E0234: Solution Sketch of Assignment 1

January 22, 2016

1. Suppose you have access to a subroutine `randbit()` which returns 0 or 1 with probability 1/2. Use this to design `randint(n)`, which takes input an integer n and returns an integer in the range $\{1, \ldots, n\}$ uniformly at random. **Hint:** First do this when n is a power of 2. How many calls in expectation to `randbit()` is made for input n?

   Solution sketch: First, let us assume that the integer n is a power of two. We call the subroutine randbit() $\log_2 n$ times and return the integer $1 + (b_{\log_2 n} \ldots b_1)_2$ as the output of randint(), where b_i is the bit returned by the i^{th} call to randbit() and $(b_{\log_2 n} \ldots b_1)_2$ is the integer whose binary representation is $b_{\log_2 n} \ldots b_1$. It is clear that randint() returns an integer in $\{1, \ldots, n\}$ uniformly at random. Now, for general n, we call the subroutine randbit() $\lceil \log_2 n \rceil$ times and return the integer $1 + (b_{\log_2 n} \ldots b_1)_2$ as the output of randint() conditioned on the fact that $1 + (b_{\log_2 n} \ldots b_1)_2 \in \{1, \ldots, n\}$; in particular, if $1 + (b_{\log_2 n} \ldots b_1)_2 \notin \{1, \ldots, n\}$, then we repeat the process again. It follows from straight forward application of Bernoulli random variable that the expected number of calls to randbit() is $\frac{2^{\lceil \log_2 n \rceil}}{n} \lceil \log_2 n \rceil$.

2. **Implement** the above algorithm in your favourite language – find out what is the equivalent of `randbit()` in it. Run your code with $n = 8$ a million times storing your answer in an array a. Lets call a pair of indices (i, j) a *streak* if the entries of a in this range are equal. Let $|j - i + 1|$ be the length of this streak. Write down the length of the longest streak in your array a.

   Solution sketch: Enjoy ☺

3. In the QuickSort algorithm done in class, let us use π to denote the order in which the pivots are chosen. That is, $\pi(1)$ is the value of the first pivot, $\pi(2)$ is the value of the second pivot, and so on. Since every number is chosen as a pivot at some time and exactly once, π will be a random permutation of the array a. Is this distribution uniform among all permutations of the array? Give a mathematical and rigorous explanation.

   Solution sketch: The distribution is not uniform. Let A be the input array of size n with maximum and minimum elements being x and y respectively. Clearly, $\Pr[\pi(1) = a] = \frac{1}{n}$. However, $\Pr[\pi(2) = x] = 0$ and $\Pr[\pi(2) = y] > 0$. Hence, the distribution is not uniform.

4. **Implement** Karger's algorithm in your favourite language. Run it on the file provided in the website. The file is the adjacency matrix of an undirected graph. Each line is a row of the matrix and different rows are separated by new lines. What is the minimum cut size? How many iterations of the subroutine did you need to detect this?

   Solution sketch: Enjoy ☺