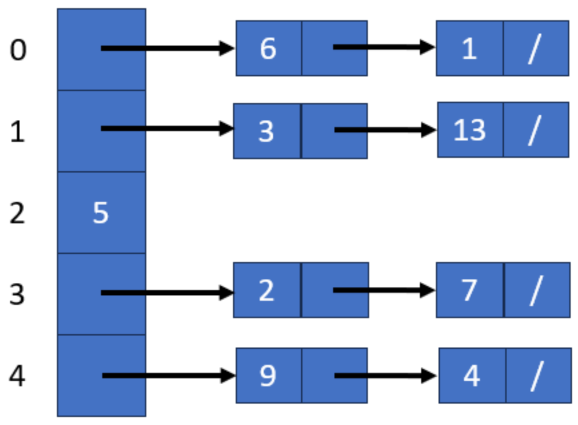
Recitation Problems - Com S 311

Week of Feb 12th - Feb 17th

1. Using a hash table size of 5, and a hash function $h(x) = (3x + 2)\%5$ for positive integers, draw the resulting hash table array when the integers $\mathbf{1, 7, 4, 6, 9, 13, 3, 2, 5}$ are added to the hash table in this order.

   (a) Use a chaining hash table that is not enlarged when elements are added. When there is a collision, add the new element at the head of the linked list at an index.



   (b) The load factor of a hash table T is the ratio of number of elements stored to the number of slots in the table. Compute the load factor of this table after adding all the elements.

   Load factor $\alpha = \frac{n}{m} = \frac{9}{5} = 1.8$, where n = number of elements = 9 and m = number of slots = 5.

2. Given an array A of integers, write an algorithm and its runtime to check if there are two elements whose sum is K. You must build a hash table T in the process of developing your algorithm and give an expected or average runtime.

Algorithm 1 Solution

```
1: for $i = 1$ to $n$ do
2:     if $K - A[i]$ is present in hash table $T$ then
3:         return true;
4:     else
5:         add $A[i]$ to hash table $T$
6: return false;
```

Searching in a hash table is expected $O(1)$ time. Adding to a hash table is $O(1)$ time. Therefore, the total runtime is expected $O(n)$ since the for loop iterates n times.

3. Given an array of distinct integers, write an algorithm using a hash table like in Problem 2 to check whether there are 4 distinct elements $x, y, z,$ and w such that $x + y = z + w$.

Algorithm 2 Solution

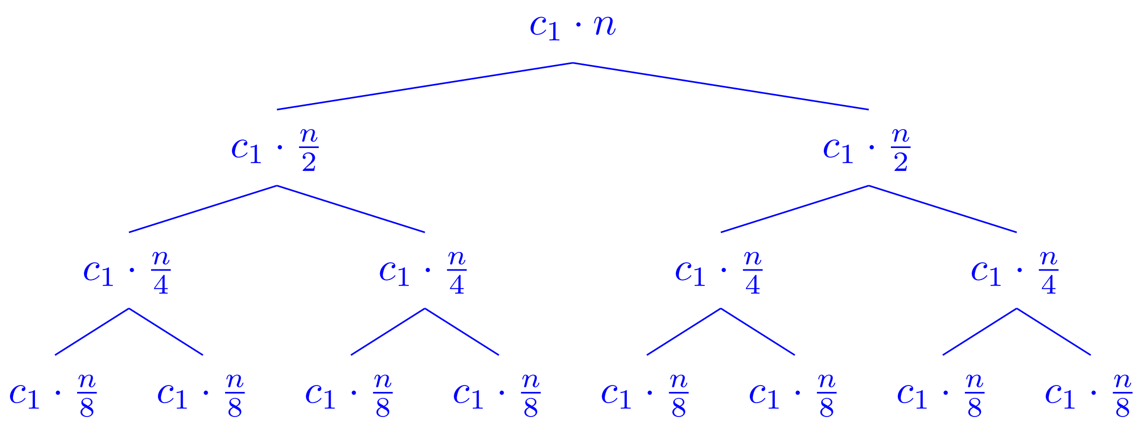
```
1: Setup a hash table $T$
2: for $i = 1$ to $n - 1$ do
3:     for $j = i + 1$ to $n$ do
4:         $v = A[i] + A[j]$;
5:         if $v$ is present in hash table $T$ then
6:             return true;
7:         else
8:             insert $v$ to hash table $T$
9: return false;
```

4. Consider the following recurrence equation to determine the runtime of Merge Sort.

$$T(n) = \begin{cases} 2 \cdot T(\frac{n}{2}) + c_1 \cdot n & \text{if } n \geq 2 \\ c_2 & \text{if } n < 2 \end{cases}$$

In the recurrence tree method to solve this recurrence equation, we draw a recurrence tree. The nodes in the recurrence tree correspond to various calls that are made during the course of the recursive algorithm.

In each node, we denote the amount of work done by that particular call (the merge step). Note, the $c_1 \cdot n$ above represents the amount of work to merge or combine. Draw the first four levels of this tree and label the work done in each node. The root node starts with:


$c_1 \cdot n$
$c_1 \cdot \frac{n}{2}$
$c_1 \cdot \frac{n}{2}$
$c_1 \cdot \frac{n}{4}$
$c_1 \cdot \frac{n}{4}$
$c_1 \cdot \frac{n}{4}$
$c_1 \cdot \frac{n}{4}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$
$c_1 \cdot \frac{n}{8}$