

CS 535 Homework 5

Due: October 14 (W), in class.

Undergrads, please answer questions 1, 2 and 4. Question 3 is a bonus. Grad students, please answer questions 2, 3 and 4. Question 1 is a bonus.

1. Suppose items with keys A N O T H E R X M P L in that order are inserted into an initially empty hash table using the hash function $11k \bmod N$ to map the k th letter of the alphabet into a table index. For each of the following, give the contents of the hash table:
 - (i) $N = 5$, and separate chaining is used to handle collisions.
 - (ii) $N = 13$, and linear probing is used to handle collisions.
2. Recall that the *load factor* of a hash table is equal to n/N where n is the number of items stored in the table and N is the size of the array. A common goal for implementing hash tables is to put a threshold on the load factor, say 0.75. Thus, if $N = 100$, for example, the number of items stored in the table must not exceed 75. Once the load factor is exceeded, the hash table is *resized*. That is, the current array is replaced by a larger one. You might ask – how large? The goal of this exercise is to compare two different policies for resizing by analyzing the amount of copying done when a total of n items are stored.

Policy 1: When A has to be resized, create an array A' whose size is k more than that of A , where k is a constant. Move the items stored in A to A' .

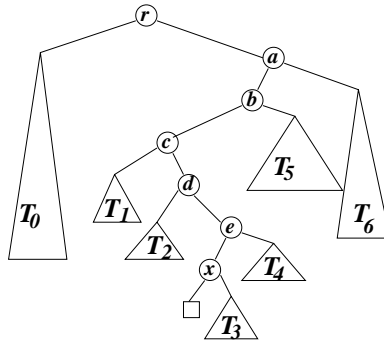
Policy 2: When A has to be resized, create an array A' whose is is twice that of A . Move the items stored in A to A' .

Suppose creating an array with a specified size takes $O(1)$ time, and “moving” an item from one array to another takes $O(1)$ time. Our goal is to determine the amount of time spent by each policy if n items are to be stored. For policies 1 and 2, please answer the following questions:

- (i) Assume that the initial size of A is 1. Suppose n items are to be stored. How many times did A have to be resized to contain the n elements? What is the final size of the array containing the n elements? For policy 1, your answer should be in terms of n and k . For policy 2, your answer should be in terms of n .
- (ii) Moving the elements of A to A' takes $O(|A|)$ time where $|A|$ denotes the size of A . From (i), we noted that A had to be resized multiple times, so elements had to be moved multiple times as well. For each policy, determine the *total time* spent copying elements. Express your answer in Big-Oh notation. Note that $2^{\log_2 n} = n$.
- (iii) Based on your answer in (ii), which policy will you choose? Why?

3. C-3.3

4. In the binary search tree below, assume that all the keys are distinct and that trees T_0, T_1, \dots, T_6 are non-empty. That is, there are items stored in these trees. A node is identified as x . Please answer the following questions *with explanations*.



- Describe the locations of the items with keys larger than $key[x]$.
- Describe the location of y such that $key[y]$ is the smallest key larger than x .
- Describe the location of z such that $key[z]$ is the largest key smaller than x .
- Based on the observations you've made in the answering question (a), given a binary search tree T whose keys are all distinct, describe an algorithm for implementing **removeGreaterThan**(k) – a method that removes all items with keys larger than k in $O(h + s)$ where h is the height of T and s is the size of the output.