**排序的应用实验报告**

姓名：陆明奇 学号：2050283 日期：2022年1月1日

1. **涉及数据结构和相关背景**

字典树又称键树，是一棵度大于等于2的树。树中的每个结点中不是包含一个或几个关键字，而是只含有组成关键字的符号。例如，若关键字是数值，则结点中只包含一个数位；若关键字是单词，则结点中只包含一个字母字符。这种树会给某种类型关键字的表的查找带来方便。

**2. 实验内容**

**2.1 问题描述**

明天，人类舰队就要迎接三体舰队的探测器——水滴了，作为增援未来部队的你（章北海）刚从“冬眠”中苏醒，就立刻思考起了保留人类文明火种的重大计划，逃亡！为了增加逃亡的成功率，你用1秒钟快速了解了所有战舰的历史表现数据（速度、火力）和目前物资储备（食物、燃料），并选出了一艘最合适的战舰。

**2.2 基本要求**

输入数据：

1. 输入包含若干张表，每一张表表示部分战舰在某个方面的数据表中有若干键值对，键为战舰的名字，值为该战舰的一项数据。输入战舰顺序不确定

2. 可以认为每一张表都表示不同的数据，且同一张表中一艘战舰只会出现一次，且每张表中都包含所有战舰的这一个数据

3. 每一项数据都有一个权值，表示这项数据的重要程度。假设表i中记录战舰j的数据为tij，表的权值为wi，那么这艘战舰的总分数计算方法为cj=t1j\*w1+t2j\*w2+ ... + tnj\*wn

4. 要求使用字典树作为基础数据结构

输出数据：

1. 输出一张大表，表的行为战舰名字，列为不同的数据，并打印总分

2. 要求所有战舰按总分从大到小排列，对列的顺序没有要求

**2.3 问题分析和解决思路**

题目要求使用字典树作为存储结构，由此我想到将每艘飞船的名字以字典树的形式存储，字典树中每个结点存储飞船名字中的一个字母。这样可以快速查找到每艘飞船的信息。我先对飞船名字中可能出现的所有字母进行排序，这样就可以用数组下标来表示这些字母。我建立了一个指针数组，数组中的元素为指向存储某一个字母的结点的指针。我在字典树的每个结点中都设置了一个指向该指针数组的二级指针。这样子便实现了单个字母按次序在树中的存储。此外，我将飞船的所有信息（所有数据和权重）都存储在一个结构体中（ship），字典树中的叶子结点中存有该结构体，表示字典树中一艘飞船的全部信息。关于从大到小输出战舰总分，只需要对ship结构体中的total进行降序排序就行了。

**2.4 数据结构设计**

typedef struct ship//飞船信息

{

float total;//总值

float speed;//速度

float speed\_w;//速度权值

float fire;//火力

float fire\_w;//火力权值

float food;//食物

float food\_w;//食物权值

float fuel;//燃料

float fuel\_w;//燃料权值

}ship;

typedef struct ship\_order//单个飞船信息

{

string name;

ship\* info;

}ship\_order;

typedef struct TrieNode//字典树结点

{

bool IsEnd;//是否为字典树的叶子结点

ship\* node;

TrieNode\*\* map;

TrieNode() {

IsEnd = false;

node = NULL;

map = new TrieNode \* [63]{}; //26个字母与0~9与下划线'\_'，共63个

}

}TrieNode;

//字典树

class Trie

{

public:

Trie();

TrieNode\* read(string name);

void insert(string name);

void init();

void set(ship\*& node, char\* kind, float data, float w);

void order();

void display();

private:

int num;

TrieNode\* root;

ship\_order\* ship\_name;

};

**2.4功能说明（函数、类）**

int find\_pos(char c)

{

if (c >= 'A' && c <= 'z') {

return int(c - 'A');

}

else if (c >= '0' && c <= '9') {

return int(c - '0' + 52);

}

else {

return 62;

}

}

功能：找到对应元素的位置

Para c：要查找位置的元素

返回值：元素所在的位置

Trie::Trie()

{

num = 0;

root = new(nothrow)TrieNode;

ship\_name = NULL;

}

功能：构造函数

返回值：无返回值

void Trie::set(ship\*& node, char\* kind, float data, float w)

{

if (!strcmp(kind, "speed")) {

node->speed = data;

node->speed\_w = w;

}

else if (!strcmp(kind, "fire")) {

node->fire = data;

node->fire\_w = w;

}

else if (!strcmp(kind, "food")) {

node->food = data;

node->food\_w = w;

}

else {

node->fuel = data;

node->fuel\_w = w;

}

}

功能：给飞船的某一数据种类赋值和权重

Para node：指向ship结构体的指针

Para kind: 数据的种类

Para data：该数据类型的值

Para w：该数据类型的权重

返回值：无返回值

TrieNode\* Trie::read(string name)

{

TrieNode\* p = root;

for (int i = 0; i < name.size(); i++) {

int pos = find\_pos(name[i]);

if (i == name.size() - 1) {

if (p->map[pos]->IsEnd == true) {

return p->map[pos];

}

else {

return NULL;

}

}

if (p->map[pos] == NULL) {

return NULL;

}

p = p->map[pos];

}

}

功能：找到存储飞船名字最后一个字母的结点的地址

Para name：飞船的名字

返回值：存储飞船名字最后一个字母的结点的地址

void Trie::insert(string name)

{

TrieNode\* p = root;

for (int i = 0; i < name.size(); i++) {

int pos = find\_pos(name[i]);

if (p->map[pos] == NULL) {

TrieNode\* node = new(nothrow)TrieNode;

p->map[pos] = node;

