

## Problem Set 5 for CS 170

### Note

When asked for an algorithm you must give (1) a brief informal description of the algorithm, (2) a precise description using pseudo-code, (3) an argument for termination and correctness of the algorithm, and (4) an analysis of the running time of the algorithm. Be clear about what the input to the algorithm is, how you measure the size of the input, and what constitutes a “step” in your running-time analysis.

### Problem 1. [Probabilities and random variables] (4 points)

One of the clubs in town just admitted a new member, bringing its total number of members to 1000. To celebrate the occasion, a committee decided to make a raffle on the upcoming Sunday. They prepared 1000 tickets, each with a number from 00 to 99, ten tickets with each number. On Sunday, they will randomly distribute these tickets—one to each member of the club. Any person who gets a ticket with a number that matches the last two digits of the year of his/her birth will win a prize.

- (a) What is the probability that the new member will get a prize?
- (b) What is the expected number of prizes the committee will have to give out?

### Problem 2. [A randomized algorithm] (5 points)

Let  $A[1..n]$  be an array of  $n$  pairwise distinct numbers. Let each call of  $RandomNum(n)$  return a number that is chosen uniformly at random from the set  $\{1, \dots, n\}$  of integers. Consider the following code:

```
for  $i = 1$  to  $n$  do swap  $A[i]$  and  $A[RandomNum(n)]$ 
```

Does this code, for every  $n$ , produce each permutation of the array  $A$  with equal probability? Why or why not? (*Hint*: Try to calculate the probability that the code gives a particular result. What happens if  $n$  is odd?)

### Problem 3. [2-universal hashing] (6 points)

Consider the functions that map a universe  $U$  of size  $M$  (for example,  $U = \{1, \dots, M\}$ ) to the range  $R_1 = \{0, 1\}$ .

- (a) Let  $M = 4$ . Is there a 2-universal set  $H$  of hash functions from  $U$  to  $R_1$  with  $|H| = 5$ ? Give an example of  $H$  if it exists, or prove that it doesn't exist.
- (b) Same question for  $M = 5$ .

**Problem 4. [2-universal vs. generalized 2-universal hashing]** (10 points)

Let  $H$  be a set of hash functions such that each function  $h \in H$  maps the universe  $U$  of keys to the range  $R_m = \{0, 1, \dots, m-1\}$ . We say that  $H$  is *generalized  $k$ -universal* if for every fixed sequence of  $k$  distinct keys  $(y_1, y_2, \dots, y_k)$  and for every function  $h$  chosen uniformly at random from  $H$ , the sequence  $(h(y_1), h(y_2), \dots, h(y_k))$  is equally likely to be any of the  $m^k$  sequences of length  $k$  with elements drawn from  $R_m$ .

- (a) Show that if the set  $H$  is generalized 2-universal, then it is also 2-universal (according to the definition in the notes).
- (b) Suppose that the universe  $U$  is the set of  $n$ -tuples of numbers drawn from  $Z_p = \{0, 1, \dots, p-1\}$ , where  $p$  is prime. Assume that  $m = p$ . Consider an element  $x = (x_0, \dots, x_{n-1}) \in U$ . For any  $n$ -tuple  $a = (a_0, \dots, a_{n-1}) \in U$ , define the hash function  $h_a$  by

$$h_a(x) = \left( \sum_{j=0}^{n-1} a_j \cdot x_j \right) \bmod p.$$

Let  $H = \{h_a \mid a \in U\}$ . Show that  $H$  is 2-universal, but not generalized 2-universal. (*Hint*: Find a key for which all functions in  $H$  give the same result.)

**Problem 5. [Curriculum revisited]** (5 points)

Recall the following problem from Homework 3:

Suppose a CS curriculum consists of  $n$  courses, all of them mandatory. The prerequisite graph  $G$  has an edge from course  $v$  to course  $w$  if  $v$  is a prerequisite for  $w$ . The number  $k$  represents the maximal number of courses a student can take each semester. Find an algorithm that, given  $G$  and  $k$ , computes the minimal number of semesters necessary to complete the curriculum, and a possible schedule that accomplishes this.

The prerequisite graph must be a dag. Somebody suggests the following algorithm. Assign to each course  $v$  a weight  $f(v)$ , which is the length of the longest path starting from  $v$ . Then find the courses that can be taken in the first semester (i.e., the courses without prerequisites). If there are  $k$  or fewer of those, choose all of them to be scheduled in the first semester; if there are more than  $k$ , choose among them the  $k$  courses with the largest weights (resolving ties in some unspecified way). Remove the courses that are scheduled in the first semester from the prerequisite graph, and repeat the same process to find the courses to be scheduled in the second semester, etc.

Construct an example which shows that this algorithm does *not* always find the minimal number of semesters, no matter how ties are resolved.