Huffman 编码 tsD14627

计 72 陈嘉杰

April 12, 2018

Contents

1 题目说明

文件 `input.txt` 中含有一个字符串，然后要把 Huffman 编码后文本的长度写入 `output.txt` 中。

2 实现思路

先统计词频，然后构造出哈夫曼树。由于这里只需要输出最后文本的长度，所以只需要对这个树进行遍历，对每个叶子结点进行统计即可得到答案。

3 程序编译环境

1. 操作系统：macOS
2. 编译器：LLVM/Clang 6.0.0

4 实现步骤

4.1 数据读入

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
scanf("%s", buffer);
len = strlen(buffer);
if (len == 1) {
  // only one char?
  // zero entropy here
  // no reasonable answer
  printf("1\n");
  return 0;
 }
```

首先把文本都保存到一个足够大的数组中。然后这里有一个有争议的地方：题目数据中有一个点，`input.txt` 中仅有一个字母 `a` 。在这种情况下，Huffman 树仅有一个叶结点。此时信息熵为 0 。我不是很确定这里应该写什么答案。还有就是，如果输入的文本是 `aaaa` ，此时信息熵仍然为 0 。那此时又如何编码呢。

4.2 建 Huffman 树

首先是数据结构：

```
struct Node {
  Node *left = NULL;
  Node *right = NULL;
  int weight = 0;
  int index = 0;

  bool operator()(const Node *a, const Node *b) {
    return a->weight > b->weight;
  }
};
```

很直观的一个二叉树，`weight` 保存子树的词频，`index` 表示叶结点对应的字母，下面的 `operator ()` 是提供给 `priority_queue` 的比较器。

接下来，统计词频，建树：

```
for (int i = 0; i < len; i++) {
  freq[buffer[i] - 'a']++;
 }
for (int i = 0; i < 26; i++) {
  if (freq[i]) {
    Node *node = new Node;
    node->weight = freq[i];
    node->index = i;
    pq.push(node);
  }
 }
while (pq.size() > 1) {
  Node *first = pq.top();
  pq.pop();

  Node *second = pq.top();
  pq.pop();

  Node *new_node = new Node;
  new_node->left = first;
  new_node->right = second;
  new_node->weight = first->weight + second->weight;
  pq.push(new_node);
}
```

这就是很常规的建立 Huffman 树的方法。

4.3 统计结果

接下来，遍历这棵树，然后统计结果：

```
int calc_length(Node *current, int cur_depth) {
  if (current->left == NULL && current->right == NULL) {
    return freq[current->index] * cur_depth;
  }
  return calc_length(current->left, cur_depth + 1) +
         calc_length(current->right, cur_depth + 1);
}

int main() {
```

```
  // ..snip..
  root = pq.top();
  printf("%d\n", calc_length(root, 0));
  return 0;
}
```

对于字母表中的每个字母，它的高度就是 Huffman 编码中的长度，乘以词频求和即是答案。最后输出即可。

5 遇到的问题和得到的收获

遇到的问题是，第一，`priority_queue` 的默认比较，由于我这里用的是 `Node *` 类型，所以会变成指针比较。所以需要自己写一个 `comparator` ，但是全局的 `operator <` 又不支持指针。所以复用了 `struct Node` 作为比较器，这样就可以了。第二就是，输入的文本只有一个字母，这个我觉得没有一个很好的答案。为了过 OJ ，只写了一个小小的判断，留下我的疑问。