

Name: **Kashif Ali**
Roll No: **20P-0648**
Section: **3D**

Lab-9 tasks
Huffman Coding

```

C 01_huffman.cpp
1 // Huffman Coding in C++
2
3 #include <iostream>
4 using namespace std;
5
6 #define MAX_TREE_HT 50
7
8 class MinHeapeapNode {
9     public:
10     int freq;
11     char item;
12     MinHeapeapNode *left, *right;
13 };
14
15 class MinHeap {
16     public:
17     int size;
18     int capacity;
19     MinHeapeapNode **array;
20 };
21
22 // Creating Huffman tree node
23 MinHeapeapNode *newNode(char item, int freq) {
24     struct MinHeapeapNode *temp = new MinHeapeapNode;
25
26     temp->left = temp->right = NULL;
27     temp->item = item;
28     temp->freq = freq;
29
30     return temp;
31 }
32
33 // Create min heap using given capacity
34 MinHeap *createMinHeap(int capacity) {
35     MinHeap *MinHeapeap = new MinHeap;
36     MinHeapeap->size = 0;
37     MinHeapeap->capacity = capacity;
38     MinHeapeap->array = (struct MinHeapeapNode **)malloc(MinHeapeap->capacity * sizeof(struct MinHeapeapNode *));
39     return MinHeapeap;
40 }

```

Task-1

Huffman coding part-a

```

42 // Print the array
43 void printArray(int arr[], int n) {
44     int i;
45     for (i = 0; i < n; ++i)
46         cout << arr[i];
47
48     cout << "\n";
49 }
50
51 // Swap function
52 void swapMinHeapeapNode(MinHeapeapNode **a, MinHeapeapNode **b) {
53     MinHeapeapNode *t = *a;
54     *a = *b;
55     *b = t;
56 }
57
58 // Heapify
59 void MinHeapeapify(MinHeap *MinHeapeap, int idx) {
60     int smallest = idx;
61     int left = 2 * idx + 1;
62     int right = 2 * idx + 2;
63
64     if (left < MinHeapeap->size && MinHeapeap->array[left]->freq < MinHeapeap->array[smallest]->freq)
65         smallest = left;
66
67     if (right < MinHeapeap->size && MinHeapeap->array[right]->freq < MinHeapeap->array[smallest]->freq)
68         smallest = right;
69
70     if (smallest != idx) {
71         swapMinHeapeapNode(&MinHeapeap->array[smallest],
72                             &MinHeapeap->array[idx]);
73         MinHeapeapify(MinHeapeap, smallest);
74     }
75 }
76
77 // Check if size is 1
78 int checkSizeOne(MinHeap *MinHeapeap) {
79     return (MinHeapeap->size == 1);
80 }

```

Task-1

Huffman coding part-b

```

82 // Extract the min
83 MinHeapeapNode *extractMin(MinHeap *MinHeapeap) {
84     MinHeapeapNode *temp = MinHeapeap->array[0];
85     MinHeapeap->array[0] = MinHeapeap->array[MinHeapeap->size - 1];
86
87     --MinHeapeap->size;
88     MinHeapeapify(MinHeapeap, 0);
89
90     return temp;
91 }
92
93 // Insertion
94 void insertMinHeapeap(MinHeap *MinHeapeap, MinHeapeapNode *MinHeapeapNode) {
95     ++MinHeapeap->size;
96     int i = MinHeapeap->size - 1;
97
98     while (i && MinHeapeapNode->freq < MinHeapeap->array[(i - 1) / 2]->freq) {
99         MinHeapeap->array[i] = MinHeapeap->array[(i - 1) / 2];
100         i = (i - 1) / 2;
101     }
102
103     MinHeapeap->array[i] = MinHeapeapNode;
104 }
105
106 // BUild min heap
107 void buildMinHeapeap(MinHeap *MinHeapeap) {
108     int n = MinHeapeap->size - 1;
109     int i;
110
111     for (i = (n - 1) / 2; i >= 0; --i)
112         MinHeapeapify(MinHeapeap, i);
113 }
114
115 int isLeaf(MinHeapeapNode *root) {
116     return !(root->left) && !(root->right);
117 }
118
119 MinHeap *createAndBuildMinHeapeap(char item[], int freq[], int size) {
120     MinHeap *MinHeapeap = createMinHeap(size);

