HW6实验报告

2252707 陈艺天

2024年3月1日

1. 排序

1.1 问题描述

使用不同排序算法进行测试, 总结各种算法特点.

1.2 问题分析与解决思路

使用自动化测试.

1.3 数据结构设计

本次所有排序均为基于向量的排序.

1.4 功能函数设计

自动化测试如下

```
constexpr int SEED = 29;

static void generate_sequence(std::vector<int> &t, const int n,
                              const bool reverse = false,
                              const bool sorted = false) {
  t.resize(n);
  if (sorted) {
    for (int i = 0; i < n; i++)
      t[i] = i;
  } else if (!reverse)
    for (int i = 0; i < n; i++)
      t[i] = rand();
  else
    for (int i = 0; i < n; i++)
      t[i] = n - i - 1;
}

class SortTest {
  using func_type = std::function<void(int *, int, int)>;
  func_type sort_func;

public:
  SortTest(func_type sort) : sort_func(sort) {}
  double runtest(const int n, const bool reverse = false,
                 const bool sorted = false) {
    srand(SEED);
    std::vector<int> test;
    generate_sequence(test, n, reverse, sorted);
    using namespace std::chrono;
    const auto begin = high_resolution_clock::now();
    sort_func(test.data(), 0, n);
    const auto end = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end - begin);
    double res = 0;
    std::cout << "Size : " << n
              << " Time : " << (res = duration.count() / 1000.0) << " ms"
              << std::endl;
    return res;
  }
};

static void basic_test(std::function<void(int *, int, int)> sort) {
  static auto print = [](const int elem) { std::cout << elem << ' '; };
```

```
  std::vector<int> test;
  srand(SEED);
  constexpr int N = 40;
  test.reserve(N);
  for (int i = 0; i < N; i++) {
    test.push_back(rand() % (3 * N));
  }

  std::for_each(test.begin(), test.end(), print);
  std::cout << '\n';
  sort(test.data(), 0, test.size());
  std::for_each(test.begin(), test.end(), print);
  std::cout << '\n';
}
```

冒泡排序

```
template <typename T>
void bubbleSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
  bool sorted = false;
  int n = high;
  while (!sorted) {
    sorted = true;
    for (int i = low; i < n - 1; i++) {
      if (elem[i] > elem[i + 1]) {
        sorted = false;
        std::swap(elem[i], elem[i + 1]);
      }
    }
    n--;
  }
}
```

选择排序

```
template <typename T>
void selectionSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
  auto find_max = [elem](const int low, const int high) {
    auto max_elem = elem[low];
    int max_pos = 0;
    for (int i = low; i < high; i++) {
      if (elem[i] > max_elem) {
        max_elem = elem[i];
        max_pos = i;
      }
    }
    return max_pos;
  };

  int n = high;
  while (--n > low) {
    std::swap(elem[n], elem[find_max(low, n)]);
  }
}
```

插入排序

```
template <typename T>
void insertionSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
  for (int i = low + 1; i < high; i++) {
    const auto target = elem[i];
    auto pos = std::upper_bound(elem + low, elem + i, target);
    for (int j = i; j > pos - elem; j--) {
      elem[j] = elem[j - 1];
    }
    *pos = target;
  }
}
```

归并排序

```
template <typename T>
static void merge(T *const elem, const int low, const int mid, const int high) {
  const int la = mid - low, lb = high - mid;
  T *const A = new T[la];
  T *const B = elem + mid;
  for (int i = 0; i < la; i++)
    A[i] = elem[i + low];

  int i = 0, j = 0, tot = low;
  while (i < la && j < lb) {
    if (A[i] <= B[j]) {
      elem[tot++] = A[i++];
    } else {
      elem[tot++] = B[j++];
    }
  }
  while (i < la)
    elem[tot++] = A[i++];
  delete[] A;
}

template <typename T>
void mergeSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
  const auto mid = (high + low) / 2;
  mergeSort(elem, low, mid);
  mergeSort(elem, mid, high);
  merge(elem, low, mid, high);
}
```

堆排序-排序

```
template <typename T>
void heapSort(T *const elem, const int low, const int high) {
  using namespace Heap;
  T *const A = elem + low;
  rank size = high - low;
  HeapBuild<T>::heapify(A, size);
  while (--size) {
    std::swap(A[0], A[size]);
    HeapBuild<T>::percolateDown(elem, 0, size);
  }
}
```

堆排序-建堆

```
//下滤算法
template <typename T>
rank HeapBuild<T>::percolateDown(T* const elem, const rank start,
                                 const rank n) {
  int i = start, j = -1;
  const auto target = elem[start];
  while (i != (j = maxIn3(elem, i, target, n))) {
    elem[i] = elem[j];
    i = j;
  }
  elem[j] = target;
  return j;
}
//Floyd建堆法
template <typename T>
void HeapBuild<T>::heapify(T* const elem, const rank n) {
  for (int i = Parent(n - 1); InHeap(i, n); i--) {
    percolateDown(elem, i, n);
  }
}
```

