Contents

Facilitator Information

Learning Objectives (for content & process)

After completing this activity, learners should be able to:

- Explain the pros & cons of various simple search algorithms.
- Explain common tradeoffs between the complexity and performance of algorithms.
- Assess the performance of simple algorithms as a function of their input size N.

This activity should help learners develop teamwork and critical thinking skills.

Prerequisites (for content & process)

Before starting this activity, learners should have:

- Previous experience with POGIL (useful but not necessary).
- Programming experience is not required for the activity, but is for some applications.

Preparation

1. Print 1 activity per 2-3 students; print worksheet and graph on **separate pages** (not 2-sided)
2. Optional: Provide the worksheet on the board, a poster, or in presentation software, so teams can see each others' work easily.

Activity Notes

- If students are unfamiliar with POGIL, it may be simpler to have teams of 3 (no Reflector).
- The facilitator should spend a minute or two introducing the activity.
  While teams work, the facilitator should circulate among the teams to monitor progress and help with problems, but should try not to provide or confirm answers to key questions.
  For more information, see http://cspogil.org or http://pogil.org
- II. Player Strategies
  - II.1. Report out: describe an algorithm. Summarize in table for class.
  - II.2-4. Report out: rankings. Summarize in table for class.
  - II.5. Mini-lecture: Describe specific examples where O() performance changes.
- III. Worst & Average Case
  - III.1. Some teams may get stuck on random guessing.
    Have everyone write down 1-2 random numbers, then list on board.
    Notice: duplicates; min & max; multiples of 5, 10; odd vs. even; etc.
    (see Random Number Distribution Table for example)
  - III.2. Report out: worst & average case results. Summarize in table for class.
- IV. Parts of this may be too complex for some CS0/CS1 students.

Things to Do

- ? Build histograms to better understand average & worst case (see right)
- Add more guidance on mean & max counts
- Add sections on: implications for search: effort required to sort, tradeoffs.
- Add applications, resources, rubrics
- Add summary section.
- Add supporting resources.

# of guesses	possible values guessed	# of values	total # values
1	50	1	1
2	25,75	2	3
3	12,37,62,87	4	7
4	6 … 94	8	15
5	3 … 97	16	31
6	2 … 99	32	63
7	1 … 100	34	100

Activity History

2010-06 drafted by Clif Kussmaul clif@kussmaul.org
2010-07...08 split into 2 activities (search and O() notation), revised
2011-01 piloted, revised
2012-12 revised
2013-08 revised based on experiences in India
2014-01 **reviewed and endorsed by The POGIL Project**

Introduction

start time:

In computing, we often must **search** in a set for a particular item. As computer scientists, we are particularly interested in searching very large sets, with thousands or millions of values. For example, the Harvard University Library has roughly 16,000,000 volumes, and the US Library of Congress has roughly 22 million cataloged books, and over 100,000,000 total items. In this activity, we use a simple game to explore some basic searching algorithms. This will also help us explore more general concepts in algorithm design and analysis, so studying searching is useful even though very few of us may need to implement searching algorithms, since efficient techniques are part of most software libraries (APIs).

Before you start, complete the form below to assign a role to each member.
If you have 3 people, combine Speaker & Reflector.

Team		**Date**	
Team Roles		**Team Member**	
Recorder: records all answers & questions, and provides copies to team & facilitator.			
Speaker: talks to facilitator and other teams.			
Manager: keeps track of time and makes sure everyone contributes appropriately.			
Reflector: considers how the team could work and learn more effectively.			

Reminders:

1. *Note the time whenever your team starts a new section or question.*
2. *Write legibly & neatly so that everyone can read & understand your responses.*

start
time:

I. (6 min) Hi-Lo Game

Hi-Lo is a number guessing game with simple rules, played by school children.

a. There are two players – A and B.
b. Player A thinks of a number from 1 to 100.
c. Player B guesses a number.
d. Player A responds with "too high", "too low", or "you win".
e. Players B and A continue to guess & respond until B wins (or gives up).

1. (1 min) How many different responses can player A give? ____

2. (1 min) When does the game end?

3. (2 min) Play the game a few times to ensure that everyone understands the rules.

4. (2 min) Optional: List up to 3 ways to clarify the rules.

start time:

II. (12 min) Player Strategies

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 – in computer science, we call these **algorithms**.
For example, one strategy is "Count upwards, starting from 1."
Name each strategy and list it in the first column of the **Hi-Lo Game: Worksheet**.
Before you continue, **review progress with the facilitator**.

2. (2 min) Evaluate each algorithm (strategy) with regard to how **quickly** it will find the right answer, by ranking from 1 (fewest guesses) to 5 (most guesses).
Add the rankings to the worksheet in a column labeled **Quick**.

