Data Structure and Algorithm
Homework #6

=== Homework Reference Solution ===

***Problem* 2.**

(1)

First, go through the array and put numbers in n clusters according to the number of digits. Since the total number of all digits is n, every number has at most n digits. This process takes O(n) steps. It's clearly that the number with more digits are higher then those with less digits. Therefore, we only have to sort within each cluster. Assume that there are k_i numbers with i digits; then we can use radix sort to sort these numbers digit by digit in time $O(ik_i)$ (i passes of Counting Sort, each with k_i numbers). Thus, together it will take $O(\Sigma\, i\, k_i)$ steps to sort all the clusters. $\Sigma\, i\, k_i$ is equal to total number of digits n. Thus, this algorithm can be run in time $O(n) + O(\Sigma\, i\, k_i) = O(n) + O(n) = O(n)$.

(2)

We use a counting sort to sort the words based on their first letter. Then, for each initial letter, we recursively sort the words with that first letter using the sort algorithm, but with the first letter of each word removed. If any one of these entries is just one letter long then we do not include it in the recursion but place it at the beginning of the results (when the recursion returns, place the first letter back on the front of each word). The base case of the recursion is when the set of words to sort is empty. From problem 2-1 , we know each counting sort call is O(k + 26) = O(k) when k words are sorted. Those, the algorithm should be run in O(n) time.

***Problem* 3.**

(1)

$\because 2^{pi} \equiv 1\ mod\ (2^p - 1), (i \geq 0)$

$$h(\boldsymbol{x}) = \left(\sum_{i=0}^{n-1} f(x_i) \times 2^{pi}\right) mod\ (2^p - 1) = \left(\sum_{i=0}^{n-1} f(x_i)\right) mod\ (2^p - 1)$$

Therefore, h(x) depends only on the summation of the characters.

(2)

initial	0	1	2	3	4	5	6	7	8	9	10
17							17				
33	33						17				
8	33						17		8		
36	33			36			17		8		
39	33			36		39	17		8		
31	33			36		39	17		8	31	
88	33			36		39	17	88	8	31	

***Problem* 4.**

(1)

The probing sequence is that $h(k)$, $h(k) + 1$, $h(k) + 1 + 2$, $h(k) + 1 + 2 + 3$, …, $h(k) + 1 + 2 + \ldots + i + \ldots$. We can write above sequence as the following form.

$h'(k,i) = \left(h(k) + \frac{i(i+1)}{2}\right) mod\ m = \left(h(k) + \frac{1}{2}i + \frac{1}{2}i^2\right) mod\ m$

Thus, $c_1 = \frac{1}{2}$ and $c_2 = \frac{1}{2}$.

(2)

To show this algorithm examines every table position in the worst case, we show that for a given key, each of the first m probes hashes to a distinct value.

Given any key k, we have to show that for any probe numbers i and j such that $0 \le i < j < m$, $h'(k,i) \neq h'(k,j)$. We do so by showing that $h'(k,i) = h'(k,j)$ yields a contradiction.

Assuming that there exists a key k and probe numbers i and j satisfying $0 \le i < j < m$, for which $h'(k,i) = h'(k,j)$. Then, $h(k) + \frac{i(i+1)}{2} \equiv h(k) + \frac{j(j+1)}{2}$ $(mod\ m)$, which in turn implies that $\frac{j(j+1)}{2} - \frac{i(i+1)}{2} \equiv 0$ (mod m)

$\therefore \frac{(j-i)(j+i+1)}{2} \equiv 0\ (mod\ m)$

We have $\frac{(j-i)(j+i+1)}{2} = rm$ for some integer $r \rightarrow (j-i)(j+i+1) = r * 2m$

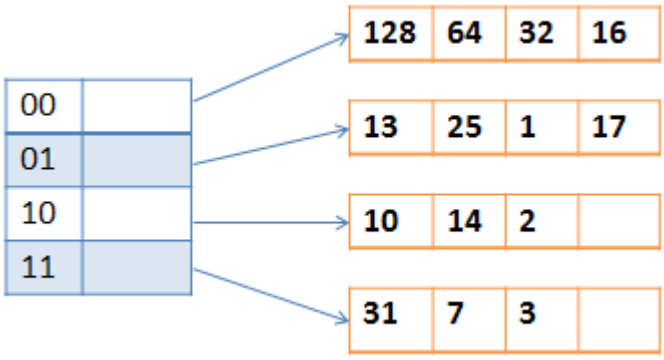
m is a power of 2, $\rightarrow (j-i)(j+i+1) = r * 2^{p+1}$ for $m = 2^p$ (p is a non-negative integer)

