You are allowed to discuss with others but not allow to use references other than the course notes and reference books. Please list your collaborators for each questions. Write your own solutions and make sure you understand them.

There are 65 marks in total (including the bonus). The full mark of this homework is 50. Your submission should be in PDF format generated by LaTeX. You may use the LaTeX template at:

https://www.overleaf.com/read/tsxxqdgjzhxx

Enjoy :).

Problem 1: Searching [15 marks]

(a) (5 marks) Consider the following `MysterySearch` algorithm.

```
// The classic binary search algorithm that we know.
int BinarySearch(int arr[], int low, int high, int x);

// The mystery search algorithm.
int MysterySearch(int arr[], int n, int x) {
  int low = 1; // arr[1] is the first element in arr[].
  while (low <= n && arr[low] < x) {
    low *= 2;
  }

  return BinarySearch(arr, low/2, min(n, low), x);
}
```

Assume that `x` can be found in the array `arr` at index r, that is, `arr`$[r]$ = `x`. What is the time complexity of `MysterySearch` in terms of r?

(b) (5 marks) Given two sorted arrays A and B, each of size n, devise an algorithm to find the k-th smallest element of among A and B. You will get full marks if the algorithm is of complexity $O(\log k)$.

(c) (5 marks) Let there be a $n \times n$ matrix $(a_{i,j})_{1\le i\le n, 1\le j\le n}$ such that $a_{i,j} \le a_{i+1,j}$ and $a_{i,j} \le a_{i,j+1}$, that is, the elements of each row or column of the matrix is in increasing order. Devise an algorithm that takes as inputs the matrix $(a_{i,j})_{1\le i\le n, 1\le j\le n}$ of size $n \times n$ and a number b, and that outputs whether there is an element $a_{i,j}$ such that $a_{i,j} = b$ for some number b. You will get full marks if the algorithm is correct of complexity $O(n^{1.59})$.

Problem 2: Multiplication [10 marks]

(a) (Matrix-vector multiplication) Recursively define a sequence of matrices as follows. Let $H_0 = 1$ and for $k > 0$ let

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}. \tag{1}$$

As such, H_k is of size $n \times n$ where $n = 2^k$. Design an algorithm to compute the matrix-vector multiplication $H_k x$, where $x \in \{0,1\}^n$. You will get full marks if the algorithm is correct and takes $O(n \log n)$ integer multiplication and addition operations.

(b) (Polynomial multiplication) Given two polynomials $P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ and $Q(x) = b_n x^n + b_{n-1}x^{n-1} + ... + b_1 x + b_0$, provide an algorithm to multiply these two polynomials in $O(n^{\log_2 3}) = O(n^{1.59})$ word operations (i.e. each addition and multiplication in computing coefficients of $P \cdot Q(x)$ is considered one word operation). You can assume that $n+1$ is a power of two. State the algorithm clearly and explain its time complexity.

Problem 3: Mergesort [10 marks]

(a) (5 marks) Given four lists of n numbers each, devise an $O(n \log n)$ algorithm to determine if there is any number common to all four lists.

(b) (5 marks) Given an array $A[1,\dots,n]$, its subarray $A[i,\dots,j]$ is called a maximal increasing sequence (MIS) if and only if it is sorted and either of the following holds:

- $i = 1$ and $j = n$,
- $i > 1, j = n$, and $A[i-1] > A[i]$,
- $i = 1, j < n$, and $A[j] > A[j+1]$,
- $i > 1, j < n$, and $A[i-1] > A[i], A[j] > A[j+1]$.

Example: in the array $(3, 1, 2, 4, -1, 9, 8)$, there are 4 maximal increasing sequences say $(3), (1, 2, 4), (-1, 9), (8)$.

Consider the following version of mergesort.

```
// merge the sorted arrays B and C into the output array.
int* merge(int B[], int C[]);

int* mergesort2(int A[1,...,n]) {
    int MIS[];
    int i = 1;

    Find j such that A[i,...,j] is a maximal increasing sequence.
    MIS = A[i,...,j];

    while(j+1 <= n) {
        Find k such that A[j+1,...,k] is a maximal increasing sequence.
        MIS = merge(MIS, A[j+1,...,k];
        j = k;
    }

    return MIS;
}
```

What is the running time of `mergesort2` in terms of the array size and the number of maximal increasing sequences in the array?

Problem 4: Counting inversions [15 marks + 10 marks]

(a) (5 marks) There are n students and each student i has 2 scores x_i and y_i. Students i and j are friends if and only if $x_i < x_j$ and $y_i > y_j$. Devise an $O(n \log n)$ algorithm to count the number of pairs of friends.

(b) (10 marks) Let there be an array of 2D pairs $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$. With a fixed constant y' a pair (i, j) is called *half-inverted* if $i < j$, $x_i > x_j$, and $y_i \geq y' > y_j$. Devise an algorithm that counts the number of half-inverted pairs. You will get full marks if your algorithm is correct of complexity no more than $O(n \log n)$.

(c) (**Bonus**: 10 marks) A pair (i, j) is inverted if $i < j, x_i > x_j$, and $y_i > y_j$. Devise an algorithm as efficient as possible that counts the number of inverted pairs.

Problem 5: Randomized binary search [Bonus: 5 marks]

In the binary search algorithm, it is typical to take the middle element of the (sub-)array of length n as the pivot in each iteration. We can also choose uniformly at random an element in the (sub-)array as our pivot. Assume that randomly choosing such a pivot takes $\Theta(1)$ time. Prove that the expected running time of the randomized binary search algorithm is $O(\log n)$.