UNIT 14A
The Limits of Computing: Intractability

Announcement

• If you made a special arrangement for the final exam with me and have not gotten an email from me, come and see me at the end of the lecture.
Last Week

- Last programming assignment PA11 is due Friday, last day of classes
- Lab Exam 2 during the recitations
  - A sample exam posted on the Schedule page
  - More exam samples on the Resources page
  - Expanded Ruby Drills on the Resources page
- OLI Module 6: Computability. Covers the material of our last unit

Computability

- Can a computer solve any possible problem that we pose to it as a program?
- In this unit we will learn that
  - Some problems are intractable: solvable but requires so much time (or space) that effectively out of reach
  - Some problems are unsolvable: no matter how fast the computer is (how big the memory is) it is impossible to solve them
Why Study Unsolvability?

- Practical: If we know that a problem is unsolvable we know that we need to simplify or modify the problem
- Cultural: Gain perspective on computation

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?
The Monkey Puzzle

• Given:
  – A set of \( N \) square cards whose sides are imprinted with the upper and lower halves of colored monkeys.
  – \( N \) is a square number, such that \( N = M^2 \).
  – Cards cannot be rotated.

• Problem:
  – Determine if an arrangement of the \( N \) cards in an \( M \times M \) grid exists such that each adjacent pair of cards display the upper and lower half of a monkey of the same color.

Example

• Is there a YES answer to the decision problem?
• If there is, is the problem tractable in general?
Algorithm

Simple brute-force algorithm:

• Pick one card for each cell of M X M grid.
• Verify if each pair of touching edges make a full monkey of the same color.
• If not, try another arrangement until a solution is found or all possible arrangements are checked.
• Answer "YES" if a solution is found. Otherwise, answer "NO" if all arrangements are analyzed and no solution is found.

Analysis

Suppose there are N = 9 cards (M = 3)

The total number of unique arrangements for N = 9 cards is:

9 * 8 * 7 * .... *1 = 9! (9 factorial)
Analysis (cont’d)

For N cards, the number of arrangements to examine is N! (N factorial)

If we can analyze one arrangement in a microsecond:

<table>
<thead>
<tr>
<th>N</th>
<th>Time to analyze all arrangements</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>362,880 µs</td>
</tr>
<tr>
<td>16</td>
<td>20,922,789,888,000 µs (app. 242 days)</td>
</tr>
<tr>
<td>25</td>
<td>15,511,210,043,330,985,984,000,000 µs</td>
</tr>
</tbody>
</table>

Reviewing the Big O Notation (1)

• We use the big O notation to indicate the relationship between the amount of data to be processed and the corresponding amount of work.

• For the Monkey Puzzle
  – Amount of data to be processed: the number of board arrangements
  – Amount of work: Number of operations to check if the arrangement solves the problem

• For very large n (size of input data), we express the number of operations as the (time) order of complexity.
Growth of Some Functions

Big O notation:
- gives an asymptotic upper bound
- ignores constants

Any function \( f(n) \) such that \( f(n) \leq cn^2 \) for large \( n \) has \( O(n^2) \) complexity

Quiz on Big O

- What is the complexity in big O for the following descriptions
  - The amount of computation does not depend on the size of input data  \( O(1) \)
    For example, work is always 3 operations, or 5 operations
  - If we double the input size the work is doubles, if we triple it the work is 3 times as much  \( O(n) \)
    For example, work is 2n + 5, or 8n
  - If we double the input size the work is 4 times as much, if we triple it the work is 9 times as much  \( O(n^2) \)
    For example, work is 2\(n^2\) + 5, or 8\(n^2\)
  - If we double the input size, the work has 1 additional operation  \( O(\log n) \)
    For example, work is 2 \(\log n\) + 5
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

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: __________
Number of Paths to Consider

Number of all possible paths = Number of all possible permutations of $N$ nodes = \( N! \)

Observe ABCGFDE is equivalent to BCGFDEA

Number of all possible unique paths = \( \frac{N!}{N} = N - 1! \)

Observe ABCGFDE has the same cost as EDFGCBA

Number of all possible paths to consider = \( \frac{(N-1)!}{2} \)

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)
  - ...
- Worst-case complexity: \( O(N!) \)  
  Note: \( N! > 2^N \) for every $N > 3$. 
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 $3 \times 3 \times \ldots \times 3 = 3^N$ possible colorings.

Satisfiability

- Given a Boolean formula with $N$ variables using the operators AND, OR and NOT:
  - Is there an assignment of boolean values for the variables so that the formula is true (satisfied)?
    - Example: $(X \text{ AND } Y) \text{ OR } (\text{NOT } Z \text{ AND } X)$
    - Truth assignment: $X = \text{True}, Y = \text{True}, Z = \text{False}$.
- How many assignments do we need to check for $N$ variables?
  - Each symbol has 2 possibilities ... $2^N$ assignments
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.

What’s Next

• For a specific decision problem, is there single tractable (polynomial-time) algorithm to solve any instance of this problem?
• If one existed, can we use it to solve other decision problems?
• What is one of the big computational questions to be answered in the 21\textsuperscript{st} century?