快速排序

```
//分割算法
template <typename T> static int partition(T *const elem, int low, int high) {
  std::swap(elem[low], elem[low + rand() % (high - low)]);
  const auto pivot = elem[low];
  while (low < high) {
    do
      high--;
    while (low < high && pivot <= elem[high]);
    if (low < high)
      elem[low] = elem[high];
    do
      low++;
    while (low < high && elem[low] < pivot);
    if (low < high)
      elem[high] = elem[low];
  }
  elem[high] = pivot;
  return high;
}

//快速排序 递归版
template <typename T>
void quickSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
  const auto mid = partition(elem, low, high);
  quickSort(elem, low, mid);
  quickSort(elem, mid + 1, high);
}
```

希尔排序

```
template <typename T>
void shellSort(T *const elem, const int low, const int high) {
  if (high - low < 2)
    return;
// 使用PS序列
  for (int d = INT_MAX; d > 0; d >>= 1) {
    // 步长为 d，矩阵宽度为d
    for (int j = low + d; j < high; j++) {
      const auto x = elem[j];
      int i{j};
      while (i - d >= low && elem[i - d] > x) {
        elem[i] = elem[i - d];
        i -= d;
      }
      elem[i] = x;
    }
  }
}
```

1.5 调试与分析

无且略.

1.6 总结

assert: 以下时间单位为ms

	10	100	1000	10000	100000	1000000	10k 逆序	10k正序
堆排序	0.001	0.008	0.142	1.992	19.658	227.235	1.107	1.155
归并排序	0.001	0.009	0.129	1.149	14.563	147.915	0.589	0.49
快速排序	0	0.005	0.071	0.714	9.343	100.241	0.384	0.348
希尔排序	0	0.005	0.088	1.144	18.911	229.019	0.28	0.178
插入排序	0	0.017	0.276	22.737	2180.79	223089	44.481	0.427
冒泡排序	0	0.006	0.364	32.323	3084.22	317779	42.283	54.273
选择排序	0	0.02	1.461	159.873	20732.3	>223089	164.606	0.013

复杂度分析

排序名称	时间复杂度	空间复杂度
堆排序	$\Theta(n \log n)$	原地
归并排序	$\Theta(n \log n)$	$O(n)$
快速排序	$O(n^2)$期望$O(n \log n)$	栈空间$O(\log n)$
希尔排序	PS序列可实现$O(n^{4/3})$输入敏感最优$O(n)$	原地
插入排序	$O(n^2)$输入敏感最优$O(kn)$, k为逆序对最大间距	原地
冒泡排序	$O(n^2)$	原地
选择排序	$O(n^2)$	原地

稳定性

排序名称	稳定性
堆排序	不稳定
归并排序	稳定
快速排序	不稳定
希尔排序	不稳定
插入排序	稳定
冒泡排序	稳定
选择排序	不稳定

2. 逆序对

2.1 问题描述

请求出整数序列A的所有逆序对个数

2.2 问题分析与解决思路

如果直接求逆序对,Brute-Force算法需要$O(n^2)$复杂度 .不足以通过本题. 因此需要更优秀的算法. 可以借助归并排序, 在排序中统计逆序对, 可以实现$\Theta(n \log n)$的复杂度.

2.3 数据结构设计

基于向量的归并排序.

2.4 功能函数设计

归并函数:

```
int mergeSortCount(const int low, const int high) {
  if (high - low < 2) return 0;
  int res = 0;
  const int mid = (low + high) / 2;
  res += mergeSortCount(low, mid);//分别计算
  res += mergeSortCount(mid, high);//分别计算

  res += mergeCount(low, mid, high);//向量归并
  return res; //返回逆序对
}
```

归并排序函数:

```
int mergeCount(const int low, const int mid, const int high) {
    int res = 0;
    int i = 0, j = mid, tot = low;
    const int la = mid - low;
    //这部分采用A作为公共缓冲区,避免多次的动态内存分配,提高效率
    if (A.size() < la) A.resize(la);
    for (int t = 0; t < la; t++) A[t] = elem[t + low];

    // 归并
    while (i < la && j < high) {
      if (A[i] <= elem[j]) {
        //如果满足左半部分小，那么不对逆序对有贡献
        elem[tot++] = A[i++];
      } else {
        elem[tot++] = elem[j++];
        //如果左半部分大
        //那么从这个元素开始后面的元素都会比右半段大
        //我们把逆序对的帐记到后者.
        res += mid - (i + low);
      }
    }
    //拷贝剩余部分元素
    while (i < la) {
      elem[tot++] = A[i++];
    }

    return res;
  }
```

2.5 调试与分析

`res += mid - (i + low);//这一步的贡献数目容易出错，因为从low开始计算还是从0开始计算,需要注意边界条件.`

2.6 总结

本题利用了分治的思想, 将原问题$T(n)$转化为子问题$2T(\frac{n}{2})$. 实现了复杂度的降低. 但同时缺陷在于空间复杂度提高为$\Theta(n)$.

