1. Let \( n \) be a positive integer, \( \beta \) be any real number greater than 1, and let \( X \) be a random variable uniformly distributed over \( \{0, \ldots, n-1\} \). Show that \( E[X^\beta] \leq n^\beta/(\beta + 1) \). Hint: estimate a sum by an integral.

2. This exercise develops a proof that the expected depth of the recursion tree for QuickSort is \( O(\log n) \).

   Again, we assume that all inputs are distinct. To set the stage for this proof, let us first define the notion of an infinite complete binary tree: this consists of a root (which is at level 0), which has two children (which are at level 1), each of these has two children, and so on. So for \( i = 0, 1, 2, \ldots \), at level \( i \) of the tree, there are \( 2^i \) nodes.

   For each node \( s \) in this tree, we define the random variable \( N_s \) to the the size of the QuickSort subproblem associated with this node. So, for example, let \( r \) be the root of the tree, and let \( p \) and \( q \) be its two children. Then \( N_r = n \), and \( N_p \) and \( N_q \) is uniformly distributed over \( \{0, \ldots, n-1\} \), and \( N_q = n-1-N_p \).

   Note that \( N_q \) is also uniformly distributed over \( \{0, \ldots, n-1\} \), although \( N_p \) and \( N_q \) are certainly not independent.

   (a) Using Exercise 1 and the law of total expectation, show by induction of \( i \) that if \( s \) is any node at level \( i \) of the tree, then \( E[N_s^2] \leq (1/3)^i n^2 \).

   (b) For \( i \geq 0 \), let \( T_i = \sum_s N_s^2 \), where the sum is over all \( 2^i \) nodes \( s \) at level \( i \) of the tree. Show that \( E[T_i] \leq (2/3)^i n^2 \). Hint: linearity of expectation.

   (c) Let \( D \) be the recursion depth of QuickSort. Using Markov’s inequality and part (b), show that \( P[D \geq i] \leq (2/3)^i n^2 \). Hint: if \( D \geq i \), what can we say about about \( T_i \)?

   (d) Using part (c), show that \( E[D] \leq 2\log_{3/2}(n) + O(1) \). Hint: Write \( E[D] = \sum_{i \geq 1} P[D \geq i] \), and split the sum in two pieces.

3. Here is another way to analyze the expected running time of QuickSort. As usual, assume that the input is a list \( L \) of \( n \) distinct items. Let \( T_L \) be a random variable representing the running time on input \( L \), and define \( \tilde{T}(n) \) the be the maximum value of \( E[T_L] \) over all lists \( L \) of length equal to \( n \). Also assume that the running time of the partition step and any other computations outside of the recursive calls is at most \( cn \), where \( c \) is an implementation-defined constant, and this bound is independent of \( L \).

   (a) Let \( X \) be a random variable representing the relative position (in sorted order) of the randomly chosen pivot. So \( X \) is uniformly distributed over \( \{1, \ldots, n\} \). Let \( S_< \) and \( S_> \) be random variables representing the running times to solve the two subproblems obtained after the partition step. We have \( T_L \leq cn + S_< + S_> \).

   Use the law of total expectation to argue that

   \[
   E[T_L] \leq cn + \frac{1}{n} \sum_{i=1}^{n} \left( E[S_< \mid X = i] + E[S_> \mid X = i] \right).
   \]

   (b) Using part (a), argue that

   \[
   \tilde{T}(n) \leq cn + \frac{2}{n} \sum_{i=1}^{n-1} \tilde{T}(i).
   \]

   (c) Prove by induction on \( n \) that

   \[
   \tilde{T}(n) \leq 2c(n \ln(n) + n). \tag{1}
   \]

   Hints: use strong induction; that is, to prove (1) for an arbitrary \( n \geq 1 \), assume that \( \tilde{T}(i) \leq 2c(i \ln(i) + i) \) for all \( i = 1, \ldots, n-1 \). Also, you may want to estimate the sum \( \sum_{i=1}^{n-1} 2c(i \ln(i) + i) \) by the integral \( \int_1^n 2c(x \ln(x) + x)dx \). Go ahead and use an online resource, such as Wolfram Alpha, to help calculate the indefinite integral \( \int x \ln(x)dx \).
4. You have a mixed pile of \( n \) nuts and \( n \) bolts and need to quickly find the corresponding pairs of nuts and bolts. Each nut matches exactly one bolt, and each bolt matches exactly one nut. By fitting a nut and bolt together, you can see which is bigger. But it is not possible to directly compare two nuts or two bolts. Design and analyze a probabilistic algorithm for this problem with an \( O(n \log n) \) expected running time. Hint: customize QuickSort to the problem. Side note: only a very complicated deterministic \( O(n \log n) \) algorithm is known for this problem.

5. A bag is an abstract data type which may hold some number of distinct items. You may query the bag to determine how many items are in the bag. This takes constant time. You may also apply the probabilistic operation \( \text{split} \) to a bag, which partitions the items in the given bag into two new bags. All you know about \( \text{split} \) is that for every pair of distinct items in the input bag, the probability that both items end up in the same output bag is \( \leq \frac{1}{2} \). Also, if a bag contains \( n \) items, applying \( \text{split} \) to the bag takes time \( O(n) \).

Your goal is to design and analyze a probabilistic algorithm that takes as input a bag containing \( n \) items \( i_1, \ldots, i_n \), and produces as output \( n \) bags \( B_1, \ldots, B_n \) such that bag \( B_j \) contains the single item \( i_j \). The ordering of the output bags is irrelevant. Your algorithm should run in expected time \( O(n \log n) \).

Here is an outline you can follow:

(a) The algorithm is the obvious divide and conquer algorithm, using the \( \text{split} \) operation to get two subproblems, and then recursing on both, as necessary. Write out this algorithm.

(b) To analyze the running time, consider any two items in the original bag, and argue that after \( d \) levels of recursion, the probability that they remain in the same bag is at most \( 2^{-d} \).

(c) Using (b), argue that for any particular item in the original bag, the probability that it is not in a bag by itself after \( d \) levels of recursion is at most \( 2^{-d(n-1)} \). Hint: union bound.

(d) Using (c), show that for any particular item \( i \) in the original bag, if \( D_i \) is the depth in the recursion tree at which \( i \) ends up in a bag by itself, then \( E[D_i] = O(\log n) \). Hint: \( E[D_i] = \sum_{d \geq 1} P[D_i \geq d] \).

(e) Finally, argue that the running time \( T \) is bounded by \( c \sum_{i} (D_i + 1) \), where the sum is over all items \( i \) in the original bag, and from this and part (d), argue that \( E[T] = O(n \log n) \). Hint: linearity of expectation.

The following is for honors students only.

H1. Exercise 2 established a depth of \( c \ln(n) + O(1) \) on the expected depth of the recursion tree of QuickSort, where \( c = 2 / \ln(3/2) \approx 4.93 \). Derive a better bound, obtaining a smaller constant \( c \), using the following method: instead of using sums of squares \( \sum_s N_s^2 \), consider sums of arbitrary powers \( \sum_s N_s^\beta \), where \( \beta \) is any constant greater than 1. If you follow the same steps as in Exercise 2, you should find a formula for \( c \) as a function of \( \beta \), and then be able to choose \( \beta \) that minimizes this function. What are the optimal values of \( \beta \) and \( c \) that you find?