WAKE FOREST UNIVERSITY

Departments *of* Computer Science and Physics

Google education

Wake @ Hanes

Google CS4HS Teacher Workshop

August 9-10, 2012

CS4 HS

Higher/Lower Guessing Game: Winning via Efficient Searching

Richard (Rick) G. Freedman

Wake Forest University/
University of Massachusetts Amherst

August 9, 2012

Higher/Lower Guessing Game

Rules of Game:

- One player chooses a *number* between *minimum* and *maximum*.
- Other players try to guess the *number*. After each guess, they are told whether their guess is higher or lower than that *number*.
- The player who correctly guesses the *number* wins. If no one is correct after a specified number of guesses, the player who chose the *number* wins.

Teenage Mutant Ninja Turtles Fall of the Footclan™. Developed by Ultra Games, subsidiary of Konami. Copyright 1991. Rights belong to Ubisoft at present (2012).

Strategies:

- Use the higher/lower hints to change guessed number.
- Eliminate as many numbers as possible with a single guess and hint.

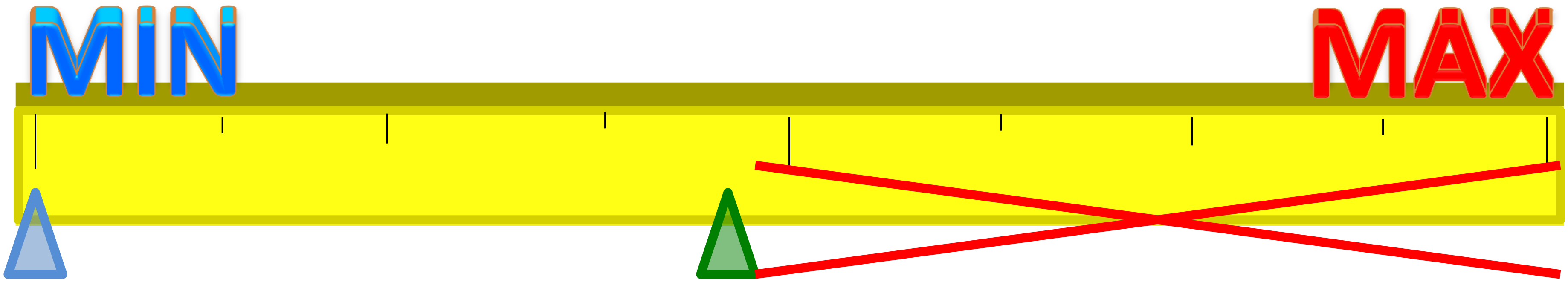
Searching for the Best Guess

Difficulty comes from number of guesses allowed.

- *(MAX – MIN* + 1*)* guesses is easiest. We can guess every number from *MIN* to *MAX* and always win. This is called a ***linear search***.
- What if we have fewer guesses? Can we still win every time? What are the fewest guesses that we need for this?

Consider:

- If guess is higher, then all numbers higher than guess are also higher. Same for lower.
- This eliminates some portion of the remaining guesses. What is the largest portion of guesses that we can remove regardless of higher/lower hint?
  - Answer: Half.

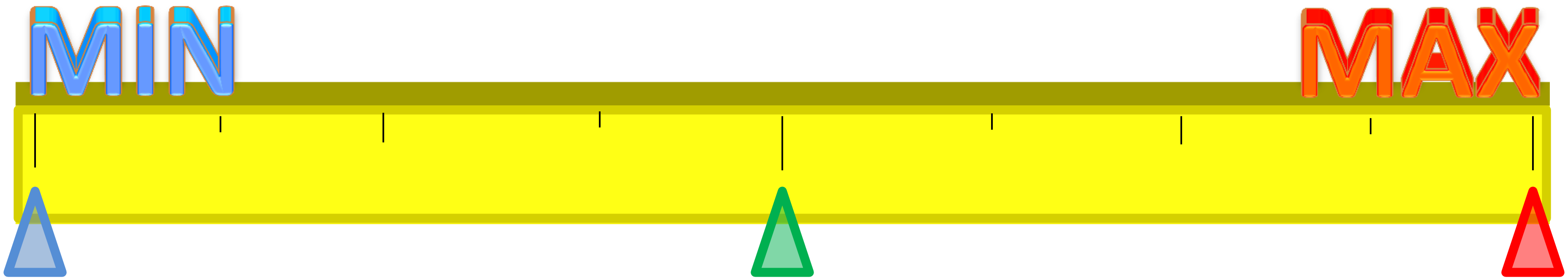


A Formula for Success

Perform ***binary search*** algorithm (by automation):

→ Begin with *MAXGUESS* = *MAX* and *MINGUESS* = *MIN*.

