# ECOO 2013

## Programming Contest Solutions and Notes, Part 2

Regional Competition (Round 2)

April 27, 2013

**Sam Scott, Sheridan College (sam.scott@sheridancollege.ca)**

# Breaking Rocks: An Efficient Solution

It has already been pointed out that there is a $3^{n-1}$ upper limit on the size of the $n^{th}$ piece. But In fact, there is a tighter upper bound on the size of any piece: 2*T+1, where T is the total of all the previous pieces in sorted order. For example, if you have already generated pieces of size 1 2 and 5, the next piece can be as big as 2*8+1 = 17, but no bigger. The reason is that the previous pieces can handle all quantities from 1 to 8 and you can subtract these quantities from a larger number but 17-8 is 9. If the next piece was 18 or bigger you wouldn't be able to weigh a 9 kg sack of corn.

This leads to a much more efficient check on your splits. Instead of using a backtracker to check all the combinations, just make sure that the first piece is of size 1, that the pieces all add up to R, and that the upper limit of 2*T+1 is respected for every piece. That leads to a solution that finishes very quickly for all test cases.

## Other Solutions?

It seems to me that dynamic programming should be possible when checking splits, though it is unlikely to be as efficient as the approach described above. It also seems to us that there might be a mathematical formula to compute the answer directly given P and R, which would be superefficient. But at the time of writing, the contest team hadn't found either solution. Let us know if you figure it out, by posting to the compsci.ca discussion forum.