Homework 1: Recursion and Backtracking. DUE: 11:59pm Mon, Sep 13, 2021

- Please type and submit PDF file to Blackboard. No late submissions will be accepted.
- **Academic honesty**: at the beginning of your paper, please include: "I understand and agree to abide by the provisions in the University of Houston Undergraduate Academic Honesty Policy" and "This is my own work."
- **Discussion is encouraged**. In each problem solution, list the names of persons that you have discussed with about that problem, except for large group discussion such as in MS Teams where a note of "group discussion" suffices. If an online source helped you, list the URL.

Solving Recurrence:

P0 (20pt): Solve the following recurrences. Give the answer in terms of Big-Theta notation. Assume base case has cost $T(1)=T(0)=1$. Ignore ceilings and floors.

1. $T(n)=3T(n/2)+n$
2. $T(n)=T(n/2)+T(n/3)+n$
3. $T(n)=2T(n-1)+2$
4. $T(n)=3T(n/3)+n^2$
5. $T(n)=T(3n/4)+T(n/3)+n$

Recursion: Reduce/Divide and Conquer:

P1 (20pt): You are given n stones (assume that n is a power of 2) each having a distinct weight. You are also given a two-pan balance scale (no weights are given). For example, given two stones, you can use the scale to compare which one is lighter by placing the two stones on the two different pans. The goal is the find the heaviest and the lightest stone by using as few weighings as possible. Give a divide and conquer strategy that uses only $\frac{3n}{2}-2$ weighings, or less.

P2 (20pt): (a) Given two sorted arrays $A[1\ldots n]$ and $B[1\ldots n]$. Describe an algorithm to find the median element in the union of A and B in $\Theta(\log n)$ time.

(b) **[optional]** Given two sorted arrays $A[1\ldots m]$ and $B[1\ldots n]$ and an integer k. Describe an algorithm to find the k-th smallest element in the union of A and B (i.e. $A\cup B$) in $\Theta(\log(m+n))$ time. [Hint: use your solution in part (a)].

P3 (20pt): You are a contestant on the hit game show "Beat Your Neighbors!" You are presented with an length n array of boxes, each containing a unique number. It costs \$100 to open a box. Your goal is to find a box whose number is larger than its neighbors in the array (left and right). Describe an $O(\log n)$ algorithm that finds a number that is bigger than either of its neighbors.

P4 (20pt): You are given an $n \times n$ matrix represented as a 2-dimensional array $A[1..n][1..n]$ (there are totally n^2 elements). Each row of A is sorted in increasing order and each column is sorted in increasing order. Your goal is to find whether some given element x is in A or not. Note that you can do only comparisons between elements (no hash table, or any other set data structures). Give an algorithm that takes $O(n)$ comparisons. [Hint: try to start on one some corner, and eliminate one row/column per move].

Backtracking:

P5 (20pt): An addition chain for an integer n is an increasing sequence of integers that starts with 1 and ends with n, such that each entry after the first is the sum of two earlier entries. More formally, the integer sequence $x_0 < x_1 < \cdots < x_l$ is an addition chain for n if and only if

- $x_0 = 1, x_l = n$
- For every index k, there are indices $i \leqslant j < k$ such that $x_k = x_i + x_j$.

The $l$ength of an addition chain is the number of elements minus 1 ; we don't bother to count the first entry. For example, <1,2,3,5,10,20,23,46,92,184,187,374> is an addition chain for 374 of length 11.

Describe a recursive backtracking algorithm to compute a minimum length addition chain for a given positive integer n. Don't analyze or optimize your algorithm's running time, except to satisfy your own curiosity. A correct algorithm whose running time is exponential in n is sufficient for full credit. [Hint: This problem is a lot more like N-Queens than text segmentation.]