→ Guess the number halfway between *MAXGUESS* and *MINGUESS*.



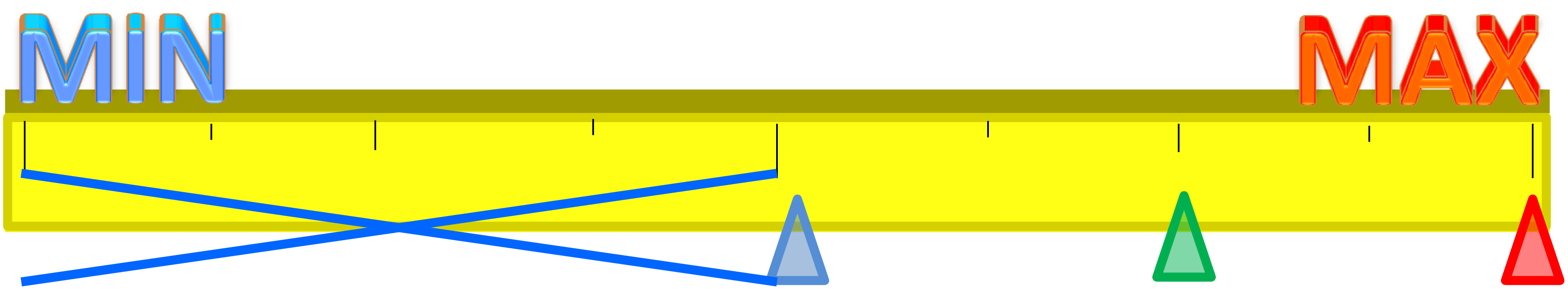
A Formula for Success

Perform ***binary search*** algorithm (by automation):

- Begin with *MAXGUESS* = *MAX* and *MINGUESS* = *MIN*.
- Guess the number halfway between *MAXGUESS* and *MINGUESS*.
  - If the guess is higher, then change *MAXGUESS* to 1 less than the guess itself.
  - If the guess is lower, then change *MINGUESS* to 1 more than the guess itself.
- Repeat the process by now guessing the new number halfway between *MAXGUESS* and *MINGUESS*.

Only half the numbers remain as possible guesses after each try.

- High school math can show that binary search will find the answer in no more than $\lceil \log_2(MAX - MIN + 1) \rceil$ guesses. This number is a lot smaller than ($MAX - MIN + 1$) as the interval gets bigger!



Rematch: Let's Try It Out!

$\lceil \log_2(MAX - MIN + 1) \rceil = \lceil \log_2(999 - 0 + 1) \rceil = \lceil \log_2(1000) \rceil = 10$ … How convenient!

MINGUESS	*MAXGUESS*	*HALFWAY*
0	999	500
0	499	250
251	499	375
251	374	312
251	311	281
251	280	265
251	264	257
251	256	253
254	256	255
254	254	254

Teenage Mutant Ninja Turtles Fall of the Footclan™. Developed by Ultra Games, subsidiary of Konami. Copyright 1991. Rights belong to Ubisoft at present (2012).

Further Computational Thinking

If a group of objects can be ordered, we can use binary search instead of linear search.

- What is the best way to order the objects? Are there some ways that are better for certain situations?
  - Numbers: least to greatest
  - Words: lexicographically
- How do we know if binary search fails (something is not in the group)?

Is there a quicker way to search than binary search?

- What conditions must be satisfied for such a method to work?

How do we put the objects in order to perform binary search?

- What are we sorting?
- Are there duplicate objects in the group?
- What are our memory and/or time limits?