If (j - i) is even (j – i + 1) must be odd, and if (j - i) is odd (j – i + 1) must be even. Because $(j-i)(j+i+1) = r * 2^{p+1}$, it means that 2^{p+1} must divide one of (j - i) and (j – i + 1). Since (j - i) < m < 2^{p+1} , it cannot be (j - i). Since (j – i + 1) <= (m - 1) + (m - 2) + 1 = 2m - 2 < 2^{p+1}, it also cannot be (j – i + 1).

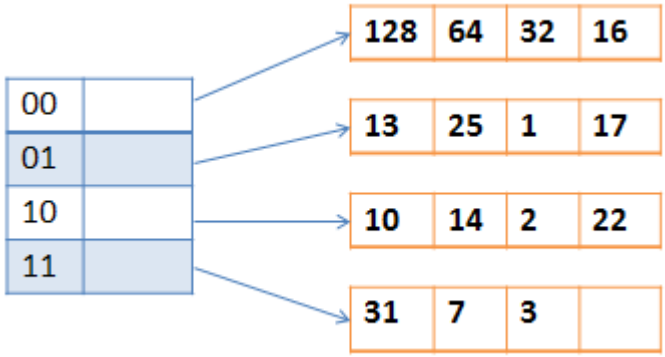
Thus we have derived a contradiction.

***Problem* 5.**

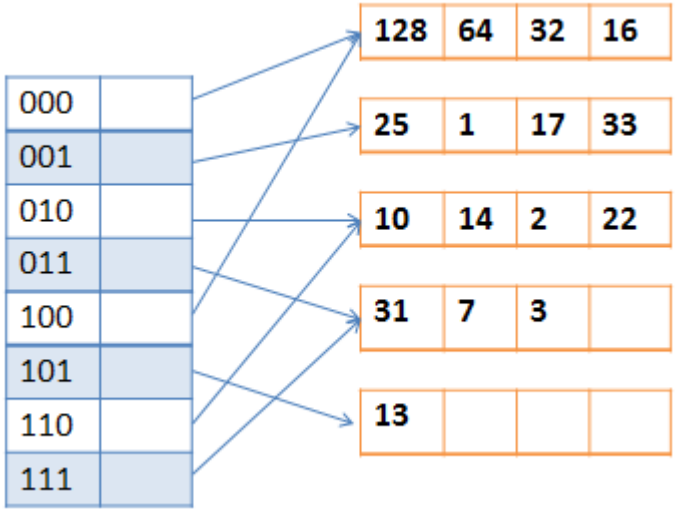
(1)



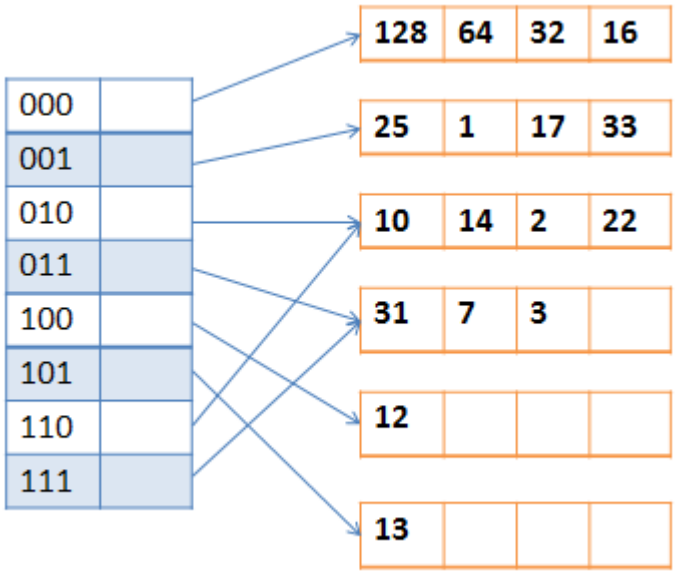
(2)



(3)



(4)


000
001
010
011
100
101
110
111
128 64 32 16
25 1 17 33
10 14 2 22
31 7 3
12
13