The Limits of Computing
Tom Cortina
Carnegie Mellon University

Comparing Big O Functions

Number of Operations

\[ O(2^n) \quad O(n^2) \quad O(n \log n) \quad O(n) \quad O(\log n) \quad O(1) \]

n (amount of data)
Classifications

• Algorithms that are \( O(N^k) \) for some fixed \( k \) are **polynomial-time** algorithms.
  – \( O(1) \), \( O(\log N) \), \( O(N) \), \( O(N \log N) \), \( O(N^2) \)
  – reasonable, **tractable**

• All other algorithms are **super-polynomial-time** algorithms.
  – \( O(2^N) \), \( O(N^N) \), \( O(N!) \)
  – unreasonable, **intractable**

Decision Problems

• A specific set of computations are classified as decision problems.

• An algorithm describes a **decision problem** if its output is simply YES or NO, depending on whether a certain property holds for its input.

• Example:
  Given a set of \( N \) shapes, can these shapes be arranged into a rectangle?
Decision Problems

Traveling Salesperson

- Given: a weighted graph of nodes representing cities and edges representing flight paths (weights represent cost)
- Is there a route that takes the salesperson through every city and back to the starting city with cost no more than $K$?
  - The salesperson can visit a city only once (except for the start and end of the trip).
Traveling Salesperson

Is there a route with cost at most 52? YES (Route above costs 50.)
Is there a route with cost at most 48? YES? NO?
Analysis

• If there are N cities, what is the maximum number of routes that we might need to compute?
• Worst-case: There is a flight available between every pair of cities.
• Compute cost of every possible route.
  – Pick a starting city
  – Pick the next city (N-1 choices remaining)
  – Pick the next city (N-2 choices remaining)
  – ...
• Maximum number of routes: \((N-1)(N-2)(N-3)\ldots(2)(1) = (N-1)!

Map Coloring

• Given a map of N territories, can the map be colored using K colors such that no two adjacent territories are colored with the same color?
• K=4: Answer is always yes.
• K=2: Only if the map contains no point that is the junction of an odd number of territories.
Map Coloring

- Given a map of $N$ territories, can the map be colored using 3 colors such that no two adjacent territories are colored with the same color?

Analysis

- Given a map of $N$ territories, can the map be colored using 3 colors such that no two adjacent territories are colored with the same color?
  - Pick a color for territory 1 (3 choices)
  - Pick a color for territory 2 (3 choices)
  - ... 
- There are $3^N$ possible colorings to check.
The Big Picture

• Intractable problems are solvable if the amount of data (N) that we’re processing is small.
• But if N is not small, then the amount of computation grows exponentially and the solutions quickly become intractable (i.e. out of our reach).
• Computers can solve these problems if N is not small, but it will take far too long for the result to be generated.
  – We would be long dead before the result is computed.

Are These Problems Tractable?

• For any one of these problems, is there a single tractable (polynomial) algorithm to solve any instance of the problem?
  – Computer scientists have not been able to prove that general tractable algorithms exist for these problems and we just haven’t found them yet.
  – Computer scientists have not been able to prove that general tractable algorithms do not exist for these problems so we should stop looking for these algorithms.
P and NP

- The class P consists of all those (decision) problems that can be solved on a computer in an amount of time that is polynomial in the size of the input.
- The class NP consists of all those (decision) problems whose positive solutions can be verified in polynomial time given the right information.

Example

- Finding the Minimum in an Array
  Solvable in polynomial time: yes
  Verifiable in polynomial time: yes
- Map Coloring
  Verifiable in polynomial time: yes
  Solvable in polynomial time: ?
- If a problem is in P, it must also be in NP.
- If a problem is in NP, is it also in P?
Complexity Classes

- **P**: If $P \neq NP$, then some decision problems can’t be solved in polynomial time. If $P = NP$, then all computable problems can be solved in polynomial time.

- **NP**: The Clay Mathematics Institute is offering a $1M prize for the first person to prove $P = NP$ or $P \neq NP$. ([http://www.claymath.org/millennium/P_vs_NP/](http://www.claymath.org/millennium/P_vs_NP/))

Watch out, Homer!
It Gets Worse...

• Tractable Problems
  – Problems that have reasonable, polynomial-time solutions

• Intractable Problems
  – Problems that may have no reasonable, polynomial-time solutions

• Noncomputable Problems
  – Problems that have no algorithms at all to solve them

Program Termination

• Can we determine if a program will terminate given a valid input?

• Example:
  ```python
  def mystery1(x):
      while (x != 1):
          x = x - 2
  ```
  – Does this algorithm terminate when x = 42?
  – Does this algorithm terminate when x = 12345?
Another Example

```python
def mystery2(x):
    while (x != 1):
        if (x % 2 == 0):
            x = x / 2
        else:
            x = 3 * x + 1

Does this algorithm terminate when x = 42?
Does this algorithm terminate when x = 12345?
Does this algorithm terminate for any positive x?
```

The Halting Problem

• Does a universal program HC exist that can take any program P and any input I for program P and determine if P terminates/halts when run with input I?
• Alan Turing showed that such a universal program HC cannot exist.
  – This is known as the Halting Problem.
Proof by Contradiction

• Assume a program HC exists that requires a program P and an input I.
  – HC determines if program P will halt when P is executed using input I.
  – We will show that HC cannot exist by showing that if it did exist we would get a logical contradiction.

\[
\begin{array}{c}
\text{P} \\
\text{I} \\
\hline
\text{HC}
\end{array}
\begin{array}{c}
\text{HALT CHECKER} \\
\hline
\text{YES} \\
\text{NO}
\end{array}
\]

HC outputs YES if P halts when run with input I
HC outputs NO if P does not halt when run with input I

What happens if the halt checker HC gets this program to check?

\begin{verbatim}
program evil(p):
    if HC(evil, evil) == true:
        while true { } // loop forever
    else:
        return          // halt
\end{verbatim}

This evil program asks the halt checker HC: “Do I halt?”
If HC says yes (true), the evil program will not halt.
If HC says no (false), the evil program halts.
Contradiction

• No matter what HC answers about the evil program, the evil program does the opposite, so HC can never answer the halting problem for the evil program.
  – Therefore, a universal halting checker HC cannot exist.
• We can never write a universal computer program that determines if ANY program halts with ANY input.
  – It doesn’t matter how powerful the computer is.
  – It doesn’t matter how much time we devote to the computation.

Contradiction in Real Life
The Limits of Computation

• **Tractable Problems**
  – Problems that have reasonable, polynomial-time solutions

• **Intractable Problems**
  – Problems that may have no reasonable, polynomial-time solutions

• **Noncomputable Problems**
  – Problems that have no algorithms at all to solve them