Lab 2

1. Sometimes MergeSort is supplemented with a secondary sorting routine (typically, InsertionSort is used) in the following way: During the recursion in MergeSort, the size of the array being sorted becomes smaller and smaller. To create a hybrid sorting routine, when a recursive call requires the algorithm to process an array with 20 or fewer elements, give this array to InsertionSort and patch in the result after it has finished. Call this hybrid algorithm MergeSortPlus.

    A. Express the steps of MergeSortPlus in the pseudo-code language we are using in class.
    B. Write the Java code for MergeSortPlus (use the implementation of MergeSort provided in the lab folder)
    C. Run tests to compare running times of MergeSort and MergeSortPlus. Which one runs faster? Explain how you tested and whether you feel your results are conclusive.

2. *Binary Trees.* A *binary tree* is a tree in which every node has at most two children.

    a. Write out 4 different binary trees, each having height = 3 – make sure that no two of your trees have the same number of nodes. (There is no need to give labels to the nodes.)

    b. Examine the trees you have drawn and decide whether the following statement is true or false:

        Every binary tree of height 3 has at most $2^3=8$ leaves.

    c. Based on your answer to b, what do you think is true in general about the number of leaves of a binary tree of height *n*?

3. **Power Set Algorithm**. Given a set X, the power set of X, denoted P(X), is the set of all subsets of X. Below, you are given an algorithm for computing the power set of a given set. This algorithm is used in the brute-force solution to the SubsetSum Problem, discussed in the first lecture. Implement this algorithm in a Java method:

    `List powerSet(List X)`

    Use the following pseudo-code to guide development of your code

```
Algorithm: PowerSet(X)
    Input: A list X of elements
    Output: A list P consisting of all subsets of X – elements of P are Sets

    P ← new list
    S ← new Set     //S is the empty set
    P.add(S)        //P is now the set { S }
    T  ← new Set
    while (!X.isEmpty() ) do
        f  ← X.removeFirst()
        for each x in P do
            T ← x ∪ {f}   // T is the set containing f & all elements of x
            P.add(T)
    return P
```

4. SubsetSum Problem: given a set S = {s_0, s_1,s_2, …, s_{n-1}}of positive integers and a non-negative integer k, is there a subset T of S so that the sum of the integers in T equals k?

Formulate your own procedure for solving the SubsetSum. Think of it as a Java method subsetsum that accepts input S and k, and outputs a subset T of S with the property that the sum of the elements in T is k if such a T exists, or null if no such T can be found.

Examples

- If S is [1, 3, 9, 4, 8, 5] and k = 21, return [9,4,8] (since 9 + 4 + 8 = 21)
- If S is [1, 3, 9] and k = 5, return null (since no such subset can be found)
- If S is [1, 3, 9, 4, 8, 5] and k = 0, return [] (since the sum of the empty set is 0)