CPSC 351 — Tutorial Exercise #16
Probability Distributions

These problems are intended to help you review ***sample spaces*** and ***probability distributions***, and the use of these to compute probabilities of events.

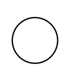
Problems To Be Solved

A ***binary tree*** is a recursively defined data structure consisting of a finite number of *nodes* and relationships between them:

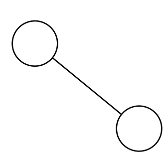
- An ***empty tree*** does not include any nodes, at all.
- Every other binary tree includes a node, called the ***root*** of the tree, along with a ***left subtree*** and a ***right subtree***, which are both binary trees. The left and right subtrees do not include the root and do not have any nodes in common.
- In pictures, the left and right subtrees are shown below the root (the left and right, respectively); if the subtrees are not empty then edges are shown between the root and the roots of these subtrees.

The ***size*** of a binary tree is the number of nodes in the tree.

For example, a binary tree with size one (so that the left and right subtrees of the root are empty tree) is drawn as follows.

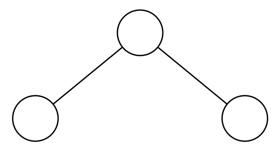


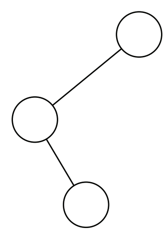
One example of a binary tree with size two is as follows.



Here, the *left subtree* is an empty tree and the *right subtree* is a binary tree with size one.

Two example binary trees with size three are as follows.





Consider now a large set (or "universe") $\mathcal{U}$ that has a ***total order***: There is a binary relation, "$<$", between elements of $\mathcal{U}$ which satisfies the following properties.

(a) *Exactly* one of the following properties is satisfied, for each pair of (not necessarily distinct) elements $\alpha, \beta \in \mathcal{U}$: Either

$$\alpha < \beta, \qquad \beta < \alpha, \qquad \text{or} \qquad \alpha = \beta.$$

(b) For all $\alpha, \beta, \gamma \in \mathcal{U}$, if $\alpha < \beta$ and $\beta < \gamma$ then $\alpha < \gamma$.

A ***binary search tree***, that stores elements of $\mathcal{U}$, is a binary tree T satisfying the following properties.

(a) Each node in T stores an element of $\mathcal{U}$. The set of elements of $\mathcal{U}$, that are stored at the nodes of T is the subset of U "represented by" T.

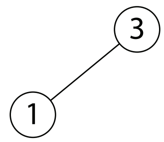
(b) If T is not an empty tree, α is the element of $\mathcal{U}$ stored at the *root* of T, and β is one of the elements of $\mathcal{U}$ stored at a node in the *left* subtree of T, then $\beta < \alpha$.

(c) If T is not an empty tree, α is the element of $\mathcal{U}$ stored at the *root* of T, and γ is one of the elements of $\mathcal{U}$ stored at a node in the *right* subtree of T, then $\alpha < \gamma$.

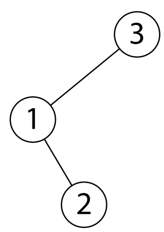
(d) The left and right subtrees of T are also binary search trees (so that each satisfies properties (a), (b) and (c) as well).

For example, suppose that $\mathcal{U} = \mathbb{Z}$, the relation "$<$" is the expected numerical order, and consider a binary search tree of size one, storing the set $\{3\}$ — so that 3 is stored at the root:

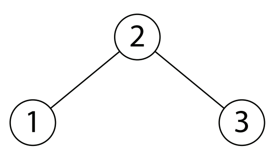
If 1 is to be added to the set that is represented (so that the tree would represent the set $\{1, 3\}$), then — since $1 < 3$ — a node storing 1 must be added to the *left* subtree:



Finally, if 2 is to be added to the set that is represented (so that the tree would represent the set $\{1, 2, 3\}$) then — since $2 < 3$ — a node storing 2 must be added to the *left* subtree, which has 1 stored at the root. Since $2 > 1$, a node storing 2 must be added to the *right* subtree of the left subtree:



On the other hand, if we inserted the same values into an initially empty binary search tree, but we used a different order — first 2, then 1 and, finally, 3 —then the following binary search tree is produced, instead.



In general, we might consider an arbitrary (infinite or large) universe $\mathcal{U}$ that has a total order, "$<$". For a positive integer n, we might consider a set of n values (or *keys*) $k_1, k_2, \ldots, k_n \in \mathcal{U}$, such that

$$k_1 < k_2 < k_3 < \cdots < k_{n-1} < k_n$$

and we might consider the experiment of inserting these values into an initially empty binary search tree, in some order. However, *in order to keep things simple,* let us suppose that $\mathcal{U} = \mathbb{Z}$, "$<$" is the usual *less than* ordering for integers, and that $k_i = i$ for each integer i such that $1 \leq i \leq n$.

Thus we are inserting the integers $1, 2, \ldots, n$ into an initially empty binary search tree, in some order. Each ***outcome*** of the experiment being considered can be represented as a sequence

$$(\alpha_1, \alpha_2, \ldots, \alpha_n)$$

consisting of the integers $1, 2, \ldots, n$, listed in some order — that is, a ***permutation*** of the sequence of integers $1, 2, \ldots, n$. The ***sample space*** Ω is the set of all such permutations. For example, if $n = 3$ then

$$\Omega = \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\} \tag{1}$$

— a set of size $3! = 6$.

1. Draw the binary search tree corresponding to each of the outcomes in the above sample space Ω (corresponding to binary search trees with size $n = 3$).

In a binary search tree, the ***left child of the root*** is the node that is the roof of the left subtree — so that there is no left child of the root if the left subtree is the empty tree. Similarly, the ***right child of the root*** is the node that is the root of the right subtree — so that there is no right child of the root if the right subtree is the empty tree. Each node can be viewed as the root of the ***subtree*** that includes that node and the nodes below it. We can now use this idea (considering the subtrees with a given node at the root, instead of the entire tree) to define the ***left child*** and the ***right child*** of each of nodes in the tree (besides the root) as well.

If x and y are nodes in a binary search tree then x is the ***parent*** of y if and only if y is either the left child of x or the right child of x. The root of a binary search tree does not have a root in that tree, while every other node has exactly one parent.

A node in a binary search tree is a ***leaf*** if and only if it does not have any children in that tree, in it is an ***internal node*** in the binary search tree, otherwise.

These terms — ***left child***, ***right child***, ***parent***, ***leaf*** and ***internal node*** — will be used in examples involving binary search trees in the rest of this course.

2. Let $\mathsf{P} : \Omega \to \mathbb{R}$ be the ***uniform probability distribution*** for this experiment (using the sample, space, Ω, as shown at line (1)).
   (a) Prove that the probability that "1 is stored at the root of the binary search tree" is $\frac{1}{3}$.
   (b) Prove that the probability that "1 is stored at a leaf" is $\frac{1}{2}$.
3. Consider the above experiment (with $n = 3$) once again, but consider a different function $\mathsf{P} : \Omega \to \mathbb{R}$ such that

   $$\mathsf{P}((1,2,3)) = \mathsf{P}((1,3,2)) = \frac{1}{4}, \quad \mathsf{P}((2,1,3)) = \mathsf{P}((2,3,1)) = \frac{1}{6}$$

   and

   $$\mathsf{P}((3,1,2)) = \mathsf{P}((3,2,1)) = \frac{1}{12}.$$

(a) Prove that this also defines a probability distribution for this experiment.

(b) Compute the probably that "1 is stored at the root", using this probability distribution.

(c) Compute the probability that 1 is stored at a leaf", using this probability distribution.