3. 最大数

3.1 问题描述

给定一组非负整数 nums, 重新排列每个数的顺序（每个数不可拆分）使之组成一个最大的整数.

3.2 问题分析与解决思路

本题本质就是一个排序问题, 如何确定每个数的序. 进一步地说, 也就是给定a, b判断$a > b$的条件是什么? 经过分析可以发现只要保证$\{ab\} > \{ba\}$即可.

3.3 数据结构设计

基于向量的快速排序

3.4 功能函数设计

```
std::string largestNumber(std::vector<int>& nums) {
  // 这里填写你的代码
  std::string res;
  res.reserve(nums.size() * 4);
  //这里cmp函数直接借助string字典序的性质. 由于 s1 + s2 与 s2 + s1是
  //等长的, 因此字典序也就是数字大小的顺序
  auto cmp = [](const int a, const int b) {
    const auto s1 = std::to_string(a), s2 = std::to_string(b);
    if (s1 + s2 > s2 + s1) return true;
    return false;
  };
  //排序
  std::sort(nums.begin(), nums.end(), cmp);
  //加入答案
  for (auto iter : nums) {
    res.append(std::to_string(iter));
  }
  return res;
}
```

3.5 调试与分析

过于简单, 无需调试.

3.6 总结

关键在于提取出关于不同数字的全序关系, 划归为排序问题.

4. 三数之和

4.1 问题描述

给你一个整数数组 nums , 判断是否存在三元组 [nums[i], nums[j], nums[k]] 满足 i != j、i != k 且 j != k , 同时还满足 nums[i] + nums[j] + nums[k] == 0 。请你返回所有和为 0 且不重复的三元组, 每个三元组占一行。

4.2 问题分析与解决思路

BF算法是不可取的, 直接达到$O(n^3)$的复杂度; 借鉴两数之和的哈希表方法也不可取, 因为无法进行不重复筛查.重复的源头就是, 选定元素a, b, c, 首先选取a, 那么在从小到大选取的过程中, a可能相同, 因此首先做的第一步就是排除第一个数相同的. 同时再利用双指针确定b, c, 确定之后排除相同的. 可以全局去重.

4.3 数据结构设计

基于向量的排序

4.4 功能函数设计

```
void threeSum() {
    std::sort(A.begin(), A.end());

    for (int n1 = 0; n1 < A.size() - 2; n1++) {  // n1最大是size()-3,
    // 进行a的去重
      if (n1 > 0 && A[n1] == A[n1 - 1]) continue;

      int n2 = n1 + 1, n3 = A.size() - 1;

      //转变为两数之和问题
      const int target = -A[n1];
      while (n2 < n3) {
        const auto sum = A[n2] + A[n3];
        //根据<target, >target进行分类
        if (sum < target) {
          n2++;
        } else if (sum > target) {
          n3--;
        } else {
          // assert 此时相等输出答案并且剔除等号
          std::cout << A[n1] << ' ' << A[n2++] << ' ' << A[n3--] << '\n';

          //剔除等号
          while (n2 < A.size() && A[n2] == A[n2 - 1]) {
            n2++;
          }
          //剔除等号
          while (n3 >= 0 && A[n3] == A[n3 + 1]) {
            n3--;
          }
        }
      }
    }
  }
```

4.5 调试与分析

两个边界条件, 剔除等号的循环什么时候停止. 首先是n2,n3的合法性,其次是与上一个相等, 因此采用while循环而不是do while.

```
//剔除等号
while (n2 < A.size() && A[n2] == A[n2 - 1]) {
n2++;
}
//剔除等号
while (n3 >= 0 && A[n3] == A[n3 + 1]) {
n3--;
}
```

4.6 总结

通过排序+双指针在复杂度中剔除掉一个$O(n)$, 从两端分别扫描, 实现了$O(n^2)$的时间复杂度. 因此本题需要提取出不变性, 也就是序列的单调性, 由此就可以通过双指针进行优化.

5.最好的战舰

5.1 问题描述

见题目. 描述过长, 不再赘述.
约定输入格式, 例如

```
Lv 22 //属性名字 权重
n 126 //战舰名字 属性值
ZikavSEzy 237 // 以下同上
cpvkF 169
NLVvpayh 122
ICsxGw 117
pGRNvSmcs 41
ndxFfZ 129
RKP 65
FTsYSApoFZ 162
wxhSC 34
ojV 172
AoGoYF 167
maLLD 241
hG 159
QMUJov 150
TEND
```

5.2 问题分析与解决思路

分析本题, 字典树可以实现的是字符串的插入与查询, 并且值得注意的是, 查询返回的是bool值.
个人认为字典树本身对此题的高效解决并无意义. 因此采用哈希表进行实现, 但是为了避免偷懒嫌疑, 仍然实现了字典树本身. 具体设计见5.3