}

if (i == name.size() - 1) {

p->map[pos]->IsEnd = true;

}

else {

p = p->map[pos];

}

}

}

功能：将名字为name的飞船存储进字典树

Para name：飞船的名字

返回值：无返回值

void Trie::init()

{

char kind[6];

float w;

cout << "请输入飞船的数量" << endl;

cin >> num;

ship\_name = new(nothrow)ship\_order[num];

for (int i = 1; i <= 4; i++) {

cout << "请输入飞船的数据种类和权重" << endl;

cin >> kind;

cin >> w;

string name;

float data;

for (int j = 0; j < num; j++) {

cout << "请输入飞船的名字和数据" << endl;

cin >> name;

if (i == 1) {

ship\_name[j].name = name;

}

cin >> data;

TrieNode\* temp;

if (read(name) == NULL) {

insert(name);

temp = read(name);

temp->node = new(nothrow)ship;

}

else {

temp = read(name);

}

set(temp->node, kind, data, w);

if (i == 4) {

ship\* p = temp->node;

p->total = p->food \* p->food\_w + p->fire \* p->fire\_w + p->speed \* p->speed\_w + p->fuel \* p->fuel\_w;

}

}

}

}

功能：初始化字典树

返回值：无返回值

void Trie::order()

{

for (int i = 0; i < num; i++) {

ship\_name[i].info = read(ship\_name[i].name)->node;

}

int gap = num;

while (gap > 1) {

gap /= 2;

for (int i = 0; i < num - gap; i++) {

int end = i;

ship\_order temp = ship\_name[end + gap];

while (end >= 0) {

if (temp.info->total < ship\_name[end].info->total) {

ship\_name[end + gap] = ship\_name[end];

end -= gap;

}

else {

break;

}

}

ship\_name[end + gap] = temp;

}

}

}

功能：对字典树中存储的所有飞船的total进行排序

返回值：无返回值

void Trie::display()

{

cout << setiosflags(ios::right) << setw(15) << "name" << setw(10) << "speed" << setw(10) << "fire" << setw(10) << "food" << setw(10) << "fuel" << setw(10) << "total" << endl;

for (int i = num - 1; i >= 0; i--) {

cout << setw(15) << ship\_name[i].name

<< setw(10) << ship\_name[i].info->speed \* ship\_name[i].info->speed\_w

<< setw(10) << ship\_name[i].info->fire \* ship\_name[i].info->fire\_w

<< setw(10) << ship\_name[i].info->food \* ship\_name[i].info->food\_w

<< setw(10) << ship\_name[i].info->fuel \* ship\_name[i].info->fuel\_w

<< setw(10) << ship\_name[i].info->total

<< endl;

}

}

功能：输出一张大表，表的行为战舰名字，列为不同的数据（乘上权值），并

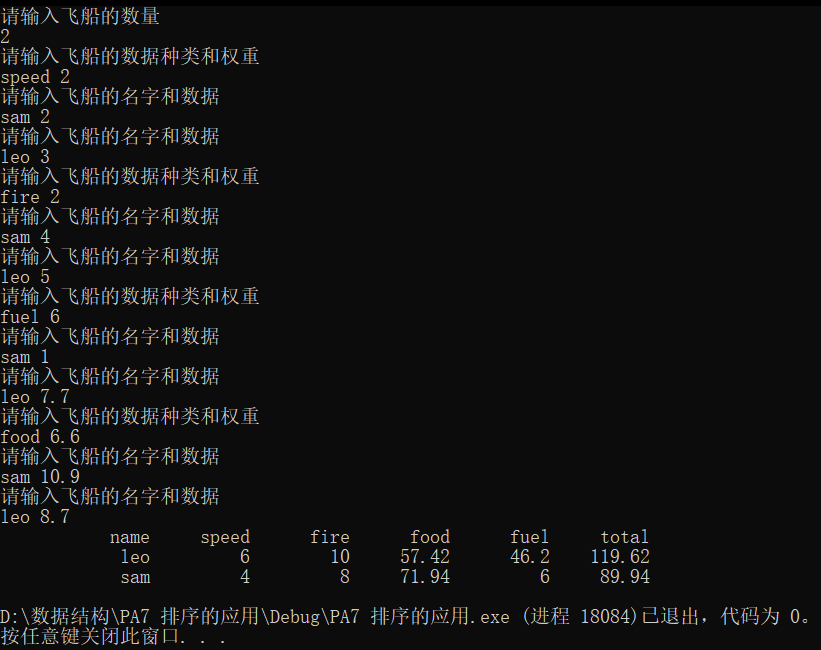
打印总分

返回值：无返回值

**2.5 调试分析（遇到的问题和解决方法）**

一开始程序会异常返回，通过单步调试，发现是由于字典树结点在构造时，map的值被赋为了NULL，没有给map指向的数组预先分配空间导致报错。之后加上了我自己定义的构造函数，问题得到解决。

（以下为程序运行截图）



**3.实验总结**

通过本次实验，我了解到了字典树的相关性质及其基本操作。熟悉了字典树在解决实际问题中的应用。它的优点是：利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较，查询效率比哈希树高。搜索字典树的方法：1) 从根结点开始依次搜索；(2) 取得要查找关键词的第一个字母，并根据该字母选择对应的子树并转到该子树继续进行检索；(3) 在相应的子树上，取得要查找关键词的第二个字母,并进一步选择对应的子树进行检索。(4) 迭代过程……

(5) 在某个结点处，关键词的所有字母已被取出，则读取附在该结点上的信息，即完成查找。