CS216: Program and Data Representation
University of Virginia Computer Science
Spring 2006 David Evans

Lecture 8: Crash Course in Computational Complexity

http://www.cs.virginia.edu/cs216

Procedures and Problems

- So far we have been talking about **procedures** (how much work is our brute force subset sum algorithm?)
- We can also talk about **problems**: how much work is the subset sum problem?

What is a problem?

What does it mean to describe the work of a problem?

Problems and Solutions

- A **problem** defines a desired output for a given input.
- A **solution** to a problem is a procedure for finding the correct output for all possible inputs.
- The **time complexity** of a problem is the running time of the best possible solution to the problem

Subset Sum Problem

- **Input:** set of n positive integers, $\{w_0, \ldots, w_{n-1}\}$, maximum weight W
- **Output:** a subset S of the input set such that the sum of the elements of $S \leq W$ and there is no subset of the input set whose sum is greater than the sum of S and $\leq W$

What is the time complexity of the subset sum *problem*?

Brute Force Subset Sum Solution

```
def subsetsum (items, maxweight):
  best = {}
  for s in allPossibleSubsets (items):
     if sum (s) <= maxweight \
           and sum (s) > sum (best)
        best = s
  return best
```

Running time $\in \Theta(n2^n)$

What does this tell us about the time complexity of the subset sum *problem*?

Problems and Procedures

- If we know a *procedure* that is that is $\Theta(f(n))$ that solves a *problem* then we know the problem is $O(f(n))$.
- The subset sum **problem** is in $\Theta(n2^n)$ since we know a **procedure** that solves it in $\Theta(n2^n)$
- Is the subset sum **problem** in $\Theta(n2^n)$?

No, we would need to **prove** there is **no better procedure**.

Lower Bound

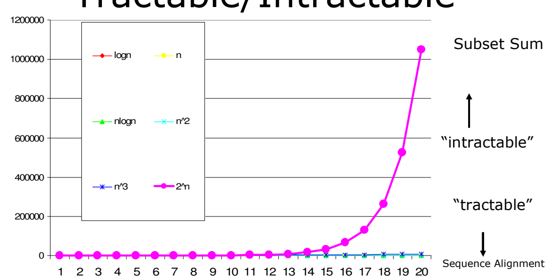
- Can we find an Ω bound for the subset sum problem?
- It is in $\Omega(n)$ since we know we need to at least look at every input element
- Getting a higher lower bound is tough

How much work is the Subset Sum Problem?

- Upper bound: $O(2^n)$
  Try all possible subsets
- Lower bound: $\Omega(n)$
  Must at least look at every element
- Tight bound: $\Theta(?)$

No one knows!

Tractable/Intractable



I do nothing that a man of unlimited funds, superb physical endurance, and maximum scientific knowledge could not do.
– Batman (may be able to solve intractable problems, but computer scientists can only solve tractable ones for large n)

Complexity Class P "Tractable"

Class P: problems that can be solved in polynomial time

$O(n^k)$ for some constant k.

Easy problems like sorting, sequence alignment, simulating the universe are all in **P**.

Complexity Class NP

Class NP: problems that can be solved in *nondeterministic* polynomial time

If we could try all possible solutions at once, we could identify the solution in polynomial time.

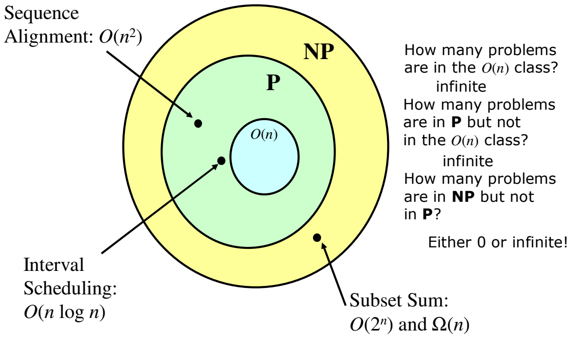
Alternately: If we had a magic guess-correctly procedure that makes every decision correctly, we could devise a procedure that solves the problem in polynomial time.

