COMP2521 20T2 Data Structures and Algorithms

Week 08 Tutorial

Hashing, Higher-order Functions

1. Hash tables are initially empty (i.e. contain no items). This fact is represented in the table by each entry having a value `NoItem`.

   Suggest suitable `NoItem` values if `Item`s are stored in the table elements and

   a. keys are `unsigned int`s
   b. keys are `int`s
   c. keys are strings

   How would things be different if table entries were pointers to `Item`s (i.e. `(Item *)`)?

2. Consider this potential hash function:

```
1. int hash(char *key, int N)
2. {
3.     int h = 0; char *c;
4.     for (c = key; *c != '\0'; c++)
5.         h = h + *c;
6.     return <b>h % N;</b>
7. }
```

   How does this function convert strings to `int`s?

   What are the deficiencies with this function and how can it be improved?

3. Now consider a slightly more sophisticated hash function

```
1. int hash(char *key, int N)
2. {
3.    int h = 0;  char *c;
4.    int a = 127; // a prime number
5.    for (c = key; *c != '\0'; c++)
6.       h = (a * h + *c) % N;
7.    return h;
8. }
```

   that converts strings into integers in the range 0..N-1.

   Show the hash values returned for each of the following strings, assuming that `N = 11`.

   a. `"cat"`
   b. `"act"`
   c. `"actor"`

You can find the ascii values for characters on any Unix machine (including MacOS) by running the command:

```
$ man 7 ascii
```

4. Consider a hash table that handles collisions via linear probing

   - N = 10 table slots, hash(k) = k % 10

   Show the result of inserting items with these keys

   a. 1, 2, 3, 4, 5, 6, 7, 8
   b. 15, 6, 20, 3, 17, 14, 33, 5, 10, 7

   into an initially empty table.

5. For a hash table that uses separate chaining for collision resolution, with the chains sorted in ascending order on key value, what are the best case and worst case costs for inserting *k* items into the table. Assume that the size of the table *N = k / 2* and that the insertion cost is measured in terms of number of key comparisons, and that the chains are sorted in key order. What will be average search cost after all the insertions are done?

6. Consider a very small hash table with only 11 entries, and a simple hash function *h(x) = x % 11*. If we start with an empty table and insert the following values in the order shown

   ```
   11  16  27  35  22  20  15  24  29  19  13
   ```

   give the final state of the table for each of the following collision resolution strategies

   a. separate chaining, with chains in ascending order
   b. linear probing
   c. double hashing, with $h_2(x) = (x \% 3) + 1$

7. Using the heap-as-array ADT from lectures, show how an initially empty heap changes when the following values are inserted.

   ```
   S  O  R  T  I  N  G  I  S  F  U  N
   ```

   At each step, show how the `fixUp()` function is used.

8. Consider the heap-as-array ADT given in lectures:

   ```
   typedef struct HeapRep {
      Item *items;  // array of Items
      int  nitems;  // #items in array
      int  nslots;  // #elements in array
   } HeapRep;
   typedef HeapRep *Heap;
   ```

   If a heap is created to hold up to 10 `Items`, and `Items` are integer values, and higher values have higher priority, show the state of the heap after each of the following operations:

