Gautam Ramasubramanian
Rutgers DIMACS Research

Basic Results about Number Guessing Game that comes from Brute Force Search

Problem Statement

You and a partner are playing a game. The partner thinks of a number from 1 to n. You guess at the number repeatedly. When you tell your partner your guess, the partner will tell you whether the guess is too high or too low. We assume the partner is 100% correct when he tell you that. However, when you get a "too high" verdict, you incur a cost α. When you geta "too low" verdict, you incur a cost β. The cost accumulates as you guess again. What is the best strategy for guessing that minimizes your cost.

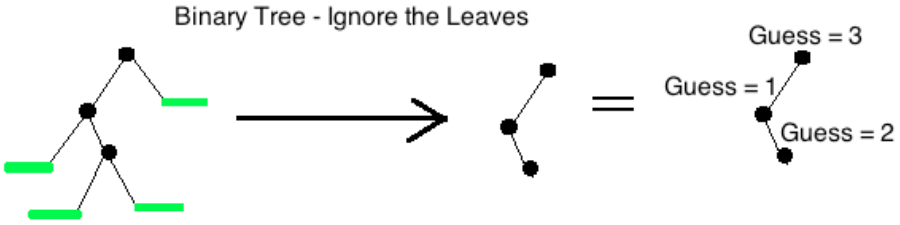
Brute Force

When n is small, we can solve this problem by using brute force. We will then use the data from these small examples to test a mathematical model of this game that can be then applied to cases where n is large.

Trees and Guessing Strategies

Each guessing strategy can be visualized by using a binary tree. This entire section will convince you of that. If you are already convinced, skip this section.

In the binary tree, we ignore the leaf nodes and concentrate only on the internal nodes. Generally, the game is played with numbers from 1 to n, so the tree must have n internal nodes. The root node indicates the first guess, and the left child and right child indicate the next guess if the current guess is too high or too low respectively.



Another way to think about it is to say that there is a bijection between guessing strategy and tree. A guessing strategy can be turned into a binary tree (leaves ignored) by the details in the previous paragraph. A less obvious point is that a binary tree can be turned into a strategy. By the shape of the tree, it is possible to tell what number each node represents, and that translates into the guessing strategy. We can use the following rules.

1. If a node has a left child (a left subtree), then recusively speaking,

$$\text{(Node number)} = \text{(Number of nodes in the left subtree)} + 1$$

2. If a node has no left child (left subtree), then there are two cases

   (a) If a node is a right child of another node, then

$$\text{(Node number)} = \text{(Node number of parent)} + 1$$

   (b) If the node is a left child of another node, or if the node is root

$$\text{(Node number)} = 1$$

This shows that there is a bijection between a binary tree with n internal nodes and a guessing strategy for a number guessing game. We can use trees to visualize the guessing strategy.

Aside: Catalan Number

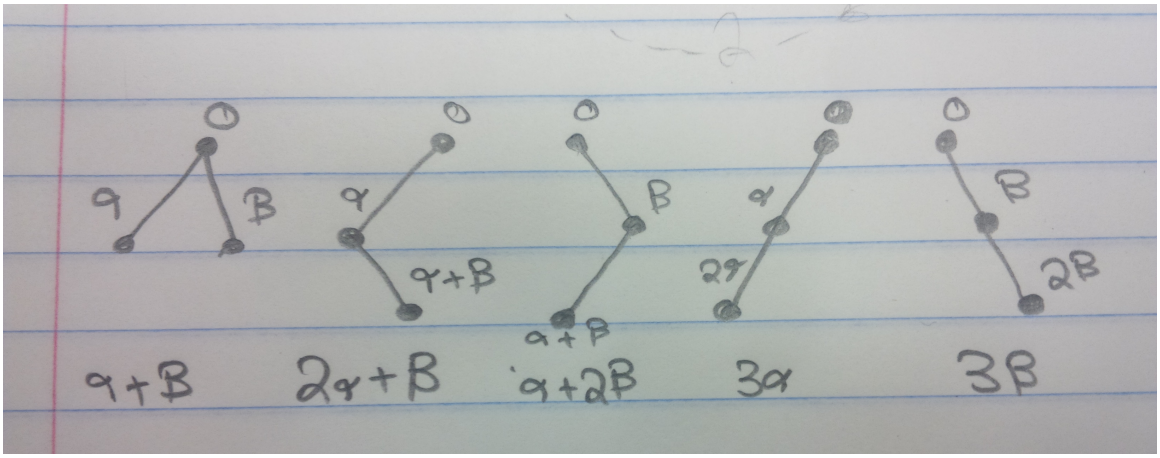
If we are convinced with the bijection between trees and guessing strategy, there is a nice corollary.