Complexity Classes

Class P: problems that can be solved in polynomial time ($O(n^k)$ for some constant k): "myopic" problems like sequence alignment, interval scheduling are all in **P**.

Class NP: problems that can be solved in polynomial time by a nondeterministic machine: includes all problems in **P** and some problems *possibly* outside **P** like subset sum

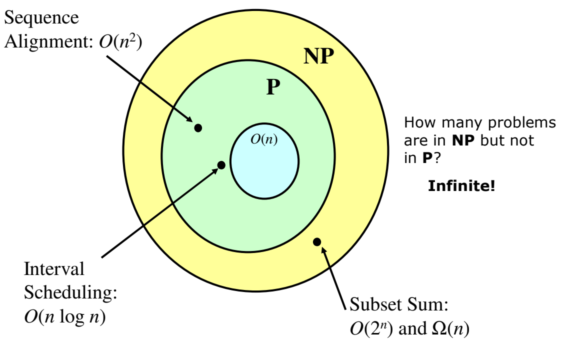
Complexity Classes: Possible View



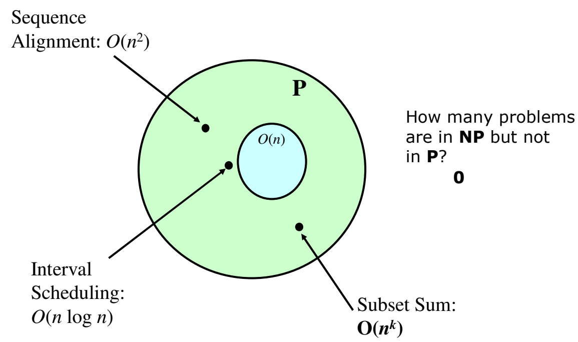
P = NP?

- Is P different from NP: is there a problem in NP that is not also in P
  - If there is one, there are infinitely many
- Is the "hardest" problem in NP also in P
  - If it is, then every problem in NP is also in P
- No one knows the answer!
- The most famous unsolved problem in computer science and math
  - Listed first on Millennium Prize Problems

Problem Classes if P ⊂ NP:



Problem Classes if P = NP:



Distinguishing P and NP

- Suppose we identify the *hardest* problem in NP - let's call it Super Arduous Task (SAT)
- Then deciding is P = NP should be easy:
  - Find a P-time solution to SAT ⇒ P = NP
  - Prove there is no P-time solution to SAT ⇒ P ⊂ NP

The Satisfiability Problem (SAT)

- Input: a sentence in propositional grammar
- Output: Either a mapping from names to values that satisfies the input sentence or **no way** (meaning there is no possible assignment that satisfies the input sentence)

SAT Example

Sentence ::= *Clause*
Clause ::= $Clause_1 \vee Clause_2$ **(or)**
Clause ::= $Clause_1 \wedge Clause_2$ **(and)**
Clause ::= ¬*Clause* **(not)**
Clause ::= (*Clause*)
Clause ::= *Name*

SAT $(a \vee (b \wedge c) \vee \neg b \wedge c)$
→ { a: **true**, b: **false**, c: **true** }
→ { a: **true**, b: **true**, c: **false** }

SAT $(a \wedge \neg a)$
→ no way

The 3SAT Problem

- Input: a sentence in propositional grammar, where each clause is a disjunction of 3 names which may be negated.
- Output: Either a mapping from names to values that satisfies the input sentence or **no way** (meaning there is no possible assignment that satisfies the input sentence)

3SAT / SAT

Is 3SAT easier or harder than SAT?

It is definitely not harder than SAT, since all 3SAT problems are also SAT problems. Some SAT problems are not 3SAT problems.

