CSC 236 T03: Search

**This is a team assignment designed as an in-class activity.**

Created by Clif Kussmaul, Muhlenberg College

See <http://cspogil.org/tiki-index.php> for more information.

Modified by Jan Pearce, Berea College

**Directions for use:**

* To use this form effectively, sign into a Google account.
* Then under “File” choose “Make a Copy” in order to be able to edit.
* Share with all team members, but allow Recorder to do the recording.
* Each yellow box should be filled with an appropriate team response..
* Download as *yourteamname-csc236T03.docx* and upload to Moodle

**Introduction**

In computing, we often must **search** in a set or a list for a particular item. As computer scientists, we are particularly interested in “big data”, so we might need to search a very large set or list, with thousands or millions of values. For example, the US Library of Congress has roughly 22 million cataloged books, and over 100,000,000 total items.

In this team activity, we use a simple game to explore some basic searching algorithms. This will help us explore more general concepts in algorithm design and analysis, so studying searching is useful even though few of us may need to implement searching algorithms from first principles, since efficient techniques are part of the application programming interface (API) of many software libraries.

First, confirm the roles and complete the form below for assigned roles of each member.

**Member Roles**

* If you have only four people, combine Quality Control Officer & Process Analyst
* If you have only three people, also combine Recorder & Spokesperson20

|  |  |
| --- | --- |
| **Team Roles** | **Member Name** |
| **Facilitator:** | **Conner Bondurant** |
| **Recorder:** | **Tradd Schmidt** |
| **Spokesperson:** | **Basanta Phuyal (Big B)** |
| **Quality Control Officer:** | **Rusty Dotson** |
| **Process Analyst:** | **Rusty Dotson** |

***Hi-Lo Game***

Hi-Lo is a number guessing game with simple rules.

1. There are two players: A and B.
2. Player A thinks of a number from 1 to 100.
3. Player B guesses a number.
4. Player A responds with “too high”, “too low”, or “you win”.
5. Players B and A continue to guess & respond until B wins (or gives up).

***I. (7 min) Player Strategies***

|  |  |
| --- | --- |
| **start time:** | 10:47 |

1. (2 min) Play the game a few times to ensure that everyone understands the rules.
2. (2 min) List up to 3 ways you can think of which would serve to clarify the rules.

|  |
| --- |
| **Ways to Clarify Rules** |
| 1. The number has to be an integer. 2. Once the number has been picked, it cannot be changed. |

1. (3 min) Describe 4-5 different strategies that Player B could use to guess numbers. Try to have a mixture of simple and clever strategies.Name each strategy and list it in the first column of the following table:.

|  |  |
| --- | --- |
| **Strategy Name** | **Strategy Description** |
| Complete Search | Start your guesses at 1 and then guess linearly towards 100 until you reach the target number. |
| Binary Search | Start in the middle of the range, then split the range in half and repeat with the range that is marked as the one that contains the number until you win. |
| Random search | Guess random numbers until the target number is guessed. |
| Guess by 10’s | Guess every 10 numbers until you find a range of 10 numbers that the target is in. |
|  |  |

***II. (8 min) Comparing Strategies***

|  |  |
| --- | --- |
| **start time:** | 10:56 |

1. (2 min) Evaluate each strategy with regard to how **quickly** it will find the right answer, by rank ordering from 1 (least guesses) to 5 (most guesses). Add the rankings to the following table:

|  |
| --- |
| **Quick** |
| 1. Binary search 2. Guess by 10’s 3. Complete search 4. Random search |

1. (2 min) Evaluate each strategy with regard to how easy it is to describe or specify, by rank ordering from 1 (easiest) to 5 (hardest). (Suppose you had to explain each strategy to a first-grader so that she could play the game.) Add the ranking to the following table:

|  |
| --- |
| **Easy** |
| 1. Random Search 2. Complete search 3. Guess by 10’s 4. Binary Search |

1. (1 min) For each strategy, multiply the quick rank above by the easy rank above and add the product to the following table:

|  |
| --- |
| **Quick x Easy (Product )** |
| 1. Random search = 4 2. Binary search = 4 3. Guess by 10’s = 6 4. Complete search = 6 |

1. (3 min) In a paragraph of complete sentences, describe the relationships between the two sets of rankings. Before you continue, review your progress.

|  |
| --- |
| **Relationship between Quick and Easy** |
| The strategies have a reverse relationship. This means that if it is the best algorithm, it is the hardest to explain. More elaborate systems tend to allow for more effective solutions. |

