Write your answers in the separate answer booklet.

Please return this question sheet and your cheat sheet with your answers.

Red text reflects corrections or clarifications given during the actual exam.

---

1. Suppose we insert n distinct items into an initially empty hash table of size $m \gg n$, using an ***ideal random*** hash function h. Recall that a collision is a set of two distinct items $\{x, y\}$ in the table such that $h(x) = h(y)$.

   (a) What is the exact expected number of collisions?

   (b) Estimate the probability that there are no collisions. *[Hint: Use Markov's inequality.]*

   (c) Estimate the largest value of n such that the probability of having no collisions is at least $1 - 1/n$. Your answer should have the form $n = O(f(m))$ for some simple function f.

   (d) Fix an integer $k > 1$. A ***k-way collision*** is a set of k distinct items $\{x_1, \dots, x_k\}$ that all have the same hash value: $h(x_1) = h(x_2) = \cdots = h(x_k)$. Estimate the largest value of n such that the probability of having no k-way collisions is at least $1 - 1/n$. Your answer should have the form $n = O(f(m, k))$ for some simple function f. *[Hint: You may want to repeat parts (a) and (b).]*

2. Quentin, Alice, and the other Brakebills Physical Kids are planning an excursion through the Neitherlands to Fillory. The Neitherlands is a vast, deserted city composed of several plazas, each containing a single fountain that can magically transport people to a different world. Adjacent plazas are connected by gates, which have been cursed by the Beast. The gates open for only five minutes every hour, all at the same time. During those five minutes, if more than one person passes through any single gate, the Beast will detect their presence.[1] However, people can safely pass through *different* gates at the same time. Moreover, anyone attempting to pass through more than one gate in the same five-minute period will turn into a niffin.[2]

   You are given a map of the Neitherlands, which is a graph G with a vertex for each fountain and an edge for each gate, with the fountains to Earth and Fillory clearly marked; you are also given a positive integer h. Describe and analyze an algorithm to compute the maximum number of people that can walk from the Earth fountain to the Fillory fountain in h hours, without anyone alerting the Beast or turning into a niffin.

---

[1] This is very bad.

[2] This is very bad.

3. Recall that a ***Bloom filter*** is an array $B[1\,..\,m]$ of bits, together with a collection of k independent ideal random hash functions $h_1, h_2, \ldots, h_k$. To insert an item x into a Bloom filter, we set $B[h_i(x)] \leftarrow 1$ for every index i. To test whether an item x belongs to a set represented by a Bloom filter, we check whether $B[h_i(x)] = 1$ for every index i. This algorithm always returns TRUE if x is in the set, but may return either TRUE or FALSE when x is not in the set. Thus, there may be false positives, but no false negatives.

   If there are n distinct items stored in the Bloom filter, then the probability of a false positive is $(1-p)^k$, where $p \approx e^{-kn/m}$ is the probability that $B[j] = 0$ for any particular index j. In particular, if we set $k = (m/n)\ln 2$, then $p = 1/2$, and the probability of a false positive is $(1/2)^{(m/n)\ln 2} \approx (0.61850)^{m/n}$.

   After months spent lovingly crafting a Bloom filter of size m for a set S of n items, using exactly $k = (m/n)\ln 2$ hash functions (so $p = 1/2$), your boss tells you that you must reduce the size of your Bloom filter from m bits down to $m/2$ bits. Unfortunately, you no longer have the original set S, and your company's product ships tomorrow; you have to do something quick and dirty. Fortunately, your boss has a couple of ideas.

   (a) First your boss suggests simply discarding half of the Bloom filter, keeping only the subarray $B[1\,..\,m/2]$. Describe an algorithm to check whether a given item x is an element of the original set S, using only this smaller Bloom filter. As usual, if $x \in S$, your algorithm ***must*** return TRUE.

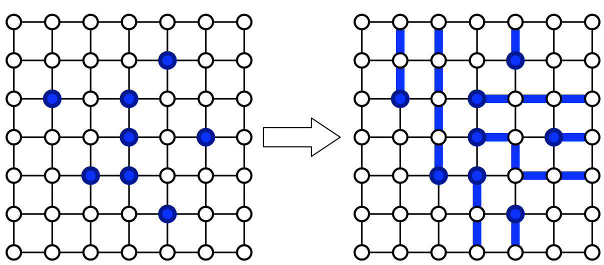
   (b) What is the probability that your algorithm returns TRUE when $x \notin S$?

   (c) Next your boss suggests merging the two halves of your old Bloom filter, defining a new array $B'[1\,..\,m/2]$ by setting $B'[i] \leftarrow B[i] \vee B[i+m/2]$ for all i. Describe an algorithm to check whether a given item x is an element of the original set S, using only this smaller Bloom filter B'. As usual, if $x \in S$, your algorithm ***must*** return TRUE.

   (d) What is the probability that your algorithm returns TRUE when $x \notin S$?

4. An $n \times n$ grid is an undirected graph with n^2 vertices organized into n rows and n columns. We denote the vertex in the ith row and the jth column by (i, j). Every vertex (i, j) has exactly four neighbors $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$, except the *boundary* vertices, for which $i = 1$, $i = n$, $j = 1$, or $j = n$.

   Let $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ be distinct vertices, called *terminals*, in the $n \times n$ grid. The ***escape problem*** is to determine whether there are m vertex-disjoint paths in the grid that connect these terminals to any m distinct boundary vertices. Describe and analyze an efficient algorithm to solve the escape problem.



A positive instance of the escape problem, and its solution.