CS5330: Week 1 Solutions

February 5, 2020

Here are solution sketches to the Week 1 problems. If anything is unclear, please talk to me or your TA.

1. Every evening, a man either visits his parents, who live northwards, or his friend, who lives southwards (but not both). In order to be fair, he goes to the bus stop every evening at a random time and takes either the northward or southward bus, whichever comes first. The two kinds of buses stop at the bus stop every 30 minutes with perfect regularity. Yet, he visits his parents only three times per month. Why?

   Solution sketch: The bus to the parents could arrive a very short time after the bus to the friend. So, if the man arrives at a random time, it's not very likely that he reaches the bus stop during the short interval in which the next bus is the one to his parents.

2. (a) 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.

   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 $1 + (b_{\lceil \log_2 n \rceil} \ldots b_1)_2$ as the output of randint() conditioned on the fact that $1 + (b_{\lceil \log_2 n \rceil} \ldots b_1)_2 \in \{1, \ldots, n\}$; in particular, if $1 + (b_{\lceil log_2 n \rceil} \ldots b_1)_2 \notin \{1, \ldots, n\}$, then we repeat the process again. It is not hard to show that the expected number of calls to randbit() is $\frac{2^{\lceil \log_2 n \rceil}}{n} \lceil \log_2 n \rceil$.

(b) **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.

Hope that was fun ☺. Exercise: What is the expected length of a streak in terms of n?

3. **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?

   Hope that was fun ☺. For the provided graph, the min cut size was 3.