Part 1:

Should have paid more attention in class and started earlier. Ran out of time.

Part 2:

1. I had a lot of difficulty with this homework. I figured out how to write min-cost-naïve easily. I am not sure how we were expected to write this function, but I chose to write a single recursive helper function rather than multiple helper functions to separate left, right, and root. I think the result is clear, efficient, and easy to understand. Translating this simplicity to min-cost was not easy.

   My original lookup table for min-cost was one-dimensional. I know this is what the instructions told us we needed but I could not figure out how to implement this with a one-dimensional table to save my life. The problem I had was that subtree costs evaluated differently depending on what level their root was placed at. To account for this, I changed my lookup table to a two-dimensional implementation. The first dimension is the level at which the subtree is rooted, and the second dimension is the list of (key weight) pairs. Once I had this list working, I was able to check if a specific set of (key weight) pairs had been evaluated from a certain level and if they had I could use that value. Otherwise it calculates the min-cost for that subtree and stores that value, plus the root node that gives that subtree its lowest value, in the lookup table. Again, I do not know if this is how we were supposed to implement this, but I cannot think of another way.

   As for the build-min-tree function I have run out of time and must finish my written portions instead of finishing the function. I think now that I finally have min-cost working it would be trivial to implement but I have to choose between something that I might not be able to finish and another large portion of the assignment.
2.
   - ‘keys-medium.dat”
     - (timed min-cost-naive data) = 14
     - (timed min-cost) = 0
   - “keys-large.dat”
     - (timed min-cost-naive data) = 4162
     - (timed min-cost) = 0
3. Ran out of time
4. Ran out of time