CS 240: Data Structures and Data Management **Spring 2023**

Tutorial 11: July 31

1. **Karp-Rabin**
   For Karp-Rabin pattern matching, consider the following hash function for strings over the alphabet $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$:

$$\begin{aligned} h(P) =& (\#\text{ of occurrences of } \texttt{A}) + 2 \times (\#\text{ of occurrences of } \texttt{C}) \\ &+ 3 \times (\#\text{ of occurrences of } \texttt{G}) + 4 \times (\#\text{ of occurrences of } \texttt{T}) \end{aligned}$$

   Given the pattern $P = \texttt{TAGCAT}$ and sequence $T = \texttt{TGCCGATGTAGCTAGCAT}$, use the table below to show all the character comparisons performed during Karp-Rabin pattern matching. Start a new pattern shift (in which character comparison occurs) in a new row. You may not need all the available space.

T	G	C	C	G	A	T	G	T	A	G	C	T	A	G	C	A	T

Table 1: Table for Karp-Rabin problem.

2. **Boyer-Moore, Revisited**
   Let $P[0...5] = \texttt{payday}$ and let $T[0...12] = \texttt{daypayplayayaya}$.

   a) Compute the Last-Suffix Array $S[0...5]$ for the pattern P.

   b) Show the search for P in T using the Boyer-Moore algorithm using only the bad character heuristic. Also, put square brackets around characters that are known to be matched, even if the algorithm matches them again.

3. **Moore's first name is 'J', not abbreviated**
   Consider using the Boyer-Moore algorithm with only the Bad Character heuristic to search for a pattern P of length m in a text T of length n, with $n > m$, where P does **not** appear in T.

   a) Give an example of a pattern P with length n and text T with length n that achieves the worst-case runtime for searching. Do not consider preprocessing time.

   b) Same question, but for the best-case runtime.

4. **Range Search**

   Assume that we have a set of n numbers (not necessarily integers) and we are inter- ested only in the number of points that lie in a range rather than in reporting all of them. Describe how a 1-dimensional range tree (i.e., a balanced binary search tree) can be modified such that a range counting query can be performed in $O(\log n)$ time (independent of k).

5. **Programming Question: 2D Range Tree Search**

   Program a 2D range search function with the given starter code. Feel free to add any helper functions or change function signatures at will.

```
#include <iostream>
#include <vector>
#include <utility>

using namespace std;

class _1DRangeTree{
    private:
        int y;
        pair<int, int> coord;
        _1DRangeTree* leftChild;
        _1DRangeTree* rightChild;
    public:
        _1DRangeTree(vector<pair<int, int>>);
        vector<pair<int, int>> rangeSearch(int left, int right);
};
class _2DRangeTree{
    private:
        int x;
        int y;
        pair<int, int> coord;
        _1DRangeTree* associated;
        _2DRangeTree* leftChild;
        _2DRangeTree* rightChild;
    public:
        _2DRangeTree(vector<pair<int,int>>);
        vector<pair<int, int>> search(int xleft, int xright, int yup,
        ydown, bool isLeftBounded, bool isRightBounded);
};

int main(){
// initialize
    _2DRangeTree Root;
    Root.search(xleft, xright, yup, ydown, false, false);

}
```