3. (2 min) Evaluate each algorithm with regard to how **easy** it is to describe or specify, by ranking from 1 (easiest) to 5 (hardest).
(Suppose you had to explain each algorithm to a first-grader so that she could play the game.)
Add the ranking to the worksheet in a column labeled **Easy**.

4. (1 min) For each algorithm, plot its Quick and Easy values on the **Hi Lo Game: Graph**.

5. (3 min) In **complete sentences**, describe the relationships between the Quick and Easy rankings, including what you see from the graph. Before you continue, **review progress with the facilitator**.

III. (10 min) Worst & Average Case Performance

start time:

1. (3 min) We could compare the algorithms' speeds using the **number of guesses**.
For each algorithm, determine the **worst case** (maximum) number of guesses required to win.
Add the numbers to the worksheet in a column labeled **Worst**.

2. (3 min) For each algorithm, determine the **average case** (typical) number of guesses required to win.
Add the numbers to the worksheet in a column labeled **Average**.
Note that the **minimum** number of guesses is always 1 – it's nice to be lucky.
Hint: there is a pattern between worst and average cases, but it **does not** apply for all strategies.

3. (2 min) List 3 reasons why the number of guesses could be a better way to compare algorithms.
Before you continue, **review progress with the facilitator**.

IV. (10 min) Effect of Input Size

start time:

1. (3 min) Assume that Player A chooses a number from 1 to 1000
(instead of 1 to 100). For each algorithm, what are the **worst case** & **average case** number of guesses?
Add the numbers to the worksheet in columns labeled "1K Worst" and "1K Average".

2. (4 min) **Optional**: Assume that Player A chooses a number from 1 to N.
(For example, N=100, N=1000, N=1,000,000)
For each algorithm, what are the **worst case** & **average case** number of guesses in terms of N?
Add the expressions to the worksheet in columns labeled "N Worst" and "N Average".
(Hint: you've already done N=100 and N=1000; consider other values before generalizing to N.)

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

Applications

1. Team Meeting Minutes (TMM) – see common description (separate doc) and rubric (below).

2. Personal Reflection Memo (PRM) – see common description (separate doc).

3. Write a program (or set of programs) to play the game using each algorithm.
The program should count the guesses and print the total when the game ends.
Record how much time it takes to code each algorithm. Is this consistent with your "Easy" rating?
Play the game 5 times with each algorithm. Is the guess count consistent with your "Quick" rating?

Team Meeting Minutes (TMM): Rubric

SPECIFIC CRITERIA	RATING	COMMENTS
I: Comprehensively synthesizes guessing algorithms.	/ 4	
II: Comprehensively synthesizes rankings & tradeoffs.	/ 4	
III: Comprehensively synthesizes worst & average case performance.	/ 3	
IV: Comprehensively synthesizes N-analysis.	/ 3	
COMMON CRITERIA	**RATING**	**COMMENTS**
Begins with a summary of the activity; ends with a summary of questions, and a list of action items, if needed.	/ 3	
Mechanics & format: Work is neat, well organized, & clearly provides all required information, including team member roles & timing data.	/ 3	
TOTAL	/ 20	

Facilitator – Answer Key

I. (6 min) Hi-Lo Game

1. There are 3 possible answers – too high, too low, you win.
2. The game ends when B wins or gives up.

II. (12 min) Player Strategies

1-4. See worksheet answer key below.
5. In general, simple algorithms tend to be slower, and faster algorithms tend to be more complex.

III. (10 min) Worst & Average Case Performance

1. Measuring time with a stopwatch is objective & quantifiable, but may be affected by other factors – math aptitude of the person playing the game, reaction time of the timer.
2-3. See worksheet key below.
4. Number of guesses is a better measure because it depends less on other factors.

IV. (10 min) Effect of Input Size

1-2. See worksheet key below.
3. Pros & cons of analyzing performance in terms of input size N.
   Pros: generalizes from specific examples
   Cons: more difficult calculations & logic.

If teams have questions about random guessing, have each student write down 1-2 random numbers, then list or plot for entire class. Typical outcomes: no multiples of 10 or 5, more odds than evens

90										X
80				X						
70			X	X	X					
60					X					
50			X					X		
40			XX	X		X		X		
30							X			
20			X					X		
10		X	X		XX			X		X
0				XX				XX		XX
	0	1	2	3	4	5	6	7	8	9

