Assignment 8. Amortized analysis, coding, etc.

Given April 5, due April 11 (Wednesday).

1. Consider a simpler implementation of the min-max heap with operations deleteMin(), deletemax(), insert(item x), getMin() and getMax(). The data structure will be a pair of heaps, the max heap holding every item with x.key > div and the min heap holding the rest. To insert, test whether x.key is greater of less than div and insert into the appropriate heap. To do deleteMin(), use the min heap, and so on. If one of the heaps becomes empty, we have to stop and refill it with half the elements in the other heap. This can be done using (randomized) quick select looking for the median (the 50% quantile. The median will be the new div. Give an amortized analysis to show that the work per operation is $O(\log(n))$, where $n$ is the maximum number of elements in the system at any given time.

2. Stenographers (used to) have symbols for common words as well as for letters. Coding common words or expressions can compress data better than just coding letters. Suppose that we have a code with codes for all the letters and a large number of words. This code is organized as a trie, with pointers to a binary code sequence at the end of each word that has one. Recall that a trie is a tree with branching factor the number of letters in the alphabet, so that allowed words correspond to paths starting at the root and ending at a pointer (here to a code). Every leaf has a pointer to a code, or we would have stopped before. In this problem, take this coding as given. One difficulty in this coding scheme is that there may be several ways to represent the same text. For example if we have codes for an, and, and drew, as well as for the individual letters, then there are several ways to code the word andrew ([an|drew], [and|r|e|w], among others). The length of the resulting coded text will depend on the lengths of the code words for an, drew, and the individual letters. Give an algorithm that finds a shortest coding of a (long) character string. Present the algorithm in detail, including the traversal of the trie to find which character sequences have codes. Use dynamic programming, where the cost to go function, $C(k)$, is the minimal code length for the last $k$ characters in the string. You can compute $C(k+1)$ from $C(k)$ by a single traversal of the trie starting from the character $k+1$ spaces from the end.

3. Each person has a taste in music. In our computer, this is represented by letting the variable person.taste have the values ROCK or JAZZ. A person is an object of data type individual. There are $N$ people, in an array individual people[N]; Each person has a number of friends, stored in person.Nfriends, and a list of friends, stored in the array person.friends. If a person has a friend who has a different taste in music, that creates discord. The total discord is computed as follows (please excuse the C bugs, I hope the meaning is clear):
```c
int discord = 0; int M;
individual person, friend;
for ( int i = 0; i < N; i++) {
    person = people[i];
    M = person.Nfriends;
    my_taste = person.taste
    for ( int j = 0; j < M; j++) {
        friend = person.friends[j];
        their_taste = friend.taste;
        if ( my_taste != their_taste ) discord++;
    }
}
```

We have the option to change the taste of $k$ people and want to do so to lower the discord as much as possible. Design a greedy strategy to do this. Your strategy should change the taste of one person at a time, at each step switching the person who, at that stage, is creating the most discord. Does this strategy give the optimal switches?