CS2321 Homework3

(115 points)

1. **Huffman Code (25 points)**
2. (5p) Generate a binary Huffman tree from the letter frequencies in the tongue twister: *she sells sea shells by the seashore*. Do include the space character (6 of them) in your tree, no need to include the period, and treat all letters as lower case.

Letter Frequency table

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| letter | S | h | e | space | l | a | b | y | t | o | r |
| freq |  |  |  |  |  |  |  |  |  |  |  |

1. (5p) Prefix Code tree with Huffman Code
2. (5p) Prefix Code word table

|  |  |  |  |
| --- | --- | --- | --- |
| letter | ASCII code in decimal | ASCII Code in Binary | Huffman code in binary |
| s |  |  |  |
| h |  |  |  |
| e |  |  |  |
| space |  |  |  |
| l |  |  |  |
| a |  |  |  |
| b |  |  |  |
| y |  |  |  |
| t |  |  |  |
| o |  |  |  |
| r |  |  |  |

1. (3p) Write the encoded data for the input “*she sells sea shells by the seashore “* with the above Huffman code
2. **(2p) Total number of bits after encoding** *she sells sea shells by the seashore* with Huffman code
3. (5p) How to decode the above binary code. Describe your idea using clear logic.
4. (30 p) Merge Sort Review

**Algorithm** Merge(A1, A2, A)

**Input**: Array A1 and A2 are sorted, each element is comparable

**Output**: A is sorted with data from A1 and A2

// Variable i points to the next element in A1, starting at 0

// Variable j points to the next element in A2, starting at 0

// Variable k points to the next element in A, starting at 0

// Repeat the following as long as both i and j are in the valid index range

/\* copy the smaller of A1[i] and A2[j] to A[k],

\* increase (i or j) and k accordingly

\*/

// Repeatedly copy the remaining elements in A1 to A, starting at index i

// Repeatedly copy the remaining elements in A2 to A, starting at index j

**Algorithm** MergeSort(A)

**Input**: array A, each element is comparable

**Output: A is sorted**

//Base case: if the size of A is less than 2, return

//copy the first half of A to A1, the second half of A to A2.

//call MergeSort to sort A1

//call MergeSort to sort A2

//call Merge() to merge sorted A1 and A2 into A

Illustrate the execution of the merge sort algorithm with the following array. Follow the example on **Text book page 533** by showing merge-sort tree of input sequence and output sequences.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 3 | 5 | 1 | 4 | 10 | 2 | 8 | 6 | 7 |

1. (30 p) Quick Sort Review

**Algorithm** Partition (A,p,r)

**Input**: Array A, p and r specify the range A[p..r]

**Output**: Use Hoare partition scheme to partition the data. Use last element as pivot. Return index value q, such that each A[p..q-1] <= A[q] <= each A[q+1 .. r]

//set the left pointer i to be p

//set the right pointer j to be r-1

//repeat the following if point i and j has not crossed each other

/\*continue move i to its right as long as i and j has not crossed

\*each other and A[i] is less than the pivot

**\*/**

/\*continue move j to its left as long as i and j has not crossed

\*each other and A[j] is greater than the pivot

**\*/**

/\*if i and j has not crossed, then swap A[i] with A[j],

\*move i forward and j backward.

**\*/**

//put pivot to the right spot: swap A[i] and A[r]

//return the partition point i

**Algorithm** QuickSort(A, p, r)

**Input**: Array A, p and r specify the subarray A[p..r]

**Output**: A[p..r] is sorted

//if there are two or more elements in the subarray; that is if p is less than r

//call partition method to divide the data

//call QucikSort to sort the data part that is to the left of the pivot

//call QucikSort to sort the data part that is to the left of the pivot

Illustrate the execution of the quick sort algorithm for the following data array by showing the input sequence and output sequence as **text book page 545**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 3 | 5 | 1 | 4 | 10 | 2 | 8 | 6 | 7 |

1. (20 points) Answer the following questions related to sorting

* What is the lower bound of comparison based sorting algorithm?
* What is stable sorting property?
* Which of the following sorting algorithms are stable?

Insertion sort, selection sort, merge sort, quick sort

* What is the worst case running time of bucket sort in terms of input size ***n*** and value range ***0.. N-1***?
* What is the worst case running time of radix sort in terms of input size ***n***, value range ***0.. N-1*** for each dimension, and total dimension number ***d***?
* How to determine is the size of bucket array?
* What is the **data structure** of each bucket?
* How to figure the data range given an array to be sorted?
* How to figure out how many dimensions to run for Radix sort?
* Give a number, say 9876, write code to figure out the digit in each place, that is in 1’s, 10’s, 100’s, 1000’s?

1. (10 points) Bucket Sort and Radix Sort Algorithm

Use radix sort to illustrate the sorting steps for 468,537,9,721,13,35,123,27. Show the result after sorting by each digit.