CMSC858N, Spring 2023, University of Maryland (Dhulipala)
HW2 (Theory), due as **PDF at Gradescope** by 5PM on March 31st

Notes: (i) List the other students with whom you discussed the problems on this assignment: *such collaboration is allowed, but you need to write up your solutions by yourself.* If you did not discuss any problems with your classmates, please indicate this. Consulting other sources (including the Web) is not allowed. (ii) Write your solutions neatly; if you are able to make partial progress by making some additional assumptions, then **state these assumptions clearly and submit your partial solution**.

1. *Leaffix and Rootfix Sums.* Given a rooted tree T with $n = |T|$ nodes, consider the *leaffix* and *rootfix* sums.

- The leaffix algorithm processes a tree from the bottom to the top. The sum for a leaf is 0. For each node v, v's sum is the sum of all of its children's sums.
- The rootfix algorithm processes a tree from the top to the bottom. The sum at the root is 0. For each node v, its sum is the parent's sum plus the value at the parent.

These concepts are illustrated in the figure below:

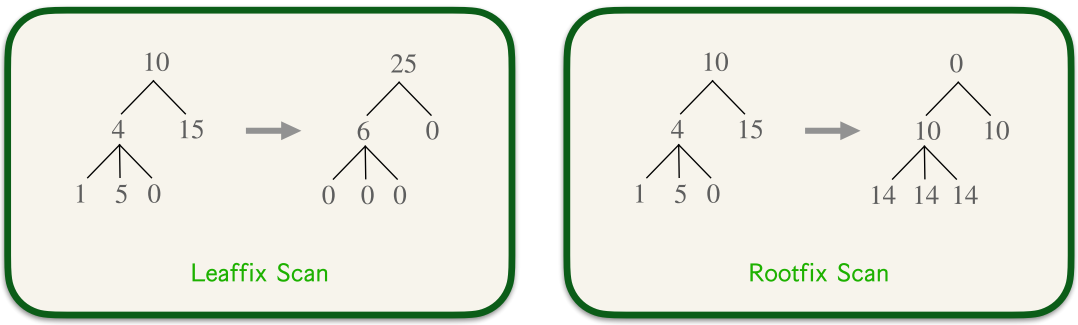

Figure 1: Leaffix and Rootfix Sums on a rooted tree, T.

Show how to compute for each node $v \in T$, the leaffix and rootfix sums of all nodes in T in $O(n)$ work and $O(\log n)$ depth. **(10 points)**

2. *Tree Contraction and Expansion.* The parallel tree contraction algorithm we saw in class *contracts* a tree T into a single node by repeatedly applying the `Contract` operator, which (1) rakes all the leaves and (2) contracts an independent set of the degree-1 children. In this problem, we will revisit tree contraction, finish fleshing out some of the details left out in class, and apply it to a new application.

- First consider *expanding* from a single node into the original tree, T by reversing the contraction process. Just like `rake` and `compress` remove nodes from the tree, let's define new operators, `unrake` and `uncompress` which reintroduce leaves, and reintroduce previously compressed degree-1 nodes, respectively. Give a high-level description of how to implement an expansion algorithm that applies any user-defined `unrake` and `uncompress` operations. The algorithm should run in the same work and depth bounds as the original tree contraction algorithm. You can give high-level pseudocode or explain in words. **(5 points)**
- Explain how to use `rake`, `unrake` and `compress` and `uncompress` to solve the subtree maximum problem, defined as follows: Given a rooted tree T we would like to compute for each $v \in v$ the max of all nodes in v's subtree. In other words, if the set of nodes in v's subtree is $S(v)$, we would like to compute $\max_{u \in S(v)} \mathsf{value}(u)$. Your algorithm should run in the same work and depth as performing parallel tree contraction on T. **(5 points)**

3. *Pretty-Printing Paragraphs.* Given an integer sequence W of length n representing the length of each word in an n-word paragraph, and a length L representing the number of characters that can fit on a line, you would like to break the words in the paragraph into lines of length at most W. The output of the algorithm is a sequence B of length n where $B[i] = 1$ if word i starts a new line.

You have the following serial code which performs line breaking. The algorithm below greedily keeps packing words on a line until adding the next word would make the length go over W; this next word will then start a new line.

```
int cur_length = W[0]; B[0] = 1;
for (i=1; i < n; ++i) {
  s += W[i];
  B[i] = 0;
  if (s > L) {
    B[i] = 1;
    s = W[i];
  }
}
```

Design a parallel algorithm to compute the same output as the serial algorithm above that runs in $O(n)$ work and $O(\log n)$ span. Do not assume that L is a constant (i.e. the work and span must be independent of L). **(15 points)** *Hint: think about using list-ranking and the Euler-Tour Technique*