University of Washington February 18, 2020
Department of Computer Science and Engineering
CSE 417, Winter 2020

Homework 6 problem 1

Problem 1 (10 points):
Suppose you are working in the quality control of a factory that produces quarters for the US government and your job is to make sure that all quarters have exactly the same weight. You are given 2^k quarters for $k \geq 2$ and you know that at most one of them can be defective. A defective quarter will weight higher or lower than normal. You are given a scale with two trays: Each time you can put a set S of quarters in the left and a set T in the right (for disjoints sets S, T). The scale will show if S is heavier than T, or T is heavier than S, or they have exactly the same weight. Design an algorithm to find the defective quarter (if it exists) by using the scale only $O(k)$ many times. (Note that your algorithm will run by a human not a computer.) Justify your algorithm is correct.

Answer:
Algorithm:

Random shuffle all quarters.
Divide 2^k quarters into two 2^{k-1} quarters, put on two trays of scale each sides, for each 2^{k-1} quarters, divide the BOTH the heavyweight and lightweight side again, and weight each 2^{k-2} pair on two trays of scale, ignore the pair which the scale have balance weight. And note down the kept half is at begining heavier or lighter. If both side are balance, the algorithm halt.

Divide the rest part of quarters in to two 2^{k-2} quarters, weight each, and ignore the 2^{k-2} quarters with more / less total weight according to the note (if note is heavier, keep heavier half, visa versa).
......
Divide the rest part of quarters with size 2^{k-i+1} in to two 2^{k-i} quarters, weight each, and ignore the 2^{k-i} quarters with more / less total weight according to the note.
......
Divide the rest part of quarters with size 2 in to two size 1 quarters, weight each, and the one quarter with more / less total weight according to the note is the defective quarter.

Proof:
correctness:
It garantees to find the defective quarter when there is at most one among all. The total amount is 2^k, which means it is bipartisable. And the first recursion leave out the half with equal weight sub-half (assume it exist defective one), so the rest must contain one defective in one half. Every recursion, the algorithm divide the problem by half, and the half with defective quarters will be revealed by comparing until only two left.
complexity:
It garantees to halt with k recursion and plus 2 extra comparisons of first and second rounds with 2^{k-1} and 2^{k-2} size. Since $T(n) = c \cdot T(\frac{n}{2})$, hence $T(n) = c \cdot \log_2 2^k + 2 = c \cdot k + 2 = O(k)$