Andrew Letz

CIS 313

Professor Proskurowski

Written Homework #4

Question 1

To accomplish this problem, instead of sorting the satellite data itself, we can sort their keys (which point to the data) first and then move the data accordingly. A simple pseudo code example would be:

```
int* keys = [0x01230412, 0x01024891, etc.]
sorted_data = []
for (int i = 0; i < keys.length; i++) { # Insertion sort our keys
      int* val = keys[i];
      int pos = i;

      while (pos > 0 && keys[pos - 1] > val) {
            keys[pos] = keys[pos - 1];
            pos = pos - 1;
      }

      keys[pos] = val;
}
for (int i = 0; i < keys.length; i++) # for each key pointer in keys
      int* keyPointer = keys[i];
      move(*keyPointer, sorted_data[i]) # Imagine a move function that
                                     # gets dereferenced data at that pointer
                                     # stores it in index i of sorted_data
}
```

As seen in the second loop, there is linear (O(n)) movement of the satellite data as it iterates once for each record.

Question 2

Given n input keys where the values can be anywhere from 1 … n^2, a worse case use of bucket sort would not be enough to yield linearity. This is because this situation is not guaranteed to have a uniform distribution of values, meaning you must still scan all n^2 buckets in order to retrieve the sorted list of values.

Question 3

Example: (A):(0100 0001) (Z):(0101 1010) (P):(0101 0000)

1. Considering the keys A-Z (65-90 in ASCII encoding), there are 26 * 26 * 26 = 17,576 different 3-capital letter string keys. This is a small amount in comparison to the number of unique 3-byte values in binary representation (2^24 = 16,777,216).
2. This would be a bad choice of a hashing function because there would be several collisions as it is possible for an ASCII character to share its first 3 bits with another character. If you took the example above, the resulting hash would be 010 010 010, which would be the same if you changed the order of the letters (APZ, PZA, etc).
3. In the case of A-Z, the first nibble of a single character will either be a 4 or 5. This means we only need the last bit of the first nibble in order to make this distinction (is it 1 or 0).
   So the first step would be to take this bit from each character and append them. (in our example, 011).
   However, the last 4 bits of A-Z are always significant as they can range from 0000 to 1111. So, the second step would be to append these as well. (now we would have 011 0001 1010 0000)
   We can now convert this binary representation into an unsigned integer and use the modulo operator of 512 to get our values into 512 buckets.
   The example value of 011 0001 1010 0000 => 12704 would turn into 416.

Question 4

Choosing any four of the first seven digits of a 10-digit phone number would result in a large amount of collisions as the numbers are not “very” unique between employees. This can be seen in a simple example, where two employees might have distinct numbers 541 346 4428 and 541 346 4529 which would be mapped to the same output if using any of the first 7 digits. With a bad function like this a majority of numbers would be hashed into fewer than ten buckets.

Question 5

c would not work because the BST property is not present in this traversal.