Programming Contest (Part II)

Data Structures Lab (CS 210)

2012

Q1. **(10 marks)** The robot has devised a new algorithm in the meantime to solve the space-complexity problem of it's earlier algorithm. When it's instructed to sort n distinct integers from $\{1, 2, \ldots, n\}$ in an ascending order, it now devises the following modified algorithm. It first stores the given elements in an array of size n (array indices start at 0 and end at $n-1$) and then applies the following algorithm.

Pick out a pair of positions i and j ($0 \leq i, j \leq n-1, i \neq j$) uniformly at random. Swap the elements stored in these positions, and check whether the current permutation is sorted in an ascending order. If it is not, then repeat the steps above. Otherwise, stop with the sorted output.

Your job is to implement the new algorithm used by the robot in an in-place manner using C/C++. You are allowed to store the input elements in an array of n integers. Only a constant number of extra variables can be used in addition to this array. Assume (even though it's only an approximately correct assumption) that the pseudo-random number generator *rand*() function in your program can generate a sequence of uniformly-random integers in any given range of integers, if you can use it properly. It's your responsibility to use the *rand*() function properly.

Input: A sequence of n distinct integers in the range $[1, n]$, manually typed on the terminal, separated by space and terminated by "ENTER". The value of n would be at most 10000, and it needs to be determined after "ENTER" is pressed.

Output: The integers in an increasing order, separated by space on the terminal.

[Termination of your code need not be guaranteed, but the algorithm must be correct within the constraints given in the problem statement.]

Q2. **(10 marks)** You have already written the C/C++ code of an $O(n^{\frac{1}{2}} \log n)$-time algorithm to identify all prime numbers of the form $2^k + 1$, which are less than or equal to a large number n. [We assumed that k can only be a positive integer and 1 is not a prime number.] Store these numbers, one in each line, in a temporary file temp.txt, and use this file to identify all prime numbers of the form $2^{2^k} + 1$. If there are p numbers in temp.txt, this algorithm (to find out all primes of the form $2^{2^k} + 1$) can take $O(pn^{\frac{1}{2}})$ time after the file is read and the numbers are stored in an array of size p.

Input: An integer n typed on the terminal followed by "ENTER".

Output: All prime numbers of the form $2^{2^k} + 1$, which are less than or equal to n, in an increasing order. The numbers should be separated by space on the terminal.

[5 marks would be awarded if your program works for any $n \leq 10^4$, and full 10 marks would be awarded if it works for any $n \leq 10^8$ within the constraints given the problem statement. You are not allowed to use any prime number table that directly gives such primes. Your algorithm should be able to identify the primes without any prior knowledge about the sequence of prime numbers.]

Q3. **(10 marks)** Write the C/C++ code of an $O(\log n)$-time algorithm to calculate 2^n for any given integer n.

Input: An integer n typed on the terminal followed by "ENTER".

Output: The integer 2^n.

[5 marks would be awarded if your program works for any $n \leq 14$, and full 10 marks would be awarded if it works for any $n \leq 100$ within the constraints given the problem statement.]