CSCI E-119
Section Notes

Section 10: November 17, 2010

1. Heaps

(a) We will now convert the following 6-element array into a valid heap. We start by taking our array and representing it as a heap in the usual manner, with the first element as the root.

```
   0    1    2    3    4    5
-------------------------------
|  7 | 11 |  5 | 39 | 16 | 20 |
-------------------------------
```

```
1. This is the corresponding
complete tree.
                     ------
                     |  7 |
                     ------
                 ____/    \____
                /              \
           ------              ------
           | 11 |              |  5 |
           ------              ------
           /    \              /
      ------    ------    ------
      | 39 |    | 16 |    | 20 |
      ------    ------    ------
```

(b) How long does it take to create a heap from an array?

2. Heapsort

Here we will run through an example of heapsort. Suppose we have the following numbers in an array: 5 3 12 8 7 4 6. We want to sort the array in ascending order.

If we interpret the array as a complete tree, it looks like this:

```
        5
      /   \
    3      12
   / \     / \
  8   7   4   6
```

(a) What is the first step of heapsort?

(b) Should we create a min-heap or a max-heap?

(c) Let's step through turning our array into an actual heap:

(d) Now that we have made the array into a valid heap, what are the next steps in heapsort?

(e)Let's step through:

3. A* and Greedy Search

Suppose we have the following original configuration of the eight puzzle. We would like to compare the first few iterations of A* search and greedy search:

	2	5
4	1	6
8	7	3

(a) What is the h(x) for this configuration (The Manhattan Distance)?

(b) What are the successors of this configuration and their h(x) values? What are the priorities assigned in A* and greedy to each of these configurations?

(c) What are the successors of successor 1?

(d) After choosing successor 1, what is the next state greedy search looks at? What about A* search?

4. Hashing

Suppose we have a 7-element hash table, and we wish to insert following words:
apple, cat, anvil, boy, bag, dog, cup, down

We use hash functions:
h1(key) = index related to first letter of the word (‘a’ = 0, ‘b’ = 1, …)
h2(key) = length of the word (ex. h2(“apple”) = 5)

Let’s go through inserting elements using linear probing and count the total length of the probes:

Now let’s try quadratic probing:

Now let’s try double hashing:

5. The probe() method in our HashTable class
The return value of the probe() method is an integer. In some cases, it represents the index of the key that we're searching for. In other cases, it represents the index of the first empty or removed cell encountered during the search for the specified key.

0	aardvark
1	▨
2	cat
3	bear
4	
5	dog
6	▨

The hashtable above has been partially filled using linear probing and the hash function h1 from problem 1. A gray cell indicates that an item has been removed.

One of the items in the table has been inserted incorrectly. Which one, and how do you know?

For each of the keys below, determine:

i. the probe length
ii. the return value of the probe() method

Assume that none of these keys are actually inserted in the table.

a. bear

b. cow

c. buffalo

d. giraffe

What is the largest probe length that we could have for this table, regardless of its contents?