山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名： 张文浩 | | 班级： 19级人工智能 |
| 实验题目：堆及其应用 | | | |
| 实验学时：2 | | 实验日期： 10.4 | |
| 实验目的：  1、掌握堆结构的定义、描述方法、操作定义及实现。  2、掌握堆结构的应用。 | | | |
| 软件开发工具：  Vscode | | | |
| 1. 实验内容   **题目描述**：哈夫曼编码。  **输入输出格式：**  **输入：**  一串小写字母组成的字符串（不超过1000000)。  **输出：**  输出这个字符串通过Huffman编码后的长度。   1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）   第二题要求输出哈夫曼编码后的长度，利用STL的类priority\_queue实现大根堆，利用一个map<char,int>存储字符及其出现的频率，然后将频率存入优先队列中，每次取出优先队列两个最小的频率，进行相加，合并树的操作，并将加后的和再次入队，直到队中只剩下一个元素。   1. 测试结果（测试输入，测试输出）     字符串abbcccdddd的哈弗曼编码长度为19   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   通过本次实验，我掌握了堆结构的定义、描述方法、操作定义及实现。在第二题中，我学习了关系哈弗曼编码的相关知识，掌握了堆结构的应用。  在构造堆的时候，应当注意排序的操作是按照层数进行的。每一次删除或者是初始化二叉树的时候，应该对其进行排序，将所有的元素中最小的那个放置到堆顶。这样才可以保证排序后的输出能够按照输出堆顶元素、弹出堆顶元素的顺序循环进行完成。如果不确定哪些时候需要排序，可以在每次更改堆的结构时更新顺序，也就是一直维护这个最小堆的结构，从而保证无论何时都可以得到由小到大的元素排列。  在第二个题目中，要求计算出霍夫曼编码的长度，也就是各个叶节点权值与路径长度乘积之和，同时这个值等于非叶节点之和。根据这个特性，无需完整地建立霍夫曼树，只需要用数组存储每个字符出现的频次，同时将其按照频次由小到大排序，取出最小的两个元素便可获得所有非叶节点包含的权重。这个思路在解决该问题时尤为重要，它让我避免了许多无用的步骤。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   #include <bits/stdc++.h>  using namespace std;  int ans;  string s;  map<char, int> mp;                                //每个字符出现的频率  priority\_queue<int, vector<int>, greater<int>> q; //存储字符权重(频率)的优先队列  int main()  {      cin >> s;      for (int i = 0; i < s.size(); i++)      {          mp[s[i]]++;      }      for (auto iter = mp.begin(); iter != mp.end(); iter++)      {          q.push(iter->second);      }      while (q.size() > 1)      {          int x, y, z;          x = q.top();          q.pop();          y = q.top();          q.pop();          z = x + y;          q.push(z);          ans += z;      }      cout << ans << endl;      system("pause");      return 0;  } | | | |