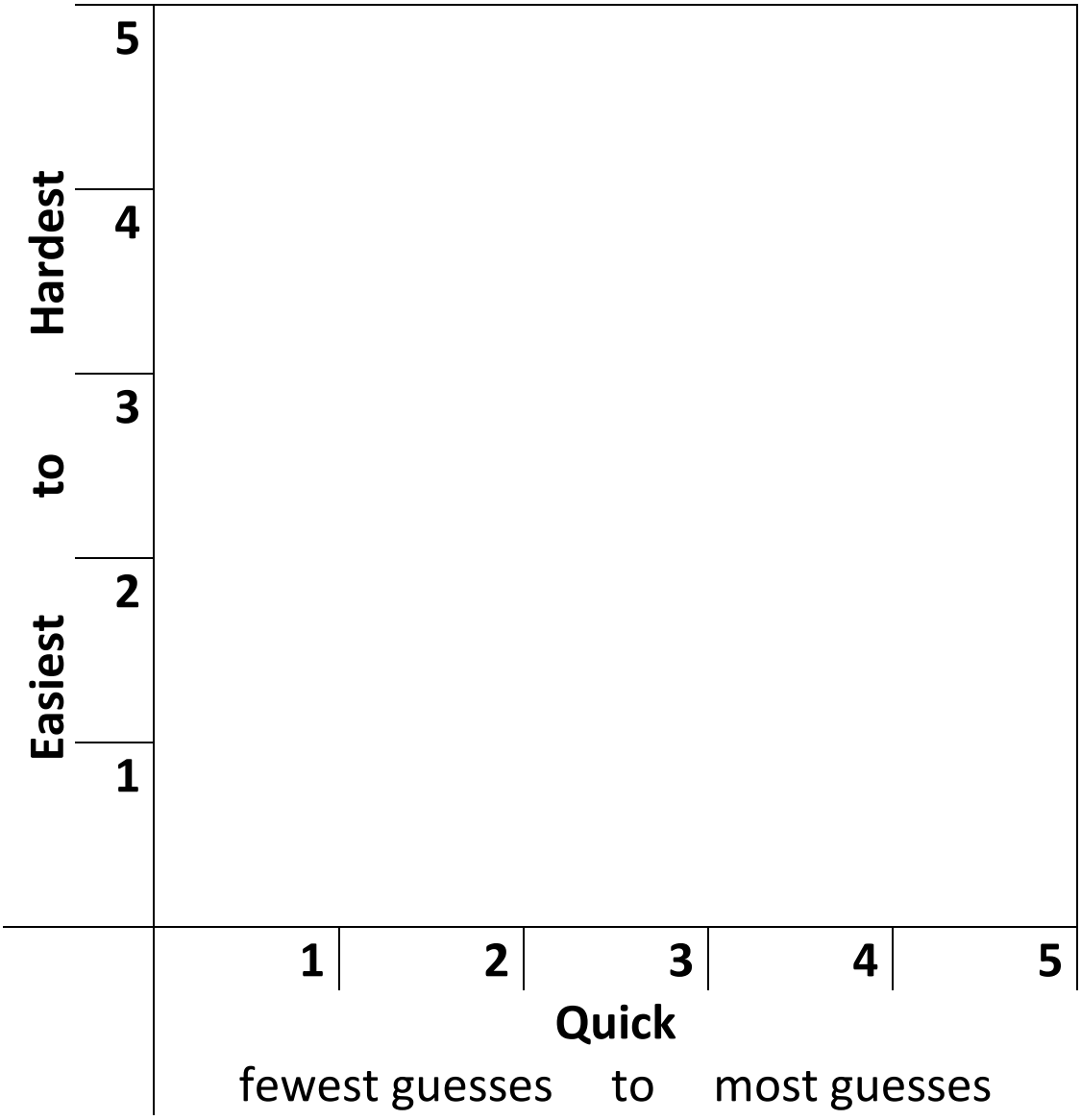
Hi-Lo Game Worksheet – Answer Key

#	Player Algorithm	II.1. Quick	II.2. Easy	II.3. Prod	III.2. Worst	III.3. Avg	IV.1. iK Worst	IV.1. 1K Aver	IV.3. N Worst	IV.3. N Aver
A	Guess numbers at random, ignoring "hi-lo" feedback, without memory (repeat guesses possible)	5	2	10	1K	50	10K	500	10N	N/2
B	Guess numbers at random, ignoring "hi-lo" feedback, but don't repeat guesses	4	3	12	100	50	1000	500	N	N/2
C	Count up from 1 (or down from 100)	3	1	3	100	50	1000	500	N	N/2
D	Count up by 10s, then down by 1s (many variations)	2	4	8	20	10	110 (30)	55 (15)		
E	Guess 50, then 25 or 75, then 12 or 38 or 62 or 88 – keep track of min & max possible values, divide range in half	1	5	5	7	6	11	10	$\log_2 N$	$\log_2 N$

Hi-Lo Game Worksheet

#	A	B	C	D	E	F
Algorithm Name	Count up from 1 until answer found					
Quick						
Easy						
Worst						
Average						
1K Worst						
1K Average						
N Worst						
N Average						

Hi-Lo Game: Graph



Random Number Distribution

90										
80										
70										
60										
50										
40										
30										
20										
10										
0										
	0	1	2	3	4	5	6	7	8	9