***III. (10 min) Worst & Average Case Performance***

|  |  |
| --- | --- |
| **start time:** | 11:04 |

1. (2 min) Brainstorm, discuss and list the pros & cons of measuring program speed with a stopwatch:

|  |  |
| --- | --- |
| **Pros of Stopwatch** | **Cons of Stopwatch** |
| Easy use. | A program can run faster than our fingers |

1. (3 min) For each strategy, determine the **worst case** (maximum) number of guesses required to win when N=100.. Add the numbers and a brief to the the following table:

|  |  |
| --- | --- |
| **Strategy Name** | **(Maximum) Worst Case Number (#) of Guesses for N=100** |
| Binary Search | 7: the ceiling of log2(100) |
| Guess by 10’s | 19: If the target number is 99 and you start at 10, you will guess up to 100 which is 10 guesses, and then you know that the target number is from 90 to 99. Then you guess 91 to 99 which is another 9 guesses. |
| Random Search | Worse case is that it falls into a loop from pseudo-randomness and never finds the right number. |
| Complete search | 100: if the number was 100, you would guess all the numbers. |
|  |  |

1. (3 min) For each strategy, determine the **average case** (typical) number of guesses required to win when N=100.. Add the numbers to the worksheet in a column labeled **Average**. Note that the **minimum** number of guesses is always 1 because of luck!

|  |  |
| --- | --- |
| **Strategy Name** | **(Typical) Average Case Number (#) of Guesses for N=100** |
| Binary Search | 5.7102 |
| Guess by 10’s | 9.5 |
| Random Search | 50 (assuming it always finishes, otherwise mathematically infinite) |
| Complete Search | 50 |
|  |  |

1. (2 min) List 3 reasons why it would be useful to have more precise, quantitative ways to measure and discuss the speed of an algorithm. Before you continue, review your progress.

|  |
| --- |
| **Three Reasons for More Precision in Measuring “Speed”** |
| There might be two algorithms with millisecond differences in between them. Therefore you need something that can measure accurately on very low levels of numbers. |
| It allows for better comparisons between algorithms. |
| You can use precise time measurements on a small scale to predict runtime of a large scale input |

***IV. (10 min) Effect of Input Size***

|  |  |
| --- | --- |
| **start time:** | 11:38 |

1. (3 min) Assume that Player A chooses a number from 1 to 1000.  
For each strategy, what are the worst case & average case number of guesses?  
Add these numbers to the following worksheet:

|  |  |  |
| --- | --- | --- |
| **Strategy Name** | **Worst # for N=1000** | **Average # for N=1000** |
| Complete | 1000 | 500 |
| Binary search | 10 | 8.9758 |
| random | It runs infinitely | 500 assuming it finishes |
| Guess by tens | 109 | 55 |
|  |  |  |

2. (4 min) Assume that Player A chooses a number from 1 to N.   
(For example, N=100, N=1000, N=1,000,000) For each strategy, compute the worst case & average case number of guesses in terms of N. (Hint: you’ve already done N=100 and N=1000, so consider other values before generalizing to N.)  
Add these mathematical expressions to the following worksheet:

|  |  |  |
| --- | --- | --- |
| **Strategy Name** | **Worst # for General N** | **Average # for General N** |
| bin | log2(n) |  |
| complete | n | n/2 |
| By tens | (n/10) + 9 | (n/20) + 5 |
| random | forever | n/2 |
|  |  |  |

3. (3 min) Describe the pros & cons of analyzing performance in terms of input size N.

|  |  |
| --- | --- |
| **Pros of Input Size Analysis** | **Cons of Input Size Analysis** |
| Allows for approximation of runtime speed at implementation time and allows for an overview of how many times your program will check a number based on your sample size. | Some algorithms are really complicated and are awful to actually analyze |

***V. (5 min) Activity Wrap-Up***

1. (3 min) Complete the following table individually on the most significant or surprising thing learned in this activity by each team member.

|  |  |
| --- | --- |
| **Member Name** | **Most Significant or Most Surprising Thing Learned** |
| **Conner bondurant** | **Guess by tens was a somewhat legit function.** |
| **Tradd Schmidt** | **The speed at which binary becomes a better function than complete as the sample size increases.** |
| **Basant Phuyal** | **I was surprised at the difficulty level of a kid compared to the most efficient algorithm. It was the opposite.** |
| **Rusty Dotson** |  |
|  |  |

1. (1 min) Please offer any suggestions for improvement of this activity from the team:

|  |
| --- |
| **Suggestions for improvement** |
| For the rules improvement, I would add a hint of “think outside the box, about things that might break the game.” |

To submit, the Recorder will download as *yourteamname-T03.docx* and upload to Moodle while all other members will simply write in the comment section for the T03 spot the names and roles of all team members.