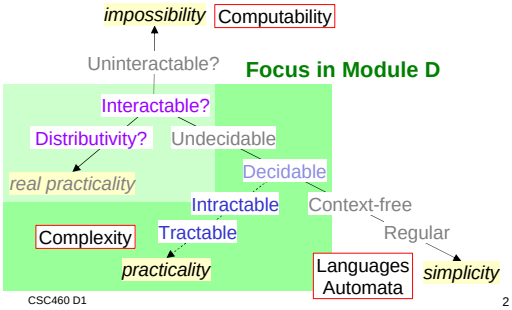
Module C Evaluation

Review your portfolio

- My comments are sporadic and scattered on Review Ex, Comprehensive Ex, Supp. Notes, and Reflective Essay.
- You are encouraged to clarify and discuss my comments.
- You can keep the folder till the next class; then, return it to me.

Overview: Theory of Computation



Module D Plan

D1	Polynomial vs. Exponential	
	– Intractability	
	– Nondeterministic Polynomial-time (NP)	
D2	NP-Completeness (NPC)	Mini research
D3	Space complexity	
D4	Parallel computation	
D5/6	Super-Turing computation	Reading
D7	Evaluation workshop	
YY	Practicum evaluation (Wed., Apr. 27)	

Content Goal 6: Complexity

a. Reviewed how to interpret the big *O* notation [game-theoretically, using the on-line graphing tool].
b. Understood that exponential growth with respect to the input data size (time complexity) is considered impractical ("intractable"), through examples.
c. Understood the class of "nondeterministic polynomial" (NP) problems, through examples. Also understood (i) why there are so many NP problems and (ii) why NP problems are essential for computer security.
d. Understood the notion of "polynomial time reducibility" and its impact on relating problems.
e. Understood the class of "nondeterministic polynomial complete" (NPC) problems, through examples. Also understood the basics of showing that a problem is in NPC.
f. Understood the essential difference between time and space complexity, as well as the hierarchy of time/space complexity.
g. Could explain (i.e., teach) this goal to CS students outside this class.

Content Goal 3: Interactivity

a. Understood the effects/limitations of parallel computation with respect to the three subareas of the traditional Theory of Computation.
b. Understood what kind of problems can*not* be adequately represented by TMs.
c. Understood the basics of super-Turing computation (more "powerful" than TMs) including its significance.
d. Was able to speculate where the theoretical underpinning of computer science should be heading, in order to offer robust analyses of a variety of computational problems.

Unit D1: Overview

- Review basic ideas about algorithm analysis
- Identify the gap between practical and impractical algorithms
- Explore some questionable cases with respect to practicality
- Preview Exercise D1 "Complexity Analysis"

Think Penn

How to Choose Particles in Japanese?

(in English-Japanese Translation)

Title: Osteoporosis in Active Women

(i) English: Osteoporosis is a disease of bone.

Japanese´: Osteoporosis-wa bone-of disease-is.

(ii) English: Young females are most affected.

Japanese´: Young female-ga most are.affected.

6/22/99 1/32 Nobo Komagata

Parsing English Texts

- Treating Hypertension in Active Patients: Which Agents Work Best With Exercise?

  Hypertension is one of the leading risk factors for cardiovascular morbidity and mortality. Unfortunately, the percentage of those who have hypertension may be on the rise as more and more Americans remain sedentary. Physicians, therefore, need to promote exercise programs (see "Exercisers: A Healthy Minority," below)—but without unwittingly sabotaging exercise capacity with antihypertensive medications. So in addition to knowing antihypertensive agents' impact on exercise, clinicians need to target therapy for varied hypertensive patients who participate in different sports and types of exercise.

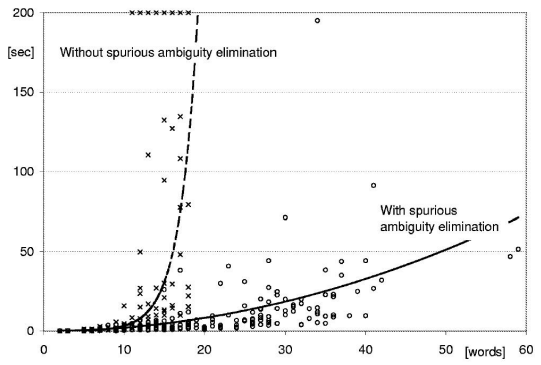


Figure 2: English Parser with and without Spurious Ambiguity Elimination

Practicality

- Where can we draw a line between *practical* and *impractical* algorithms, esp. facing *realistically large data*?
- Focus on the effect of input data size

Digital Circuits (Sample Problem #13)

- To design digital logic circuits, we need to analyze the relation between inputs and the output. For example, the following formula represents a circuit with four inputs and one output, each of which could take either 0 or 1 ('+' for OR, '·' for AND, and "' for NOT).

  output = $(i_1' + i_2 + i_3 + i_4) \cdot (i_1' + i_2 + i_3 + i_4') \cdot (i_1' + i_2' + i_3 + i_4')$
- Brute-force algorithm?

