1 How many length-25 character strings are there?

1. More than the number of hairs on your head.
2. More than the number of Web pages in existence
3. More than the total amount of storage available on Earth
4. More than the number of atoms in the universe

(3) is correct. There are 26^{25} 25-letter strings, this quantity has magnitude roughly 10^{35}. The number of hairs on the human has is about 10^5, the number of webpages is 10^{12} and the number of bits storage available on earth is 10^{25}. The number of atoms in the universe is about 10^{80}.

2 What's the running time fo an educated implementation of the sorted array-based algorithm for the $2 - SUM$ problem?

1. $\Theta(n)$
2. $\Theta(n \cdot logn)$
3. $\Theta(n^{1.5})$
4. $\Theta(n^2)$

The bottleneck of the algorithm is the sort which is $\Theta(n \cdot logn)$. Once we sorted the array we can compute a candiate for the solution by searching for the $t - x$ elements in the array using binary search where t is the target value and x is an element in the array for all x in the array.

3 Consider n people with random birthdays, with each of the 366 days of the year equally likely. How large does n need to be before there is at least a 50% change that two people have the same birthday?

1. 23
2. 57
3. 184
4. 367

(1) is the correct answer. We compute the complement that is we compute the probability of no two people having the same birthday.The probability that k people don't have the same is $\frac{(365)\cdot(364)\cdot(363)...(365-k+1)}{365^n}$. From this we subtract by one to obtain $1 - \frac{(365)\cdot(364)\cdot(363)\cdots(365-k+1)}{365^n}$.

4 Consider a hash table with length $n \geq 1$, and let h be the has function with $h(k) = 0$ for every key $k \in U$. Suppose a data set S is inserted into the hash table, with $|S| \leq n$. What is the typical running time of subsequent $LOOKUP$ operations.

1. $\Theta(1)$ with chaining, $\Theta(1)$ with open addressing.
2. $\Theta(1)$ with chaining, $\Theta(|S|)$ with open addressing.
3. $\Theta(|S|)$ with chaining, $\Theta(1)$ with open addressing.
4. $\Theta(|S|)$ with chaining, $\Theta(|S|)$ with open addressing.

(4) is correct. Regardless of implementation methods we will have to in the worst case search all the way until the end of the corresponding data structure to check if a certain key exists in our data structure.

5 Why is it impractical to use a completely random choice of a hash function?

1. Actually, it is practical.
2. It is not deterministic.
3. It would take too much space to store.
4. It would take too much time to evaluate.

(3) and (4) is correct.

6 Which hash table strategy is feasible for loads larger than 1?

1. Both chaining and open addressing.
2. Neither chaining nor open addressing.
3. Only chaining.
4. Only open addressing.

(3) is correct. the α must be less than one at all times or else an infinite loop can occur with most implementations of open-addressing.

7 Suppose a data set S is inserted into a bloom filter that uses m hash functions and a length-n bit array. Under our heuristic assumptions, what is the probability that the array's first bit is set to 1?

1. $(\frac{1}{n})^{|S|}$

2. $(1 - \frac{1}{n})^{|S|}$
3. $(1 - \frac{1}{n})^{m \cdot |S|}$
4. $1 - (1 - \frac{1}{n})^{m \cdot |S|}$

(4) is correct.

8 Which of the following is not a property you would expect a well-designed hash function to have?

1. The hash function should spread out every data set roughly evenly across its range.
2. The hash function should be easy to compute.
3. The hash function should be easy to store.
4. The hash function should spread out most data sets roughly evenly across its range.

(1) it is impossible to guarantee that a hash function should spread out every data set roughly evenly across its range. In fact there exists a pathological dataset for every hash function.

9 A good hash function mimics the gold standard of a random function for all practical purposes, so it's interesting to investigate collisions with a random function. If the location of two different keys $k_1, k_2 \in U$ are chosen independently and uniformly at random across n array positions, what is the probability that k_1 and k_2 will collide.

1. 0
2. $\frac{1}{n}$
3. $\frac{2}{n \cdot (n-1)}$
4. $\frac{1}{n^2}$

(2) is correct. We first insert the key k_1, it has n slots available to be placed in. We now insert k_2, the probability that k_2 maps to the same index as k_1 is $\frac{1}{n}$.

10 We interpreted our heuristic analysis of bloom filters in section 12.6 by specializing it to the case of 8 bits of space per key inserted into the filter. Suppose we were willing to use twice as much space. What can you say about the corresponding false positive rate, according to our heuristic analysis, assuming that the number m of hash tables is set optimally?

1. The false positive rate would be less than 1%.
2. The false positive rate would be less than 0.1%.
3. The false positive rate would be less than 0.01%.
4. The false positive rate would be less than 0.001%.

(2) is correct. The estimate of false positive frequencies is given by the equation $(1 - e^{-\frac{m}{b}})^m$. Since m is set optimally the value is the same as $ln2 \cdot b$. hence our equation simplifies down to $(\frac{1}{2})^{(ln2) \cdot 16}$. Let $b = 16$ we get $(\frac{1}{2})^{(ln2) \cdot b} \approx 0.0005$ which is less than 0.04%.