5.3 数据结构设计

```
class BestShip {
public:
  //读取一张表
  void read_table();
  //Debug使用
  void print_weight_index() const;
  void print_all_ships() const;
  //输出最终结果
  void output() const;
  //solve
  void select_and_solve();

private:
  void calc_score();

  struct Ship {
    std::string ship_name;//ship的名字
    std::vector<float> ship_properties_value;//ship的属性值
    float score = 0;//ship的得分
  };

  struct Property {//存储属性，由于所有ship都共用一个属性，因此只存储一份
    float property_weight;
    const std::string property_name;
    Property(const std::string &name, const float w = 0)
        : property_name(name), property_weight(w) {}
  };

  //这里采用哈希表，把属性名字映射到rank
  std::unordered_map<std::string, const int> property_name_to_index;
  //这里采用哈希表，把战舰名字映射到rank
  std::unordered_map<std::string, const int> ship_name_to_index;
  //存储所有属性
  std::vector<Property> properties;
  //存储所有ship
  std::vector<Ship> allships;

  //辅助变量
  int weight_id = 0;
  int ship_id = 0;
  int max_shipname_length = 0;
```

```
  int max_propertyname_length = 0;
  bool first_read = true;
};
```

5.4 功能函数设计

```
//只有第一次读取ship_name时插入战舰名字与位置的映射
//第一次读取属性name的时候插入属性name与位置的映射
void BestShip::read_table() {
  std::string property_name;
  float weight{};
  std::cin >> property_name >> weight;
  property_name_to_index.insert({property_name, weight_id++});
  max_propertyname_length =
      std::max(max_shipname_length, (int)property_name.size());
  properties.emplace_back(property_name, weight);

  std::string ship_name;
  float value{};
  std::cin.clear();
  while (std::cin >> ship_name) {
    if (ship_name == "TEND")
      break;
    std::cin >> value;

    if (first_read) {
      // 还没有将这个name加入映射之中
      ship_name_to_index.insert({ship_name, ship_id++});
      // 还没有构造足够的ship
      allships.emplace_back();
      max_shipname_length =
          std::max(max_shipname_length, (int)ship_name.length());
    }
    const auto cur_index = ship_name_to_index[ship_name];
    // 添加当前属性的值
    if (first_read)
      allships[cur_index].ship_name = ship_name;

    allships[cur_index].ship_properties_value.push_back(value);
  }
  if (first_read)
    first_read = false;
}
```

```cpp
//计算与排序
void BestShip::calc_score() {
  //按权求和计算属性
  auto calc = [this](Ship &ship) {
    ship.score = 0;
    for (int i = 0; i < ship.ship_properties_value.size(); i++) {
      ship.score +=
          ship.ship_properties_value[i] * properties[i].property_weight;
    }
  };
  std::for_each(allships.begin(), allships.end(), calc);
}

void BestShip::select_and_solve() {
  calc_score();
  //依照score进行排序
  auto cmp = [](const Ship &a, const Ship &b) { return a.score > b.score; };
  std::sort(allships.begin(), allships.end(), cmp);
}
```

5.5 调试与分析

问题主要在于vector越界, 以及何时插入哈希映射, 如果多次插入会造成效率低下；在使用unordered_map的时候, 采用const int避免映射的改变.

5.6 总结

假定N个战舰, P条属性, 借助哈希表, 读取需要$O(N*P)$的时间, 存储需要$O(N*P)$的空间, 每个战舰计算分数需要$O(P)$的时间, 排序需要$O(N\log N)$. 因此问题在相对优秀的复杂度得到解决.

5.7 测试结果

输入(由程序生成, 生成测试程序见后续说明)

5

ckFKAXKljH 20
wxhSC 275
ICsxGw 200
pGRNvSmcs 3
RKP 161
ndxFfZ 201
ojV 255
QMUJov 202
AoGoYF 182
maLLD 117
n 232
NLVvpayh 89
hG 118
ZikavSEzy 229
cpvkF 49
FTsYSApoFZ 82
TEND

jWkUnXZZMm 18
pGRNvSmcs 234
ojV 1
NLVvpayh 271
FTsYSApoFZ 113
AoGoYF 285
QMUJov 12
ICsxGw 245
cpvkF 121
ZikavSEzy 181
RKP 96
maLLD 135
wxhSC 59
n 39
ndxFfZ 224
hG 43
TEND

oElHkaGQC 15
RKP 223
ICsxGw 161
cpvkF 103
n 105

FTsYSApoFZ 225
NLVvpayh 92
ZikavSEzy 122
wxhSC 100
pGRNvSmcs 213
AoGoYF 235
maLLD 260
ojV 119
QMUJov 93
ndxFfZ 264
hG 187
TEND