Cross-Country Interview (Sample Problem #16)

- Suppose that you are invited for interviews (graduate schools or industry jobs) in a number of cities in the US. Unfortunately, they do not pay for your travel. So, you will need to minimize the expense.
- Simple-minded algorithm?

Faster Algorithms

- Count from 0 to the user-specified n

```
n <- input
for (int i = 0; i < n; ++i) {
    output <- i;
}
```

- Display elements of a $n \times m$ matrix

```
n <- input; m <- input; // set elements in matrix
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        output <- matrix[i, j];
    }
}
```


Time Complexity

(asymptotic analysis)

- The speed/time performance of an algorithm as a function of the input data size, *ignoring the constant factor*
- Can be analyzed with the big-O notation (and its variants)

Big-O Notation

- Informal idea: Behaves roughly like such an such function popular use of '=' instead of '$\in$'
- Definition #1: $f(n) \in O(g(n))$ if there are constants $c > 0$ and $n_0 \geq 1$ such that $f(n) \leq c\, g(n)$ for every integer $n \geq n_0$.
- Definition #2:

$$f(n) \in O(g(n)) \text{ if } \lim_{n \to \infty} \frac{f(n)}{g(n)} = c \text{ for some } c \geq 0$$

Group Exercise 1

A. Explain why $n^2 \in O(2^n)$

B. Explain why $2^n \notin O(n^2)$

- Definition #1: $f(n) \in O(g(n))$ if there are constants $c > 0$ and $n_0 \geq 1$ such that $f(n) \leq c\, g(n)$ for every integer $n \geq n_0$.
- Definition #2:

$$f(n) \in O(g(n)) \text{ if } \lim_{n \to \infty} \frac{f(n)}{g(n)} = c \text{ for some } c \geq 0$$

Common Time Complexity Classes

- $O(\log n)$ = Logarithmic
- $O(n)$ = Linear
- $O(n \log n)$
- $O(n^2)$ = Quadratic
- $O(n^3)$ = Cubic
- Note: $O(n^i) \subseteq O(n^j)$ where $i \leq j$

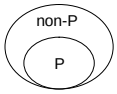
Examples?

Polynomial vs. Exponential

- Polynomial-Time (Bound) Problems
  P = $O(n^k)$ for any constant $k \geq 1$
  - Includes faster algorithms as well: $O(\log n) \subseteq O(n \log n) \subseteq$ P
- Exponential-Time (Bound) Problems
  Exp = $O(c^n)$ for any constant $c > 1$
  - Includes faster algorithms as well: P $\subseteq$ Exp
- (strictly) Exponential problems: Exp – P

Intractability

- Tractable problems = P
- Intractable problems = Complement of P
  - Including exponential problems (brute-force algorithms are not acceptable except for toy examples)



Is the gap clear cut?

Sample Problems, Revisited

- Digital Circuits = SATISFIABILITY (SAT)
  - Example: (*p* or *q*) and (*p* or (not *q*)) and ((not *p*) or *r*)
  - Search space: Exponential
  - Practical algorithm?
- Cross-country interviews = TRAVELING SALESPERSON (TSP)
  - Search space: Exponential
  - Practical algorithm?

SAT & TSP: Properties

- Search space: Exponential
- Exhaustive search ⇒ Exponential
- Checking an answer ⇒ Linear
- No polynomial algorithm has been found. ⇒ Most likely intractable
  - But nobody has proved the nonexistence. ⇒ Could be tractable
- Practical approach ⇒ Approximate

Approximate Solutions

- Classical local search (SAT, TSP)
- Greedy algorithm (SAT, TSP)
- Dynamic programming (TSP)
- Simulated annealing (SAT)
- Genetic algorithm (SAT, TSP)
- Neural network (TSP)

How to Solve It: Modern Heuristics by Michalewicz and Fogel

Nondeterminism

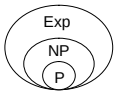
- Possibility of exponential branching
- Does not change the power of TMs
- Could change the time complexity of processing certain NP problems

cf. simulation of nondeterministic TM

Class NP

- Class of Nondeterministic polynomial-time problems
  - E.g., SAT, TSP, and many others
- Definition #1: A solution can be *verified* in polynomial time.
- Definition #2: A nondeterministic TM can decide in polynomial time.

Connection to computer security?



Group Exercise 2

A. Show that a scheduling problem (find a combination that satisfy some constraint) is in NP, using the both definitions.

B. Is a NP problem tractable? Explain?

•Definition #1: A solution can be *verified* in polynomial time.

•Definition #2: A nondeterministic TM can decide in polynomial time.

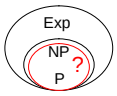
Sample NP Problems

- Cryptography (Sample Problem #11)
- Monkey Puzzle (#12)
- Professor Assignment (#14) ~ scheduling problem
- Knapsack Problem (#15)
- CPU Register Allocation (#17)
- Map Coloring (#18)

Why so many?

Unit Summary

- Use of the big-O notation for analyzing time complexity
- Hierarchy of time complexity classes: P (tractable), NP (open), Exp (intractable)
  - No polynomial algorithms have been found for any of the NP problems.
  - P = NP?



Summary Question

- Do you understand the definition of NP?
  - Don't leave until your answer becomes yes.
- Questions/Comments/Suggestions