Ph/CS 219a, Fall 2022
Quantum Computation
Prof. Alexei Kitaev

Problem set #5
Due Thursday, November 3, 2022

1. The $O(\cdots)$ notation. [5 points] We have used this notation in class: $f(n) \leq O(g(n))$ if there are some constants c and n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$. The O notation allows one to compare the asymptotic growth of different functions: $f \leq O(g)$ means that f grows more slowly, or at least not faster than g. Note that two function can grow at the same rate, e.g., $f(n) = 2n$ and $g(n) = n + 100$. In some cases neither $f \leq O(g)$ nor $g \leq O(f)$, e.g., $f(n) = n$ and $g(n) = n^2 \sin^2(\log n)$ (of course, this example is rather contrived).

Sort the following functions in order of their asymptotic growth (i.e., "$\lesssim$" stands for the O-relation and "$\sim$" stands for the bi-directional O-relation):

$$n, \quad n \log n, \quad (\log_2 n)^n, \quad (\ln n)^n, \quad , n^{\ln n}, \quad 2^n, \quad n!, \quad \binom{2n}{n}, \quad \ln(n!), \quad \ln \binom{2n}{n}.$$

You may use the Stirling approximation: $n! \approx \sqrt{2\pi n}(n/e)^n$.

2. The threshold function. [15 points] To decrease the chance of error in a probabilistic computation, one generally needs to run the algorithm several times and choose the result that occurs most frequently. For simplicity, let us assume that the output of each run, x_j is a single bit. Then the final result is given by the majority function $\mathrm{MAJ}_n(x_1, \ldots, x_n)$, which is equal to 1 if and only if more than a half of its arguments are 1. Now, let us define a more general function:

$$\mathrm{THRESHOLD}_{n,k}(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum_{j=1}^n x_j < k, \\ 1 & \text{if } \sum_{j=1}^n x_j \geq k. \end{cases} \tag{1}$$

The problem is to implement it by a Boolean circuit.

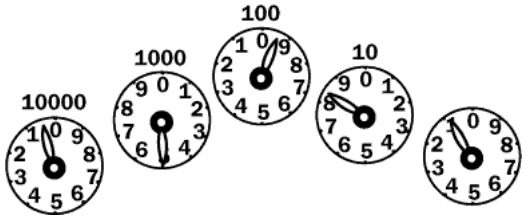
a) We first try the simplest, though not the most efficient method. Let us compute $\sum_{j=1}^n x_j$ using this encoding: a number $y \in \{0, \ldots, n\}$ is represented as $\underbrace{1 \ldots 1}_{y} \underbrace{0 \ldots 0}_{n-y}$. Then the threshold function is just the k-th bit of the sum. Describe and/or draw the circuit corresponding to this calculation; determine its size and depth.

b) A better strategy is this: we compute $\sum_{j=1}^n x_j - k$ in binary and compare the result with 0. (Negative numbers are represented using the two's complement, e.g., $-2 = \overline{1 \ldots 110}$.) Note that the addition of $-k$ and the comparison are relatively cheap; the main work is to compute the sum $\sum_{j=1}^n x_j$. Estimate the size of the resulting circuit in two cases: (i) when the numbers $x_1, \ldots, x_n$ are added sequentially and (ii) when the addition is organized in a binary tree.

c) In the last case, apply the parallelization tricks discussed in class (namely, the logarithmic depth adder and the parallel 3-2 adder) and estimate the total depth. You are not required to recalculate the size, but please make sure that it only changes by a constant factor.

3. Measurement decoding. [Extra credit 15 points] (Strange as it sounds, this problem has important applications in quantum computation. We will use the algorithm in class!)

Some old electricity meters look like this:



Apparently, it takes a special skill to read them quickly. Let us consider a mathematical version of the meter reading problem. We will assume that the dials have eight digits instead of ten and rotate at speeds that differ by powers of 2. The meter may roll over the highest value, which is equal to 1.

Let x be a real number *modulo* 1, i.e., we do not distinguish between numbers that differ by an integer. We may also think of x as a point on the circle that is obtained from the interval $[0, 1]$ by identifying the points 0 and 1. However, the actual value of x is unknown. Instead, we are given individual dial readings $s_0, \ldots, s_{n-1} \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ such that

$$2^j x \in \Big(\frac{s_j - 1}{8},\, \frac{s_j + 1}{8}\Big) \pmod 1 \qquad \text{for } j \in \{0, \ldots, n-1\}, \tag{2}$$

where the notation $(a, b) \pmod 1$ stands for the interval from a to b on the circle. Assuming that the data are consistent, we need to find a number $y = \overline{.y_1 \ldots y_{n+2}} = \sum_{j=1}^{n+2} 2^{-j} y_j$ (where $y_j \in \{0, 1\}$) such that

$$x - y \in \big(-2^{-(n+2)},\, 2^{-(n+2)}\big) \pmod 1. \tag{3}$$

a) Construct a circuit of size $O(n)$ for the solution of this problem. (Describe the circuit in reasonable detail, e.g., as a composition of blocks operating with a constant number of bits.)

b) Parallelize the computation by analogy with the parallel addition algorithm discussed in class. Specifically, construct a circuit of size $O(n)$ and depth $O(\log n)$. You don't have to draw the whole circuit. Just describe intermediate data (like the 0-1-carry values in the addition algorithm), elementary blocks, and the way they are grouped in levels.