Introduction to Algorithmic Complexity: Searching

Instructor: Nihshanka Debroy
(Slides borrowed from Tammy Bailey and Dr. Forbes)
Algorithm Analysis

- Determine the amount of resources an algorithm needs to run (computation time, space in memory)
- **Running time**: number of basic operations performed
  - additions, multiplications, comparisons
  - usually grows with size of input
  - faster to add 2 numbers than to add 2,000,000!
Schedule students, minimal conflicts

- Given student requests, available teachers
  - write a program that schedules classes
  - Minimize conflicts

- Add a GUI too
  - Web interface
  - ...
  - ...
One better scenario

I can’t write this program because it’s provably impossible
Another possible scenario

I can’t write this program but neither can all these famous people
Limitations of computer science

Major reasons useful calculations cannot be done:

- Execution time of program is too long
  - problem can take years or centuries to finish
- Problem is not computable
  - no computer program can solve the problem
- We do not know how to write a program to solve the problem
  - problems that can conceivably be solved
  - includes many problems in artificial intelligence and computer vision, such as understanding language, object recognition, tracking, prediction
Functions

- A function is a correspondence between a collection of possible *input values* and a collection of *output values* such that each possible input is assigned a unique output.
- Example: the addition function
  - inputs are value pairs, outputs are values representing the sum of each input pair
  - output is unique, exactly one value for each input pair
- Process of determining output for given input is called *computing* or *evaluating* the function.
- The ability to evaluate functions gives us the ability to solve problems
  - to solve the addition problem we must evaluate the addition function.
Function evaluation

- A decision problem is a question with a YES/NO answer:
  - Is 613511 a prime number?
  - Is my birthday on a Sunday next year?
  - Did I pass the CPS1 final exam?
- Sometimes a particular answer from a set of possible solutions is required
  - What is the smallest prime factor of 613511?
  - On what day of the week is my birthday next year?
  - What grade will I get on the CPS1 final exam?
- The process of answering questions where a particular answer is uniquely determined from the given information can be viewed as the evaluation of a function
Computable functions

- Algorithms are the means by which we formally express the evaluation of a function.
- Can we design algorithms to evaluate all functions?
  - No!
- Exist functions so complex that there is no well-defined step-by-step process for determining their output based on their input values.
- The evaluation of these functions lies beyond the capabilities of any algorithmic system.
- These functions are said to be **noncomputable**.
- Functions whose output values can be determined algorithmically from their input values are said to be **computable**.
  - functions having an algorithm that solves them
“Does extraterrestrial life exist?”

- Q: Is it possible to write a Java program to answer the following question: *Does extraterrestrial life exist?*
- A: Yes!

- The question is answered either by a program that returns “yes” or by a program that returns “no”.
- We just don’t know which program is correct.
A noncomputable problem

- Problem: List all subroutines that input an integer and return an integer
  - here is one short subroutine:
    ```c
    int sub1(int x)
    {
        return 1;
    }
    ```
  - and here is another:
    ```c
    int sub2(int x)
    {
        return 2;
    }
    ```
- Can make infinitely many such subroutines
Types of Problems

- **Tractable**
  - Problems that *can* be solved by a computer in a “reasonable” amount of time.

- **Intractable**
  - Problems that *can’t* be solved by a computer in a “reasonable” amount of time,
  - But *can* be solved eventually.

- **Non-computable**
  - Problems that can *never* be solved by a computer.
Is there a path from Ann to Bob?
Is there a path from Ann to Bob?
Can you color this map with 4 colors?
Can you color this map with 3 colors?
Can you color this map with 3 colors?
Can you color this graph with 3 colors?
Not impossible, but impractical

- **Towers of Hanoi**
  - How long to move \( n \) disks?

- **What combination of switches turns the light on?**
  - Try all combinations, how many are there?
  - Is there a better way?
Travelling Salesperson

- Visit every city exactly once
- Minimize cost of travel or distance
- Is there a tour for under $2,000? less than 6,000 miles?
- Is close good enough?

