completeness

- Some NP-complete problems have solutions that are, on the average, polynomial.
- Restrict the problem
- Approximation
- Heuristics
- Buy a Supercomputer

Summary

- Complexity
- P
- NP
- NP-completeness

So There You Have It!

- I bet you thought I forgot!
  - What is a language?
  - What is a class of languages?
- Thanks for playing.

Where to next?

- Language Design / Compiler Construction
  - Formal Languages
  - Compiler Construction Lab
- Complexity & Computation
  - Complexity and Computability

The Theory Hall of Fame

- Stephen Cole Kleene
  - Regular expressions & FAs
- Noam Chomsky
  - The grammar guy
- W. Ogden
  - Pumping Lemma for CFLs
- Kurt Godel
  - Kicked off study of computability
- Alan Turing
  - Turing Machines

The Theory Hall of Fame

- Alonzo Church
  - Unlimited support for TMs
- Emil Post
  - Post Correspondence Problem
- H.G. Rice
  - Decision problems for TMs
- Stephen Cook
  - NP-completeness
- Aho, Hopcraft, and Ullman
  - Bell Labs guys – work on programming language

The Theory Hall of Fame

- YOU
  - If you find a polynomial solution to any problem in NP-complete!

Next Time

- Final exam review
- Saturday, Nov 16: Final exam
  - 7-1420
  - 12:30pm – 2:30pm
  - Closed book
  - 1 page study guide okay.
- Course Evaluations