Exercises 21 : lower bounds on sorting, bucket sort, independence (modified April 6 – dropped 2 questions formerly Q3,Q4)

Questions

1. Give a *lower* bound on the best case scenario for any comparison-based sorting algorithm, as a function of n. That is, you choose a sorting algorithm and a problem size n. This defines a binary decision tree. You then consider the problem instance that involves the fewest number of comparisons ("best case"). The question asks for a lower bound on that number.

2. Similar to the previous question, but now give an *upper* bound on the best case scenario for any comparison-based sorting algorithm, as a function of n.

3. What is the sample space for bucket sort ?

4. For bucket sort, suppose you don't restrict yourself to [0,1). The xj can all be doubles. How would you deal with this range?

5. Consider a sample space S and two different outcomes s1 and s2 in S. Intuitively, the two outcomes are not independent since if one outcome occurs in an experiment then the other does not. Show that they are not independent, according to the mathematical definition given in the lecture.

6. Consider coin flipping with an unfair coin where the fail probability is p0 and the success probability is 1-p0. I showed in class that the expected value of number X of flips until you get a head is 1/(1 – p0). What is intuition here for what happens when p0 is near 0, 1 ?

7. Recall E3 (hashtable) Question 9 and E19 (average case) Q2. Consider the event E that no two people have birthdays on the same date. Write this event as the intersection of a number of independent events. Use these independent events to calculate the probability of the event E.

Answers

1. The best case occurs for a problem instance in which the path from root to leaf in the decision tree is as short as possible. It is possible to make a decision tree in which one of the paths has length n-1. The sorting algorithm might start out by asking "is x1 < x2?" and if the answer is yes it might then ask "is x2 < x3?" and if the answer is yes then it might ask "is x3 < x4?" etc. and then "is x_n-1 < xn ?". If the answers are all yes'es, then we know the ordering and we've determined it in n-1 questions. Note you have to ask at least that many questions to be sure that you know the true order.

2. The best case is the shortest path from root to leaf in a decision tree. A upper bound on the shortest path is obtained when the shortest path is as long as possible, which happens when the

tree is balanced. In this case, all paths are of length log (n!). Using an argument similar to what we saw in class, log (n!) < log (n^n) = n log n.

3. One way to define the sample space is to consider n-tuples of real numbers or doubles in [0,1). However, what really matters is the buckets that these numbers get put into. So a better definition of the sample space would be the different ways of putting n numbers into n buckets. The sample space is of size n^n since each x_j could go into any of the n buckets.

4. Take one pass through the points and find the largest and smallest values. Then remap the values by computing y_j = (x_j – x_min)/ (x_max – x_min). This maps the x_j to the interval [0, 1]. You can make n buckets on that interval and apply the algorithm from class. To print out the y values instead of the x's, you need to "invert" that mapping: x_j = x_min + (x_max – x_min)*y_j.

5. Two events E1 and E2 are independent if p(E1 and E2) = p(E1) * p(E2). The question asks about two events E1 = {s1} and E2 = {s2} where we assume p({s1}) > 0 and p({s2} > 0. Each of these events consists of single outcome and so p({s1} ∩ $\{s2\}$) = $p(\{\}) = 0$. But since p({s1}) and p($\{s2\}$) are both different from 0, their product p({s1}) p($\{s2\}$) must also be different from 0. Thus, p({s1} ∩ $\{s2\}$) ≠ p({s1}) p($\{s2\}$) and so the events are not independent.

6. As p0 goes to 0, 1-p0 goes to 1 and the formula says that the expected value of X goes to 1. That is makes sense since, when the probability of a success is near 1, X is almost never different from 1. As p0 goes to 1, the probability of a success is close to 0, so it takes more coin flips for a success. Why *exactly* the expected number goes like 1 over the probability of success is *not intuitively clear* (at least not to me) but that's what the math says.

7. Recall the sample space is of size 365^n. Take the first person. He/she has a birthday. We don't know what it is and we don't care. Now consider the second person. We can define an event E2 that this second person has a different birthday than the first person. The probability of this event is 364/365. Next define an event E3 that the third person has a different birthday than the first person and the second person. This event is assumed to be independent of E2 and the probability of this event is 363/365. Continuing with this reasoning, the probability that no two of the n people have the same birthday is

   p(E) = p(E2) * p(E3) * p(E4) * …. p(E_n) = (364/365) * (363/365) * ... * (365 – (n – 1)) / 365.