Try all paths, from every starting point -- how long does this take?

a, b, c, d, e, f, g
b, a, c, d, e, f, g ...
Halt or not

- Does the following code eventually terminate?

```python
while (x > 1)
{
    if (x > 2)
        x = x - 2;
    else
        x = x + 2;
}
```

- What if x is 8? How about 9?
Halting

```c
int programA( int x )
{
    while( x == x )
        x = x;
    return x;
}
```

```c
int programB( int x )
{
    while( x > 10 )
        x = x;
    return x;
}
```

- Programs that run forever do not halt
  - programA
  - programs with infinite loops
- Programs may halt on some inputs but not on others
  - programB
- Programs with no loops always halt
  - programs composed of assignment statements, if statements and the return statement
Programs that read programs

- Almost every problem related to the behavior of programs is noncomputable
  - programs to check for property X in the behavior of all other programs
  - halting, equivalence, printing, correctness
- We can write programs to check almost any syntactic feature of programs
  - a compiler is a program that reads programs
  - we can measure the length of programs, number of statements, characters, arithmetic expressions
Computable or noncomputable?

1. Write a program that inputs a sequence of characters and returns true if it is a legal Java program.
2. Write a program that inputs a sequence of characters and returns true if any permutation of these characters is a legal Java program.
3. Write a program that inputs a program in any programming language and returns true if the characters can be rearranged to form a legal Java program.
4. Write a program that inputs a Java program and returns true if the program computes the sum of its inputs.
5. Write a program that inputs two Java programs and returns true if both programs always produce the same output.
Computable or noncomputable?

1. Computable
2. Computable
3. Computable
4. Noncomputable
5. Noncomputable
Program execution time

- How long a program takes to perform computations on large inputs
- Computable problems are basically divided into two classes
- Computations that can be completed in a reasonable time period are called **tractable**
  - a computation is tractable if its running time on N inputs is logarithmic or polynomial in N
- **Intractable** computations are those that cannot be realistically completed except for small examples
  - a computation is intractable if its running time on N inputs is exponential in N
Tractable or intractable?

1. Write a program that inputs a sequence of characters and returns true if it is a legal Java program
   - tractable

2. Write a program that inputs a sequence of characters and returns true if any permutation of these characters is a legal Java program
   - intractable

3. Write a program that inputs a program in any programming language and returns true if the characters can be rearranged to form a legal Java program
   - intractable
Examples

- Tractable problems
  - searching or sorting $N$ inputs
  - computing the sum of positive integers $\leq N$
  - finding the largest element in a $N \times N$ array
  - counting the number of characters in a Java program

- Intractable problems
  - computing all permutations of $N$ inputs
  - Towers of Hanoi
  - Traveling salesperson
  - Bin packing
  - Graph coloring
Traveling salesperson

- A traveling salesperson must visit each of some number of cities before returning home
  - knows the distance between each of the cities
  - wants to minimize the total distance traveled while visiting all of the cities
- In what order should the salesperson visit the cities?

Try all paths, from every starting point — how long does this take? 
\{a, b, c, d, e, f, g\}, 
\{b, a, c, d, e, f, g\}, ...
Graph coloring

- Color the vertices of a graph such that no edge connects two identically colored vertices using the minimum number of colors.
Linear Growth

- Grade school addition
  - Work is proportional to number of digits $N$
  - *Linear* growth: $kN$ for some constant $k$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- How many reads? How many writes? How many operations?

