Kunal Patel

CSE 310 Recitation 5

Objectives:

1. Divide-and-Conquer algorithm design & analysis
2. Hash table

Instruction

1. For all recitation: **the solution should be clearly typed or written and must be saved in .pdf or .jpg format**. Note: **unreadable answer receives no credits!**
2. All recitation must be submitted through the link posted on Blackboard, we do NOT accept any hand-in submissions or submissions sent through emails!

Question

1. Given an integer array *A*, A[i] and A[j] are ***inverted*** if i < j, but A[i] > A[j]. Consider the following array with 10 elements. In it, 1 < 2 but (A[1] = 5) > (A[2] = 4), so 5-4 is an inversion. In this array there are a total of 15 inversions: 4-2, 4-3, 5-2, 5-3, 5-4, 8-2, 8-3, 8-6, 8-7, 9-2, 9-3, 9-6, 9-7, 9-8, and 10-7.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	**5**	**4**	**9**	**8**	**2**	**3**	**6**	**10**	**7**

(a) [3 pts] Write a divide-and-conquer algorithm to count the number of inversions in an integer array A of size *n*.

```
INVERSION(A, start, end)
{
  num = 0;
  if (start < end)
  { int m = (start + end)/2;
    INVERSION(A, start, end)
    INVERSION(A, m+1, end)
    num += MERGE(A, start, m, end);
  }
  return num;
}
```

sub1 = (mid - start)+1
sub2 = end - mid

```
MERGE(A, start, mid, end)
{
  while(i < length(sub1) && j < length(sub2))
  {
    if(L[i] <= R[j]) // left sub ≤ right sub
    { A[k] = L[i]; // not needed
      i++;
    }
    else
    {
      A[k] = R[j]; // not needed
      j++;
      count++;
    }
    k++;
  return count;
}
```

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

(b) [2 pts] Set up a recurrence relation for the run time of your algorithm in the worst case. Briefly explain how each term in the recurrence arises. Solve the recurrence using the Master Method. $T(n) = 2T(n/2) + \Theta(n)$ ← 2 recurrence calls of $n/2$ split

case 2: $f(n) = \Theta(n^c \log^k n)$ $c = \log_2 2 = 1$

$T(n) = \Theta(n \log n)$

2. [2 pts] Demonstrate what happened when we insert the keys 6, 29, 20, 16, 21, 34, 13, 18, 11 into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \bmod 9$. Draw the resulting hash table.

T	
18	0
	1
11, 20, 29	2
21	3
13	4
	5
6	6
34, 16	7
	8

$h(6) = 6 \bmod 9 = 6$
$h(29) = 29 \bmod 9 = 2$
$h(20) = 20 \bmod 9 = 2$ (collision)
$h(16) = 16 \bmod 9 = 7$
$h(21) = 21 \bmod 9 = 3$
$h(34) = 34 \bmod 9 = 7$ (collision)
$h(13) = 13 \bmod 9 = 4$
$h(18) = 18 \bmod 9 = 0$
$h(11) = 11 \bmod 9 = 2$ (collision)

3. Suppose you are given a universe of elements U = 85,46,65,34,39,98,17 to be inserted into a hash table and number of slots in the table is 5.

(a) [1 pt] What is the load factor?

$\frac{7}{5} = 1.4$

(b) [2 pts] To resolve collision using chaining method draw the final content of the hash table with hash function $h(k) = k \bmod 5$. How many computations at the most do you think you're required to search for any element in the final hash table.

T	
65, 85	0
46	1
17	2
98	3
39, 34	4

$h(85) = 85 \bmod 5 = 0$
$h(46) = 46 \bmod 5 = 1$
$h(65) = 65 \bmod 5 = 0$ (collision)
$h(34) = 34 \bmod 5 = 4$
$h(39) = 39 \bmod 5 = 4$ (collision)
$h(98) = 98 \bmod 5 = 3$
$h(17) = 17 \bmod 5 = 2$

since $n = 7$, worst case running time is $O(n)$. The running time is proportional to the length of the list, in worst case, which means in worst case you would need to search the whole list.