

2. We can take a binary search approach to this problem. We weigh  $\frac{1}{2}$  of the coins on one side and the other on the opposite side. Whichever is greater will be the  $\frac{1}{2}$  we want to consider, as this will be the pile of coins that contains the heavy coin. We can then only consider the heavier half. We can continually do this until we either have one coin in each division. This would clearly identify which coin is the heavier coin. However, if there is a case where we need to divide an odd number in half, we can divide the pile into almost half and remove one coin from the larger pile. If by chance, we pick the heavier coin, then the scale will balance and we can terminate our algorithm. On the other hand, if the scale does not balance, we can discard the coin we set aside, as we know that it is not the heavier coin and repeat the process assuming there is an equal number of coins on each side.

Run time: For simplicity, assume that the initial pile contains  $2^n$  coins. At each step, we are dividing the size of the comparisons by 2. In the worst case, we would find the coin at the last comparison. For example, with a pile of 16 coins, in the worst case, we compare 8 and 8 coins, 4 and 4, 2 and 2, 1 and 1, and then find the answer. This would be 4 comparisons, which would be  $\log_2(16) = 4$ . More generally, we can find the solution in less than or equal to  $\log_2(2^n) \rightarrow \log(n)$  comparisons. This is  $O(\log(n))$  minutes.

Cost: Assume once more 16 coins. At each step, the cost is  $8+8=16$ ,  $4+4=8$ ,  $2+2=4$ , and  $1+1=2$ .

Generalizing in terms of  $n$ , in total, this would be  $1n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n$ . Summing we get

$n(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} \dots)$ . We can estimate the worst-case sum as  $n(\frac{1}{1-\frac{1}{2}}) = 2n = O(n)$ .