$N = 2$

$N = 4$
Quadratic Growth

- Grade school multiplication
  - Work is proportional to square of number of digits $N$
  - Quadratic growth: $kN^2$ for some constant $k$

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
1 \\
3 \\
\hline
3 \\
\end{array}
\quad
\begin{array}{c}
\quad
7 \\
\quad
8 \\
\quad
\star 4 \\
\quad
2 \\
\hline
1 \\
5 \\
\quad
6 \\
\quad
\end{array}
\quad
\begin{array}{c}
\quad
4 \\
\quad
2 \\
\quad
7 \\
\quad
8 \\
\star \quad
6 \\
\quad
8 \\
\quad
4 \\
\quad
2 \\
\hline
8 \\
5 \\
\quad
5 \\
\quad
6 \\
\end{array}
\]

$N = 2$

- How many reads? How many writes? How many operations?

\[
\begin{array}{c}
2 \\
5 \\
6 \\
6 \\
8 \\
0 \\
0 \\
0 \\
0 \\
\hline
2 \\
9 \\
2 \\
7 \\
0 \\
0 \\
7 \\
6 \\
\end{array}
\quad
\begin{array}{c}
\quad
N = 4
\end{array}
\]
Searching

- Determine the location or existence of an element in a collection of elements of the same type
- Easier to search large collections when the elements are already sorted
  - finding a phone number in the phone book
  - looking up a word in the dictionary
- What if the elements are not sorted?
Sequential search

- Given a collection of $n$ unsorted elements, compare each element in sequence.
- Worst-case: Unsuccessful search
  - search element is not in input
  - make $n$ comparisons
  - search time is $linear$
- Average-case:
  - expect to search $\frac{1}{2}$ the elements
  - make $n/2$ comparisons
  - search time is $linear$
public class LinearSearch
{
    public static void main(String[] args) {
        int[] input = {4, 5, 23, 36, 44, 101, 12, 19, 156, 77, 81, 2, 81};
        int goal = 0;
        System.out.print("Enter a number to search for: ");
        goal = Keyboard.readInt();
        boolean inArray = linearSearch(input, goal);
        System.out.println("Was the entry found: "+inArray);
    }

    public static boolean linearSearch(int[] input, int goal)
    {
        boolean found = false;
        for(int i=0; i<input.length; i++)
        {
            if(input[i] == goal)
            {
                found = true;
            }
        }
        return found;
    }
}
Searching sorted input

- If the input is already sorted, we can search more efficiently than linear time
- Example: “Higher-Lower”
  - think of a number between 1 and 1000
  - have someone try to guess the number
  - if they are wrong, you tell them if the number is higher than their guess or lower
- Strategy?
- How many guesses should we expect to make?
Logarithms Revisited

- Power to which any other number $a$ must be raised to produce $n$
  - $a$ is called the base of the logarithm
- Frequently used logarithms have special symbols
  - $\lg n = \log_2 n$ logarithm base 2
  - $\ln n = \log_e n$ natural logarithm (base e)
  - $\log n = \log_{10} n$ common logarithm (base 10)
- If we assume $n$ is a power of 2, then the number of times we can recursively divide $n$ numbers in half is $\lg n$
Best Strategy

- Always pick the number in the middle of the range
- Why?
  - you eliminate half of the possibilities with each guess
- We should expect to make at most
  \[ \lg 1000 \approx 10 \] guesses
- Binary search
  - search \( n \) sorted inputs in logarithmic time
Binary search

- Search for 9 in a list of 16 elements

```
1 3 4 5 5 7 9 10 11 13 14 18 20 22 23 30
```

```
1 3 4 5 5 7 9 10
```

```
5 7 9 10
```

```
9 10
```

```
9
```
Sequential vs. binary search

- Average-case running time of sequential search is linear
- Average-case running time of binary search is logarithmic
- Number of comparisons:

<table>
<thead>
<tr>
<th>n</th>
<th>sequential search</th>
<th>binary search</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>4096</td>
<td>2048</td>
<td>12</td>
</tr>
<tr>
<td>65536</td>
<td>32768</td>
<td>16</td>
</tr>
</tbody>
</table>
Running Time

- Linear Search: $O(n)$ time
- Binary Search: $O(lg n)$ time