

目录

目录

1.项目背景

2.需求分析

3.设计思路

4.核心代码说明

数据结构

贪心算法

5.使用方法及函数功能演示

输入数据

输出数据

1.项目背景

求解最优化问题的算法通常需要经过一系列步骤，在每个步骤都面临多种选择。对于许多最优化问题，使用动态规划算法来求最优解有点杀鸡用牛刀，可以使用更简单、更高效的算法——**贪心算法**，这种算法会在每一步都做当时看起来最佳的选择，即通过总是做出局部最优解，来达到全局最优解。

贪心问题并不保证得到最优解，但对于很多问题确实可以得到最优解。如经典的木材切割问题，本项目解决的问题就属于木材切割问题的变形：

农夫要修理牧场的一段栅栏，他测量了栅栏，发现需要 N 块木头，每块木头长度为整数 L_i 个长度单位，于是他购买了一个很长的，能锯成 N 块的木头，即该木头的长度是 L_i 的总和。但是农夫自己没有锯子，请人锯木的酬金跟这段木头的长度成正比。为简单起见，不妨就设酬金等于所锯木头的长度。例如，要将长度为20的木头锯成长度为8，7和5的三段，第一次锯木头将木头锯成12和8，花费20；第二次锯木头将长度为12的木头锯成7和5花费12，总花费32元。如果第一次将木头锯成15和5，则第二次将木头锯成7和8，那么总的花费是35（大于32）

2.需求分析

项目功能要求：

- 输入格式：输入第一行给出正整数 N （ $N \leq 104$ ），表示要将木头锯成 N 块。第二行给出 N 个正整数，表示每块木头的长度
- 输出格式：输出一个整数，即将木头锯成 N 块的最小花费

3.设计思路

在该问题中，无论采用怎样的切割方式，最终获得的木头数量和每段长度都是确定的，而每次切割的成本等于切割后木头长度和，即当前木块长度，换句话说，采用自底向上的思路，可以视作固定长度的木块依次组合，最终形成完整木块，我们能调整的是组合顺序。每一次组合的成本为当前长度和，因此，要使总花费最小，一定要让长度越长的木块越靠后组合，即长度越长的木块要越早切割掉。以每次切割作为子问题，每次切割掉最长的木块为子问题最优解，合起来即为全局最优解。

4.核心代码说明

数据结构

采用vector存储需要截成的木头长度

main部分

```
1  cout << "how many pieces of wood do you want to get? "<<endl;
2  cin >> N;
3  cout << "Please enter the length of each wood:" << endl;
4  for (int i = 0; i < N; i++) {
5      cin >> temp;
6      lengths.push_back(temp);
7  }
8  greedProblem(lengths, sum);
```

存储后作为参数传入贪心算法的函数

贪心算法

贪心算法通过递归实现，递归终止条件为储存所有木块长度的vector大小变为1（即所有木块都切割完毕）。每一次将vector中长度按升序排列，然后计算长度和（即该次切割总花费），随后将vector尾，即最大数弹出（相当于切割掉长度最长木块），然后再调用该函数（开始下一次切割）

```
1  void greedProblem(vector<int>& lengths,int sum) {
2      if (lengths.size() == 1) {
3          cout << "The least cost is:" << sum;
4          return;
5      }
6      sort(lengths.begin(), lengths.end());
7      for (auto i : lengths)
8          sum += i;
9      lengths.pop_back();
10     greedProblem(lengths, sum);
11 }
```