3SAT Example

Sentence ::= *Clause*
Clause ::= $Clause_1 \vee Clause_2$ **(or)**
Clause ::= $Clause_1 \wedge Clause_2$ **(and)**
Clause ::= ¬*Clause* **(not)**
Clause ::= (*Clause*)
Clause ::= *Name*

$$\text{3SAT} (\quad (a \vee b \vee \neg c) \\ \wedge (\neg a \vee \neg b \vee d) \\ \wedge (\neg a \vee b \vee \neg d) \\ \wedge (b \vee \neg c \vee d))$$

→ { a: **true**, b: **false**, c: **false**, d: **false**}

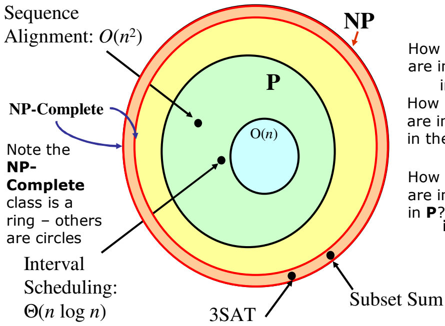
NP Completeness

- Cook and Levin proved that 3SAT was NP-Complete (1971): as hard as the hardest problem in NP
- If any 3SAT problem can be transformed into an instance of problem Q in polynomial time, than that problem must be no harder than 3SAT: Problem Q is **NP-hard**
- Need to show in NP also to prove Q is **NP-complete.**

Subset Sum is NP-Complete

- Subset Sum is in NP
  - Easy to check a solution is correct?
- 3SAT can be transformed into Subset Sum
  - Transformation is complicated, but still polynomial time.
  - A fast Subset Sum solution could be used to solve 3SAT problems

Problem Classes if P ≠ NP:



How many problems are in the $\Theta(n)$ class?
infinite
How many problems are in **P** but not in the $\Theta(n)$ class?
infinite
How many problems are in **NP** but not in **P**?
infinite

NP-Complete Problems

- Easy way to solve by trying all possible guesses
- If given the "yes" answer, quick (in P) way to check if it is right
  - Assignments of values to names (evaluate logical proposition in linear time)
  - Subset – check if it has correct sum
- If given the "no" answer, no quick way to check if it is right
  - No solution (can't tell there isn't one)

Traveling Salesperson Problem

- –Input: a graph of cities and roads with distance connecting them and a minimum total distant
- –Output: either a path that visits each with a cost less than the minimum, or "no".

- If given a path, easy to check if it visits every city with less than minimum distance traveled

Graph (Map) Coloring Problem

- –Input: a graph of nodes with edges connecting them and a minimum number of colors
- –Output: either a coloring of the nodes such that no connected nodes have the same color, or "no".

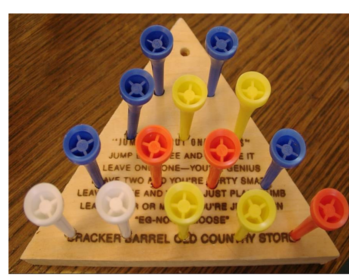
If given a coloring, easy to check if it no connected nodes have the same color, and the number of colors used.

Minesweeper Consistency Problem

- –Input: a position of n squares in the game Minesweeper
- –Output: either a assignment of bombs to squares, or "no".

- If given a bomb assignment, easy to check if it is consistent.

Pegboard Problem



Pegboard Problem

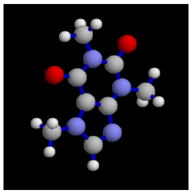
- Input: a configuration of n pegs on a cracker barrel style pegboard
- Output: if there is a sequence of jumps that leaves a single peg, output that sequence of jumps. Otherwise, output **false**.

If given the sequence of jumps, easy ($O(n)$) to check it is correct. If not, hard to know if there is a solution.

Drug Discovery Problem

– Input: a set of proteins, a desired 3D shape

– Output: a sequence of proteins that produces the shape (or impossible)



Caffeine

If given a sequence, easy (not really – this may actually be NP-Complete too!) to check if sequence has the right shape.

Is it ever *useful* to be confident that a problem is hard?

Charge

- PS3 can be turned in up till 4:50pm Friday: turn in to Brenda Perkins in CS office (she has folders for each section)
- Exam 2 will be handed out Wednesday
  – Send me email questions you want reviewed