Corollary: The number of guessing strategies for the number guessing game with n numbers is C_n, where C_n is the nth Catalan number.

Proof: C_n is just the number of binary trees there are with n internal nodes, so since there is a bijection between trees and guessing strategies, C_n is also the number of guessing strategies for the number guessing game with n numbers.

When $n = 3$

If we consider when $n = 3$, $C_3 = 5$, so there are 5 guessing strategies total. We can enumerate the different strategies, and evaluate the different overall costs for each strategy.



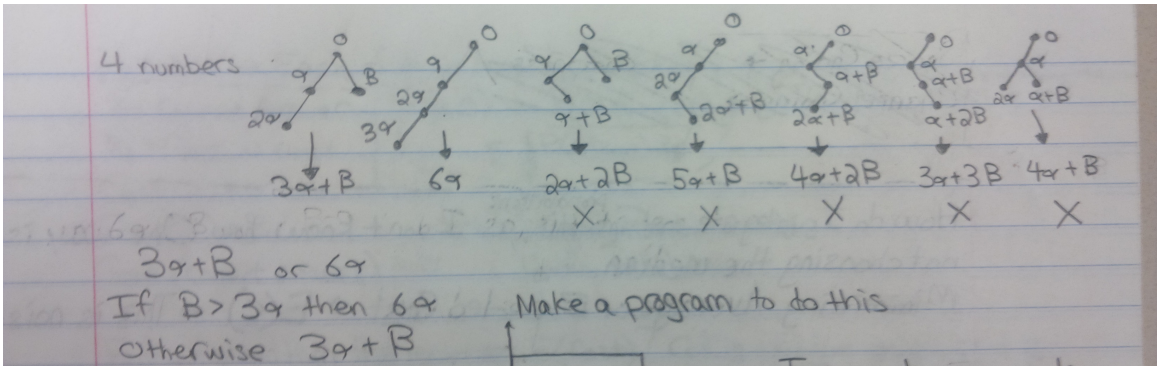
It is easy to see that the best strategy, the one that will give the minimum cost, will depend on the values of α and β. There are three cases.

1. If $2\alpha < \beta$, then the 4th strategy from the left is the best one (minimal cost).
2. If $\alpha > 2\beta$, then the 5th strategy from the left is the best one (minimal cost).
3. Else, the 1st strategy is the best one.

We can see that the best trees for when $\alpha < \beta$ are the mirror image of the best trees when $\alpha > \beta$, and it makes sense for them to be, so we only consider the cases where $\alpha < \beta$ to simplify our brute force search.

When $n = 4$

If we consider when $n = 4$, $C_3 = 14$, so there are 14 guessing strategies total. Since we limit our search to only cases where $\alpha < \beta$, it makes sense to restrict our search to the 7 trees that have more left children than right children. The other 7 will be the mirror images of these 7 trees.



The costs for the other seven trees will be the same except the α and β get swapped. The other trees will have more β terms, so in the case when $\alpha < \beta$, their costs will be more. Therefore, we will ignore them. Most of the strategies are not the best under and circumstance, but there are 2 that have potential.

1. If $\alpha < \beta < 3\alpha$, then the first strategy from the left is the best (minimal cost)
2. If $\beta > 3\alpha$, then the second strategy from the left is the best (minimal cost)

The mirror images of those trees will be the best strategies in the case where $\beta < \alpha$.