UeOwyZvcp 2
n 74
hG 127
RKP 128
QMUJov 20
ICsxGw 262
ndxFfZ 296
maLLD 295
pGRNvSmcs 64
AoGoYF 202
cpvkF 111
wxhSC 188
FTsYSApoFZ 160
ZikavSEzy 224
ojV 205
NLVvpayh 126
TEND

Lv 22
n 126
ZikavSEzy 237
cpvkF 169
NLVvpayh 122
ICsxGw 117
pGRNvSmcs 41
ndxFfZ 129
RKP 65
FTsYSApoFZ 162
wxhSC 34
ojV 172

```
AoGoYF 167
maLLD 241
hG 159
QMUJov 150
TEND
```

输出

//这是对表的直接输出，最后输出在后半部分

```
Ship name : wxhSC
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   275
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   59
p_name :    oElHkaGQC   p_weight :  15   p_value :   100
p_name :    UeOwyZvcp   p_weight :  2    p_value :   188
p_name :    Lv          p_weight :  22   p_value :   34

Ship name : ICsxGw
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   200
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   245
p_name :    oElHkaGQC   p_weight :  15   p_value :   161
p_name :    UeOwyZvcp   p_weight :  2    p_value :   262
p_name :    Lv          p_weight :  22   p_value :   117

Ship name : pGRNvSmcs
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   3
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   234
p_name :    oElHkaGQC   p_weight :  15   p_value :   213
p_name :    UeOwyZvcp   p_weight :  2    p_value :   64
p_name :    Lv          p_weight :  22   p_value :   41

Ship name : RKP
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   161
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   96
p_name :    oElHkaGQC   p_weight :  15   p_value :   223
p_name :    UeOwyZvcp   p_weight :  2    p_value :   128
p_name :    Lv          p_weight :  22   p_value :   65

Ship name : ndxFfZ
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   201
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   224
p_name :    oElHkaGQC   p_weight :  15   p_value :   264
p_name :    UeOwyZvcp   p_weight :  2    p_value :   296
p_name :    Lv          p_weight :  22   p_value :   129

Ship name : ojV
Total score : 0
```

```
p_name :    ckFKAXKljH  p_weight :  20   p_value :   255
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   1
p_name :    oElHkaGQC   p_weight :  15   p_value :   119
p_name :    UeOwyZvcp   p_weight :  2    p_value :   205
p_name :    Lv          p_weight :  22   p_value :   172

Ship name : QMUJov
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   202
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   12
p_name :    oElHkaGQC   p_weight :  15   p_value :   93
p_name :    UeOwyZvcp   p_weight :  2    p_value :   20
p_name :    Lv          p_weight :  22   p_value :   150

Ship name : AoGoYF
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   182
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   285
p_name :    oElHkaGQC   p_weight :  15   p_value :   235
p_name :    UeOwyZvcp   p_weight :  2    p_value :   202
p_name :    Lv          p_weight :  22   p_value :   167

Ship name : maLLD
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   117
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   135
p_name :    oElHkaGQC   p_weight :  15   p_value :   260
p_name :    UeOwyZvcp   p_weight :  2    p_value :   295
p_name :    Lv          p_weight :  22   p_value :   241

Ship name : n
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   232
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   39
p_name :    oElHkaGQC   p_weight :  15   p_value :   105
p_name :    UeOwyZvcp   p_weight :  2    p_value :   74
p_name :    Lv          p_weight :  22   p_value :   126

Ship name : NLVvpayh
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   89
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   271
p_name :    oElHkaGQC   p_weight :  15   p_value :   92
```

```
p_name :    UeOwyZvcp   p_weight :  2    p_value :   126
p_name :    Lv          p_weight :  22   p_value :   122

Ship name : hG
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   118
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   43
p_name :    oElHkaGQC   p_weight :  15   p_value :   187
p_name :    UeOwyZvcp   p_weight :  2    p_value :   127
p_name :    Lv          p_weight :  22   p_value :   159

Ship name : ZikavSEzy
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   229
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   181
p_name :    oElHkaGQC   p_weight :  15   p_value :   122
p_name :    UeOwyZvcp   p_weight :  2    p_value :   224
p_name :    Lv          p_weight :  22   p_value :   237

Ship name : cpvkF
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   49
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   121
p_name :    oElHkaGQC   p_weight :  15   p_value :   103
p_name :    UeOwyZvcp   p_weight :  2    p_value :   111
p_name :    Lv          p_weight :  22   p_value :   169

Ship name : FTsYSApoFZ
Total score : 0
p_name :    ckFKAXKljH  p_weight :  20   p_value :   82
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   113
p_name :    oElHkaGQC   p_weight :  15   p_value :   225
p_name :    UeOwyZvcp   p_weight :  2    p_value :   160
p_name :    Lv          p_weight :  22   p_value :   162




_____________________________________
//这里是最后结果输出
Ship name : AoGoYF
Total score : 16373
p_name :    ckFKAXKljH  p_weight :  20   p_value :   182
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   285
```

```
p_name :    oElHkaGQC   p_weight :  15   p_value :   235
p_name :    UeOwyZvcp   p_weight :  2    p_value :   202
p_name :    Lv          p_weight :  22   p_value :   167

Ship name : ndxFfZ
Total score : 15442
p_name :    ckFKAXKljH  p_weight :  20   p_value :   201
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   224
p_name :    oElHkaGQC   p_weight :  15   p_value :   264
p_name :    UeOwyZvcp   p_weight :  2    p_value :   296
p_name :    Lv          p_weight :  22   p_value :   129

Ship name : ZikavSEzy
Total score : 15330
p_name :    ckFKAXKljH  p_weight :  20   p_value :   229
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   181
p_name :    oElHkaGQC   p_weight :  15   p_value :   122
p_name :    UeOwyZvcp   p_weight :  2    p_value :   224
p_name :    Lv          p_weight :  22   p_value :   237

Ship name : maLLD
Total score : 14562
p_name :    ckFKAXKljH  p_weight :  20   p_value :   117
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   135
p_name :    oElHkaGQC   p_weight :  15   p_value :   260
p_name :    UeOwyZvcp   p_weight :  2    p_value :   295
p_name :    Lv          p_weight :  22   p_value :   241

Ship name : ICsxGw
Total score : 13923
p_name :    ckFKAXKljH  p_weight :  20   p_value :   200
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   245
p_name :    oElHkaGQC   p_weight :  15   p_value :   161
p_name :    UeOwyZvcp   p_weight :  2    p_value :   262
p_name :    Lv          p_weight :  22   p_value :   117

Ship name : ojV
Total score : 11097
p_name :    ckFKAXKljH  p_weight :  20   p_value :   255
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   1
p_name :    oElHkaGQC   p_weight :  15   p_value :   119
p_name :    UeOwyZvcp   p_weight :  2    p_value :   205
p_name :    Lv          p_weight :  22   p_value :   172
```