5.使用方法及函数功能演示

双击 `greedy_problem.exe` 运行程序，按照系统提示继续操作

输入数据

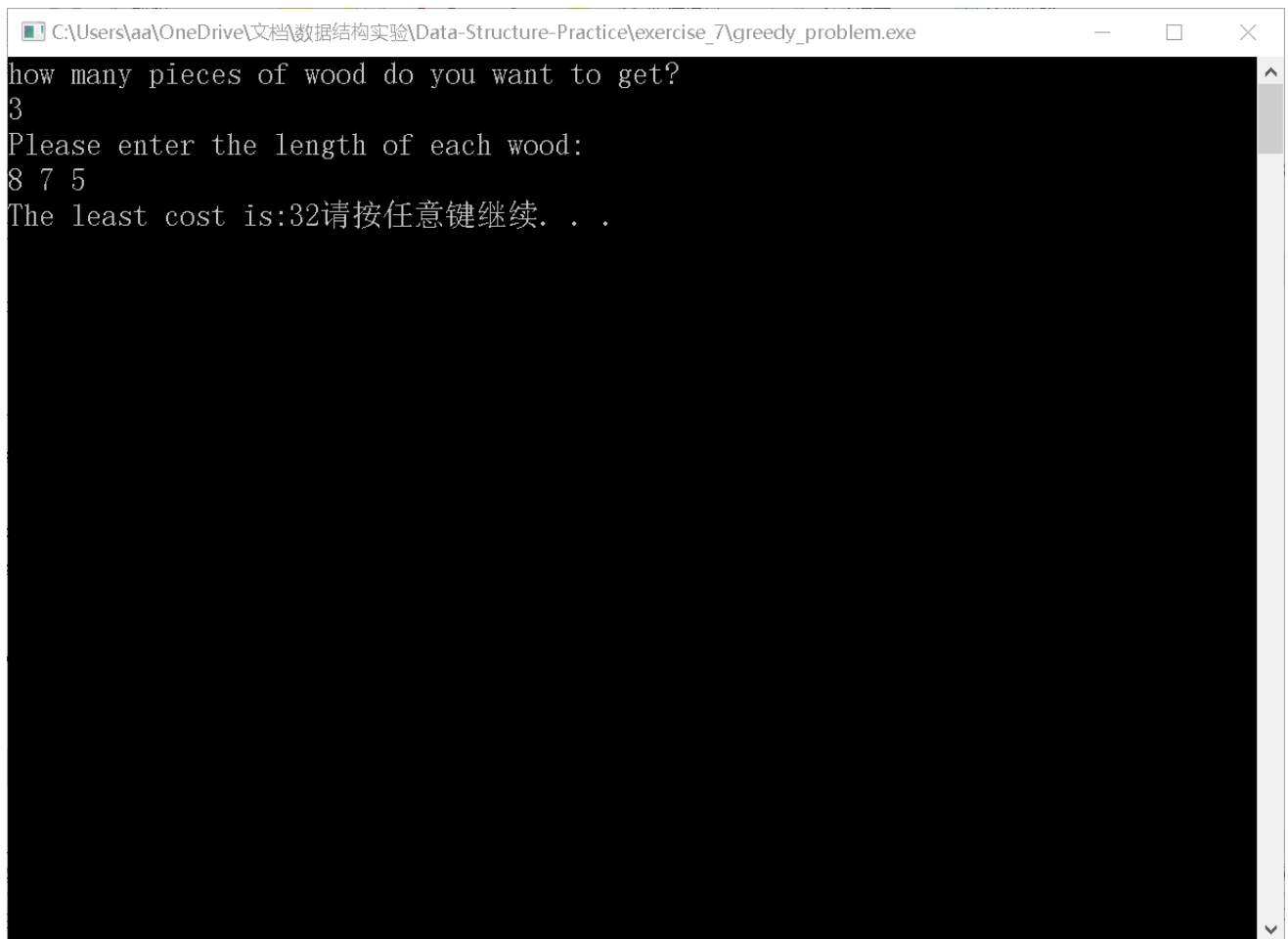
- 输入木块数N
- 输入每块木头长度

```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_7\greedy_problem.exe
how many pieces of wood do you want to get?
```

```
C:\Users\aa\OneDrive\文档\数据结构实验\Data-Structure-Practice\exercise_7\greedy_problem.exe
how many pieces of wood do you want to get?
3
Please enter the length of each wood:
8 7 5
```

输出数据

输出的整数即为把木头锯成N块的最小花费



```
C:\Users\aa\OneDrive\文档\数据结构实验\Data-Structure-Practice\exercise_7\greedy_problem.exe
how many pieces of wood do you want to get?
3
Please enter the length of each wood:
8 7 5
The least cost is:32请按任意键继续. . .
```

若要再次运行程序则重新启动exe文件