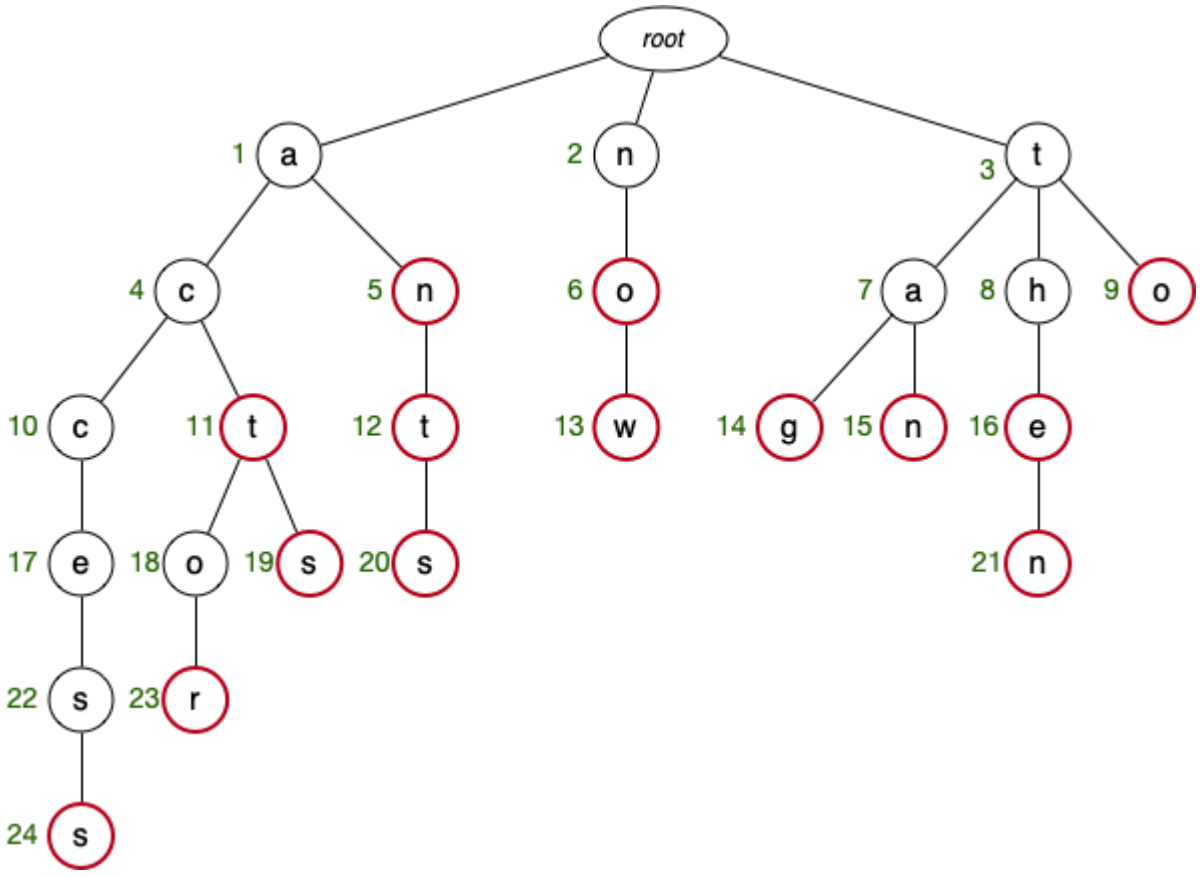
```
Heap h;  Item it;
h = newHeap(10);
insert(h, 10);
insert(h, 5);
insert(h, 15);
insert(h, 3);
insert(h, 16);
insert(h, 13);
insert(h, 6);
it = delete(h);
insert(h, 2);
it = delete(h);
it = delete(h);
it = delete(h);
it = delete(h);
it = delete(h);
```

9. Repeat the above sequence of insertions and deletions, but this time draw the heap as a binary tree, with the highest value at the root and the values decreasing as you move down any branch.

   Reminder:

```
new  i10  i5  i15  i3  i16  i13  i6  d  i2  d  d  d  d  d
```

10. Consider the following trie:



Show which nodes would be visited when searching for each of the following words:

a. `now`
b. `the`
c. `ant`
d. `access`
e. `actor`
f. `action`
g. `actors`
h. `tang`
i. `a`

11. Under what circumstances would inserting a new word into a trie not result in adding any new nodes? What *would* be the effect on the tree of inserting the word?

12. Show the final trie resulting from inserting the following words into an initially empty trie.

```
so   boo   jaws   boon   boot   axe   jaw   boots   sore
```

    Does the order that the words are inserted affect the shape of the tree?

13. What are the two cases when deleting a word from a trie? How is each case handled?
