Chapter 11: Limitations of Algorithms

Est modus in rebus, sunt certi denique fines.
(Horace)

Roughly, "Everything has limits."

Introduction

- The power of algorithms is limited.
- Lower bounds (Ω) to many problems are known, i.e., no algorithm can undercut them.
- Some problems cannot be fully solved.
- Many problems are considered intractable, such as the class of NP-complete problems.
- Numerical calculations are affected by truncation, roundoff, overflow, underflow, and cancellation.

P, NP, and NP-complete Problems

Tractable, Intractable, Undecidable Problems

- A *problem* specifies requirements on the inputs and the outputs. A *decision problem* has two possible outputs (yes/no).
- A problem is *tractable* if some algorithm can solve every instance in polynomial time, i.e., $O(n^d)$. Otherwise, the problem is *intractable*.
- A problem is *decidable* if some algorithm can solve every instance; otherwise the problem is *undecidable*.

The Halting Problem

- In the halting problem, the input can be any problem P and input I, and the output is whether P halts given input I.
- The halting problem is a decision problem.
- The halting problem is undecidable.
- Suppose there is an algorithm A that solves the halting problem. Construct a program Q as follows.
- $Q(P)$ halts if $A(P, P)$ says P doesn't halt on P. $Q(P)$ doesn't halt if $A(P, P)$ says P halts on P.
- What happens on $Q(Q)$?

P and NP Problems

- A decision problem is polynomial (in class P) if it can be solved in polynomial time.
- A decision problem is in class NP if it can be solved in nondeterministic polynomial time.
- A nondeterministic algorithm
  - inputs a decision problem instance I,
  - guesses a string S, and
  - outputs yes if some guess shows that I's answer is yes.
  - Example: Is 1517 a composite number? There is a guess (37) that shows the answer is yes.

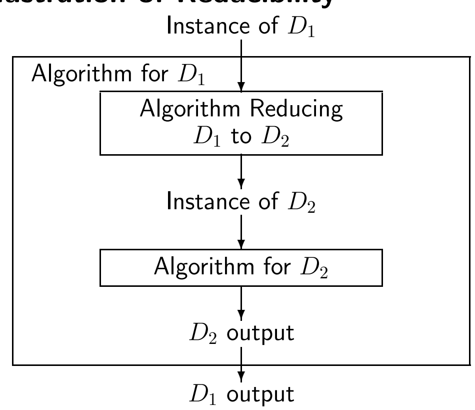
Interesting NP Problems

- Hamiltonian circuit. Does the graph have a cycle going through each vertex exactly once?
- Traveling salesman.
- Knapsack problem.
- Partition problem. Can a set of n integers be split into 2 subsets with the same sum?
- Bin packing. Given n items of different sizes and m bins of the same size, can the items be put into the bins?
- Graph coloring. Can a graph be colored with a given number of colors?
- Integer linear programming. Does a LP instance have an integer solution?

Polynomial Reducibility

- A decision problem D_1 is *polynomially reducible* to D_2 if a polynomial algorithm transforms instances of D_1 to D_2 such that:
  - all yes instances of D_1 are transformed to yes instances of D_2, and all no instances of D_1 are transformed to no instances of D_2.
- A polynomial algorithm for D_2 can be used to create a polynomial algorithm for D_1.
- Examples:
  - Hamiltonian circuit to traveling salesman
  - partition to bin packing
  - graph coloring to integer LP

Illustration of Reducibility



NP-Complete Problems

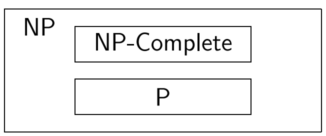
- A decision problem D is *NP-complete* if
  - D belongs to class NP, and
  - every NP problem is poly. reducible to D.
- CNF-satisfiability is NP-complete. An instance contains clauses. Clauses contain literals. Literals are boolean variables or negations. Can an assignment give each clause a true literal?

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

- To show that D_2 is NP-complete, show that:
  - D_2 belongs to class NP, and
  - NP-complete D_1 is poly. reducible to D_2

P=NP?

- A large number of problems have been shown to be NP-complete.
- A polynomial algorithm for any NP-complete problem would result in polynomial algorithms for all NP-complete problems.
- Because no such algorithm has been found, the best guess is that P $\neq$ NP.



- However, no one has proven P $\neq$ NP.

Approximating Infinite Calculations

- We know infinite sums for many expressions:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- Also infinite algorithms, e.g., calculate $\sqrt{n}$ by:

$x_0 = (1+n)/2$ and $x_{i+1} = (x_i + n/x_i)/2$

- A finite calculation results in *truncation errors*.
- For example, stopping at $i = 5$ results in:

$e \approx 2.71666\ldots$ where $e = 2.71828\ldots$

$\sqrt{1000} \approx 32.74\ldots$ where $\sqrt{1000} = 31.62\ldots$

Roundoff Errors

- Floating-point numbers have a limited accuracy.
- When does $1 + x$ result in x?
- When does $1 + x$ result in 1?
- Overflow. When does $x * x$ result in inf?
- Underflow. When does $x * x$ result in 0?
- Subtractive cancellation. The quadratic formula solution:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

can be inaccurate if b^2 is relatively large.