作业 PA7 实验报告

姓名：钱子贤 学号：2054170 日期：2021 年 12 月 28 日

实验报告格式按照模板来即可，对字体大小、缩进、颜色等不做要求

实验报告要求在文字简洁的同时将内容表示清楚

1. 涉及数据结构和相关背景

字典树（又叫单词查找树、TrieTree），是一种树形结构，典型应用是用于统计，排序和保存大量的字符串（但不仅限于字符串）。主要思想是利用字符串的公共前缀来节约存储空间。很好地利用了串的公共前缀，节约了存储空间。字典树主要包含两种操作，插入和查找
是一种哈希树的变种,常用于,统计,排序,保存大量字符串(但不仅限于字符串),主要实现方法是利用串的公共前缀来减少查询时间，减少了不必要的比较,不仅节约了存储空间，而且检索的效率比哈希表要高。

2. 实验内容

建树操作;

```
const int maxn = 100;//提前估计好可能会开的节点的个数
int tot;              //节点编号，模拟申请新节点，静态申请
int trie[100][4]; //假设每个节点的分支有4个(分别代表历史表现数据（速度、火力）和目前物资储备（食物、燃料））
bool vis[100];      //判断该节点的分支类型,也可以开int纪录出现次数
```

数组方式实现

要写代码实现一个Trie首先就要确定如何存储一个Trie结构。这里用一个二维数组来存储：

```
int trie[MAX_NODE][CHARSET];
int tot;
```

其中MAX_NODE是trie中最大能存储的节点数目，CHARSET是字符集的大小，k是当前trie中包含有多少个节点。Trie[i][j]的值是0表示trie树中i号节点，并没有一条连出去的边，满足边上的字符标识是字符集中第j个字符（从0开始）；trie[i][j]的值是正整数x表示trie树中i号节点，有一条连出去的边，满足边上的字符标识是字符集中第j个字符，并且这条边的终点是x号节点。

插入操作

```
void insert(char* s, int rt)//参数是字符串和节点数,建立字典树
{
    for (int i = 0; s[i]; i++)
    {
        int x = s[i] ;
        if (trie[rt][x] == 0) //若不存在该节点，开新节点
```

```
        {
            trie[rt][x] = ++tot;//表示字符的编号
        }
        rt = trie[rt][x];    //代表该字符在rt层节点
    }
    vis[rt] = true;//整个字符串读完后，在vis数组中记录第rt层为单词结尾
}
```

查询

```
bool find(char* s, int rt)
{
    for (int i = 0; s[i]; i++)
    {
        int x = s[i] ;
        if (trie[rt][x] == 0)
        {
            return false;//节点不存在，说明不存在，直接返回
        }
        rt = trie[rt][x];
    }
    return vis[rt];  //如果是被标记的，则说明该串在树中
}
```

初始化

```
tot = 0;//一开始没有节点
int rt = ++tot;//申请一个根节点
memset(trie[rt], 0, sizeof(trie[rt]));//初始化根节点
memset(isw, false, sizeof(isw));//初期化标记数组
```

删除

```
int dealTrie(Trie* T)
{
    if (T == NULL)
       return 0;
    for (int i = 0; i < Max; i++)
       if (T->next[i] != NULL)
          dealTrie(T->next[i]);
    free(T);
    return 0;
```

其余则按照正常的输入输出模型，排序算法去做，本题难点在于字典树的建立，而不是输入输出等其他的格式的模型，所以必须先弄明白字典树的含义。

实验结果：

```
请输入飞船的参数、数值和权重：
speed w=6
ship1 10
ship2 6
ship3 7
ship4 8
ship5 7
fire w=4
ship1 7
ship2 10
ship3 9
ship4 8
ship5 7
food w=8
ship1 6
ship2 7
ship3 8
ship4 8
ship5 10
fuel w=9
ship1 7
ship2 6
ship3 9
ship4 8
ship5 8

结果从大到小排序为：
      speed fire  food  fuel  total
ship3 8     9     8     9     229
ship5 7     7     10    8     222
ship4 8     8     8     8     216
ship1 10    7     6     7     199
ship2 6     10    7     6     186
最优选择为：
ship3 total=229
```

3. 实验总结

字典树的优点

节约空间

例如在现实生活中，我们需要存储很多 URL（Uniform Resoure Locator：统

一资源定位器）是 WWW 页的地址。

大多数以“http://www ”开头，若以字典树存储，则 50 亿个“http://www ”只需要 10 个存储单位即可，而非 50 亿*10 个。

快速检索

字典树能很好地利用串的公共前缀，节约了存储空间，同时用它来检索同样有着比较高的效率。

我们需要将一个单词列表建出一棵单词查找树，满足：

（1）根结点不包含字母，除根结点外的每个都仅含一个大写英文字母；

（2）从根结点到某一节点，路径上经过的字母依次连起来所构成的单词，称为该结点对应的单词。单词列表中的每个词，都是该单词查找树某个结点所对应的单词；

（3）在满足上述条件下，该单词查找树的结点数最少，统计出该单词查找树的结点数目。

若在查询过程中使用字典树进行查找。依然对每个单词进行遍历。然后将 n 个单词分别拆开，两段分开的单词进行字典树查询。

如果两段单词都在字典树中查询成功说明被查询单词即其中一个 ans。

对字典树进行查询时，无须遍历 n 个单词，只用对被查询单词在树上走出 len 的查询次数。

字典树还有很多除了字符串之外的应用，如对数字的二进制 01 串构造字典树可以快速按位查询数字是否存在。当结合字符串匹配算法 KMP 时，利用 KMP 的利用失配信息思想快速继续匹配，可以实现 AC 自动机算法。只需给每个节点增加 fail 指针指向失配时快速指向的节点。

当对构造好的字典树进行 BFS 预处理并增加直达叶子节点的指针时，可以解决输入最小操作数的问题。

总之以字典树为基础可以改造出各种有用而高效的处理方法。