Ship name : NLVvpayh
Total score : 10974
p_name : ckFKAXKljH p_weight : 20 p_value : 89
p_name : jWkUnXZZMm p_weight : 18 p_value : 271
p_name : oElHkaGQC p_weight : 15 p_value : 92
p_name : UeOwyZvcp p_weight : 2 p_value : 126
p_name : Lv p_weight : 22 p_value : 122

Ship name : FTsYSApoFZ
Total score : 10933
p_name : ckFKAXKljH p_weight : 20 p_value : 82
p_name : jWkUnXZZMm p_weight : 18 p_value : 113
p_name : oElHkaGQC p_weight : 15 p_value : 225
p_name : UeOwyZvcp p_weight : 2 p_value : 160
p_name : Lv p_weight : 22 p_value : 162

Ship name : RKP
Total score : 9979
p_name : ckFKAXKljH p_weight : 20 p_value : 161
p_name : jWkUnXZZMm p_weight : 18 p_value : 96
p_name : oElHkaGQC p_weight : 15 p_value : 223
p_name : UeOwyZvcp p_weight : 2 p_value : 128
p_name : Lv p_weight : 22 p_value : 65

Ship name : n
Total score : 9837
p_name : ckFKAXKljH p_weight : 20 p_value : 232
p_name : jWkUnXZZMm p_weight : 18 p_value : 39
p_name : oElHkaGQC p_weight : 15 p_value : 105
p_name : UeOwyZvcp p_weight : 2 p_value : 74
p_name : Lv p_weight : 22 p_value : 126

Ship name : hG
Total score : 9691
p_name : ckFKAXKljH p_weight : 20 p_value : 118
p_name : jWkUnXZZMm p_weight : 18 p_value : 43
p_name : oElHkaGQC p_weight : 15 p_value : 187
p_name : UeOwyZvcp p_weight : 2 p_value : 127
p_name : Lv p_weight : 22 p_value : 159

Ship name : wxhSC
Total score : 9186

```
p_name :    ckFKAXKljH  p_weight :  20   p_value :   275
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   59
p_name :    oElHkaGQC   p_weight :  15   p_value :   100
p_name :    UeOwyZvcp   p_weight :  2    p_value :   188
p_name :    Lv          p_weight :  22   p_value :   34

Ship name : QMUJov
Total score : 8991
p_name :    ckFKAXKljH  p_weight :  20   p_value :   202
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   12
p_name :    oElHkaGQC   p_weight :  15   p_value :   93
p_name :    UeOwyZvcp   p_weight :  2    p_value :   20
p_name :    Lv          p_weight :  22   p_value :   150

Ship name : cpvkF
Total score : 8643
p_name :    ckFKAXKljH  p_weight :  20   p_value :   49
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   121
p_name :    oElHkaGQC   p_weight :  15   p_value :   103
p_name :    UeOwyZvcp   p_weight :  2    p_value :   111
p_name :    Lv          p_weight :  22   p_value :   169

Ship name : pGRNvSmcs
Total score : 8497
p_name :    ckFKAXKljH  p_weight :  20   p_value :   3
p_name :    jWkUnXZZMm  p_weight :  18   p_value :   234
p_name :    oElHkaGQC   p_weight :  15   p_value :   213
p_name :    UeOwyZvcp   p_weight :  2    p_value :   64
p_name :    Lv          p_weight :  22   p_value :   41
```