```

Task-1

Huffman coding

part-c

```

119 MinHeap *createAndBuildMinHeapeap(char item[], int freq[], int size) {
120     MinHeap *MinHeapeap = createMinHeap(size);
121
122     for (int i = 0; i < size; ++i)
123         MinHeapeap->array[i] = newNode(item[i], freq[i]);
124
125     MinHeapeap->size = size;
126     buildMinHeapeap(MinHeapeap);
127
128     return MinHeapeap;
129 }
130
131 MinHeapeapNode *buildHfTree(char item[], int freq[], int size) {
132     MinHeapeapNode *left, *right, *top;
133     MinHeap *MinHeapeap = createAndBuildMinHeapeap(item, freq, size);
134
135     while (!checkSizeOne(MinHeapeap)) {
136         left = extractMin(MinHeapeap);
137         right = extractMin(MinHeapeap);
138
139         top = newNode('$', left->freq + right->freq);
140
141         top->left = left;
142         top->right = right;
143
144         insertMinHeapeap(MinHeapeap, top);
145     }
146     return extractMin(MinHeapeap);
147 }
148 void printHCodes(MinHeapeapNode *root, int arr[], int top) {
149     if (root->left) {
150         arr[top] = 0;
151         printHCodes(root->left, arr, top + 1);
152     }
153
154     if (root->right) {
155         arr[top] = 1;
156         printHCodes(root->right, arr, top + 1);
157     }

```

Task-1

Huffman coding part-d

```

157     }
158     if (isLeaf(root)) {
159         cout << root->item << " | ";
160         printArray(arr, top);
161     }
162 }
163
164 // Wrapper function
165 void HuffmanCodes(char item[], int freq[], int size) {
166     MinHeapeapNode *root = buildHfTree(item, freq, size);
167
168     int arr[MAX_TREE_HT], top = 0;
169
170     printHCodes(root, arr, top);
171 }
172
173 int main() {
174     char arr[] = {'A', 'B', 'C', 'D'};
175     int freq[] = {5, 1, 6, 3};
176
177     int size = sizeof(arr) / sizeof(arr[0]);
178
179     cout << "Char | Huffman code ";
180     cout << "\n-----\n";
181     HuffmanCodes(arr, freq, size);
182 }

```

Task-1

Huffman coding part-e

```
kashiii@kashiii: ~/Documents
kashiii@kashiii:~/Documents$ ./01_huffman.exe
Char | Huffman code
-----
C | 0
B | 100
D | 101
A | 11
kashiii@kashiii:~/Documents$ |
```

Task-1
Huffman output-1


```

173 int main() {
174     char arr[] = {'k', 'a', 's', 'h', 'i', 'f'};
175     int freq[] = {5, 1, 6, 3};
176
177     int size = sizeof(arr) / sizeof(arr[0]);
178
179     cout << "Char | Huffman code ";
180     cout << "\n-----\n";
181     HuffmanCodes(arr, freq, size);
182 }

```

Task-1 Huffman output-2

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
kashiii@kashiii:~/Documents$ g++ 01_huffman.cpp -o 01_huffman.exe
```

```
kashiii@kashiii:~/Documents$ ./01_huffman.exe
```

```
Char | Huffman code
```

```
-----
```

```

s | 000
a | 00100
h | 00101
k | 0011
i | 01
f | 1

```

```
kashiii@kashiii:~/Documents$
```



```

173 int main() {
174     char arr[] = {'F', 'A', 'S', 'T'};
175     int freq[] = {5, 1};
176
177     int size = sizeof(arr) / sizeof(arr[0]);
178
179     cout << "Char | Huffman code ";
180     cout << "\n-----\n";
181     HuffmanCodes(arr, freq, size);
182 }

```

Task-1 Huffman output-3

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

kashiii@kashiii:~/Documents$ g++ 01_huffman.cpp -o 01_huffman.exe
kashiii@kashiii:~/Documents$ ./01_huffman.exe
Char | Huffman code
-----
A | 000
F | 001
T | 01
S | 1
kashiii@kashiii:~/Documents$

```

"FAST" as input and the frequency is "5" and "1"

Thank You

...