5.8 完整代码

第零部分：CMakeLists.txt

```
set(CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS ON)

add_executable(TrieTest TrieTest.cc)
add_executable(generate_test generate_test.cc)
add_executable(TheBestShip BestShip.cc)
add_library(Trie SHARED Trie_impl.cpp)

target_link_libraries(TrieTest PUBLIC Trie)
```

注意顶层cmake还有

```
cmake_minimum_required(VERSION 3.20)

project(HW6 CXX)

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/bin)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

include_directories(./)

if(CMAKE_CXX_COMPILER_ID STREQUAL MSVC)
    add_compile_options(/utf-8)
endif()

add_subdirectory(./MaxNumber)
add_subdirectory(./Sort)
add_subdirectory(./Inversion)
add_subdirectory(./ThreeSum)
add_subdirectory(./Trie)
```

第一部分：Trie实现-Trie.hpp

```cpp
#include <array>
#include <string>
#include <vector>

class Trie {
  static constexpr int CHARACTER_NUMBER = 26;

public:
  Trie(const int capacity = 30) {
    buffer.reserve(capacity * 4);
    expand();
  };
  bool insert(const std::string &s);
  bool find(const std::string &s) const;
  int size() const { return _size; }
  bool operator[](const std::string &s) const { return find(s); }

protected:
  void expand() { buffer.emplace_back(); }
  struct TrieNode {
    friend Trie;
    int &operator[](const int index) { return child[index]; }
    int operator[](const int index) const { return child[index]; }

  private:
    std::array<int, CHARACTER_NUMBER> child{};
    int node_count = 0;
  };
  std::vector<TrieNode> buffer;
  int tot = 0;
  int _size = 0;
};
```

第二部分：Trie实现-Trie_impl.cpp

```cpp
#include "Trie.hpp"
#include <cctype>

bool Trie::insert(const std::string &s) {
  int p = 0;
  _size++;
  for (const auto iter : s) {
    const int id = std::tolower(iter) - 'a';
    if (buffer[p][id] == 0) {
      expand();
      buffer[p][id] = ++tot;
    }
    p = buffer[p][id];
  }
  buffer[p].node_count++;
  return true;
}

bool Trie::find(const std::string &s) const {
  int p = 0;
  for (const auto iter : s) {
    const int id = std::tolower(iter) - 'a';
    const auto &node = buffer[p];
    if (node[id] == 0)
      return false;
    p = node[id];
  }
  return buffer[p].node_count;
}
```

第三部分：BestShip实现-BestShip.cc

```
#include <algorithm>
#include <iomanip>
#include <iostream>
#include <string>
#include <unordered_map>
#include <vector>

class BestShip {
public:
  void read_table();
  void print_weight_index() const;
  void print_all_ships() const;
  void output() const;
  void select_and_solve();

private:
  void calc_score();

  struct Ship {
    std::string ship_name;
    std::vector<float> ship_properties_value;
    float score = 0;
  };

  struct Property {
    float property_weight;
    const std::string property_name;
    Property(const std::string &name, const float w = 0)
        : property_name(name), property_weight(w) {}
  };

  std::unordered_map<std::string, const int> property_name_to_index;
  std::unordered_map<std::string, const int> ship_name_to_index;

  std::vector<Property> properties;
  std::vector<Ship> allships;

  int weight_id = 0;
  int ship_id = 0;
  int max_shipname_length = 0;
  int max_propertyname_length = 0;
```

```
  bool first_read = true;
};

void BestShip::output() const {
  using std::setw;
  std::cout << std::setiosflags(std::ios_base::left);
  auto print = [this](const Ship &ship) {
    for (int i = 0; i < properties.size(); i++) {
      std::cout << setw(12) << "p_name : " << setw(max_propertyname_length + 2)
                << properties[i].property_name;
      std::cout << setw(12) << "p_weight : " << setw(5)
                << properties[i].property_weight;
      std::cout << setw(12) << "p_value : " << ship.ship_properties_value[i]
                << "\t";
      std::cout << std::endl;
    }
    std::cout << std::endl;
  };

  for (const auto &iter : allships) {
    std::cout << "Ship name : " << setw(max_shipname_length + 2)
              << iter.ship_name << std::endl;
    std::cout << "Total score : " << iter.score << std::endl;
    print(iter);
  }
}

void BestShip::print_weight_index() const {
  for (const auto &iter : property_name_to_index) {
    std::cout << iter.first << ' ' << iter.second << '\n';
  }
}

void BestShip::print_all_ships() const {
  for (const auto &iter : ship_name_to_index) {
    std::cout << iter.first << ' ' << iter.second << '\n';
  }
}

void BestShip::read_table() {
  std::string property_name;
  float weight{};
  std::cin >> property_name >> weight;
```

```
  property_name_to_index.insert({property_name, weight_id++});
  max_propertyname_length =
      std::max(max_shipname_length, (int)property_name.size());
  properties.emplace_back(property_name, weight);

  std::string ship_name;
  float value{};
  std::cin.clear();
  while (std::cin >> ship_name) {
    if (ship_name == "TEND")
      break;
    std::cin >> value;

    if (first_read) {
      // 还没有将这个name加入映射之中
      ship_name_to_index.insert({ship_name, ship_id++});
      // 还没有构造足够的ship
      allships.emplace_back();
      max_shipname_length =
          std::max(max_shipname_length, (int)ship_name.length());
    }
    const auto cur_index = ship_name_to_index[ship_name];
    // 添加当前属性的值
    if (first_read)
      allships[cur_index].ship_name = ship_name;

    allships[cur_index].ship_properties_value.push_back(value);
  }
  if (first_read)
    first_read = false;
}

void BestShip::calc_score() {
  auto calc = [this](Ship &ship) {
    ship.score = 0;
    for (int i = 0; i < ship.ship_properties_value.size(); i++) {
      ship.score +=
          ship.ship_properties_value[i] * properties[i].property_weight;
    }
  };
  std::for_each(allships.begin(), allships.end(), calc);
}
```

```
void BestShip::select_and_solve() {
  calc_score();
  auto cmp = [](const Ship &a, const Ship &b) { return a.score > b.score; };
  std::sort(allships.begin(), allships.end(), cmp);
}

int main() {
  freopen("BestShip.txt", "r", stdin);
  BestShip T;
  int N = 0;
  std::cin >> N;
  while (N--) {
    T.read_table();
  }
  T.output();
  std::cout << "\n\n_____\n\n";

  T.select_and_solve();
  T.output();
  //  T.print_weight_index();
  //  T.print_all_ships();
  // T.output();
}
```

第四部分: 测试生成代码-generate_test.cc

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

constexpr unsigned SEED = 0;

static std::string random_alphabet_string(const int L = 10) {
  std::string res;
  res.reserve(L);
  int len = rand() % 10 + 1;
  while (len--) {
    const char ch = rand() % 26 + (rand() % 2 ? 'A' : 'a');
    res += ch;
  }
  return res;
}
struct Property {
  float property_weight;
  const std::string property_name;
  Property(const std::string &name, const float w = 0)
      : property_name(name), property_weight(w) {}
};

class RandomTest {
public:
  RandomTest(const int n = 15, const int p = 5) : N_ship(n), N_property(p) {}
  void generate_cases();

private:
  void generate_ship_name();
  void generate_property();
  void generate_table(const Property &property) const;
  std::vector<std::string> ship_name;
  std::vector<Property> property;
  const int N_ship;
  const int N_property;
  static constexpr int property_value_limits = 300;
  static constexpr int weight_value_limits = 30;
};

void RandomTest::generate_ship_name() {
```

```
  ship_name.reserve(N_ship);
  for (int i = 0; i < N_ship; i++) {
    auto cur_name = random_alphabet_string();
    while (std::find(ship_name.begin(), ship_name.end(), cur_name) !=
           ship_name.end())
      cur_name = random_alphabet_string();
    ship_name.emplace_back(std::move(cur_name));
  }
}

void RandomTest::generate_property() {
  property.reserve(N_property);

  for (int i = 0; i < N_property; i++) {
    auto cur_name = random_alphabet_string();

    auto p = [&cur_name](const Property &name) {
      return cur_name == name.property_name;
    };
    while (std::find_if(property.begin(), property.end(), p) != property.end())
      cur_name = random_alphabet_string();

    property.emplace_back(cur_name, float(rand() % weight_value_limits));
  }
}

void RandomTest::generate_table(const Property &property) const {
  std::cout << property.property_name << ' ' << property.property_weight
            << std::endl;
  std::vector<bool> chosen(ship_name.size(), false);
  for (int i = 0; i < ship_name.size(); i++) {
    int target = rand() % ship_name.size();
    while (chosen[target]) {
      target = rand() % ship_name.size();
    }
    chosen[target] = true;

    std::cout << ship_name[target] << ' ';
    std::cout << rand() % property_value_limits << std::endl;
  }
  std::cout << "TEND\n" << std::endl;
}
```

```
void RandomTest::generate_cases() {
  std::cout << N_property << "\n\n";
  generate_ship_name();
  generate_property();
  for (const auto &prop : property) {
    generate_table(prop);
  }
}
int main() {
  freopen("BestShip.txt", "w", stdout);
  RandomTest T;
  T.generate_cases();
}
```