项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 曹峰源

学 号： 1951328

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目录

[1. 背景及分析 3](#_Toc57657852)

[1.1 背景 3](#_Toc57657853)

[1.2 分析 3](#_Toc57657854)

[2. 设计 3](#_Toc57657855)

[2.1 类结构设计 3](#_Toc57657856)

[2.2 成员与操作设计 3](#_Toc57657857)

[2.2.1 结点类 3](#_Toc57657858)

[2.2.2 链表类 3](#_Toc57657859)

[2.2.3 自定义cutprice类 3](#_Toc57657860)

[3. 功能实现 6](#_Toc57657861)

[3.1 流程图 6](#_Toc57657862)

[3.2 结果显示 7](#_Toc57657863)

1. 背景及分析
   1. 背景

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*Li*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*Li*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）。

* 1. 分析

对于该题，可以使用霍夫曼树实现，这就需要每一次找出数列中最小的值，在下述代码中，我没有将霍夫曼树完全实现出来，而是利用霍夫曼树的思想，将所有权值（即每一段木头的长度）保存到链表中，然后每一次都找出链表中的最小值，寻找有两次就将两次取值加到要求的price值中，再将加值存入链表。这样循环直到链表为空就能实现霍夫曼树的构造。

1. 设计
   1. 类结构设计

这段程序的实现使用链表，这也就包含了结点类。另外，代码还新定义了一个cutprice类，将所要计算的功能包装起来，需要计算时在主函数调用该类中的calprice()函数即可。

* 1. 成员与操作设计
     1. 结点类

node：

私有成员：

int length = 0;//定义该段的长度

node\* next = NULL;//定义并初始化下一个指针

* + 1. 链表类

linklist：

私有成员：

node\* first;//头指针

node\* current;//用于移动的current指针

公有成员函数：

1. linklist()//构造函数
2. {
3. first = **new** node;
4. current = first;
5. }
6. ~linklist()//析构函数
7. {
8. current = first->next;
9. **delete** first;
10. **while** (current != NULL)
11. {
12. node\* p = current;
13. current = current->next;
14. **delete** p;
15. }
16. }
17. **void** insert(**int** length)//用于在链表的最后插入结点，结点的length值为参量
18. {
19. current = first;
20. **while** (current->next != NULL)
21. {
22. current = current->next;
23. }
24. node\* p = **new** node;
25. p->length = length;
26. current->next = p;
27. }
29. **bool** del(**int** length)//用于删除链表中出现的第一个length值等于参量的结点
30. {
31. current = first;
32. **while** (current->next != NULL)
33. {
34. **if** (current->next->length == length)
35. {
36. **break**;
37. }
38. current = current->next;
39. }
40. **if** (current->next == NULL)
41. {
42. **return** **false**;
43. }
44. node\* p = current->next;
45. current->next = current->next->next;
46. **delete** p;
47. **return** **true**;
48. }
50. **bool** findmin(**int**& x)//找到链表中length最小的值，并将其赋值给x
51. {
52. current = first->next;
53. **if** (current == NULL)
54. {
55. **return** **false**;
56. }
57. x = current->length;
58. **while** (current != NULL)
59. {
60. **if** (current->length < x)
61. {
62. x = current->length;
63. }
64. current = current->next;
65. }
66. **return** **true**;
67. }
69. **bool** isempty()//判断链表是否为空
70. {
71. current = first->next;
72. **if** (current == NULL) **return** **false**;
73. **else** **return** **true**;
74. }

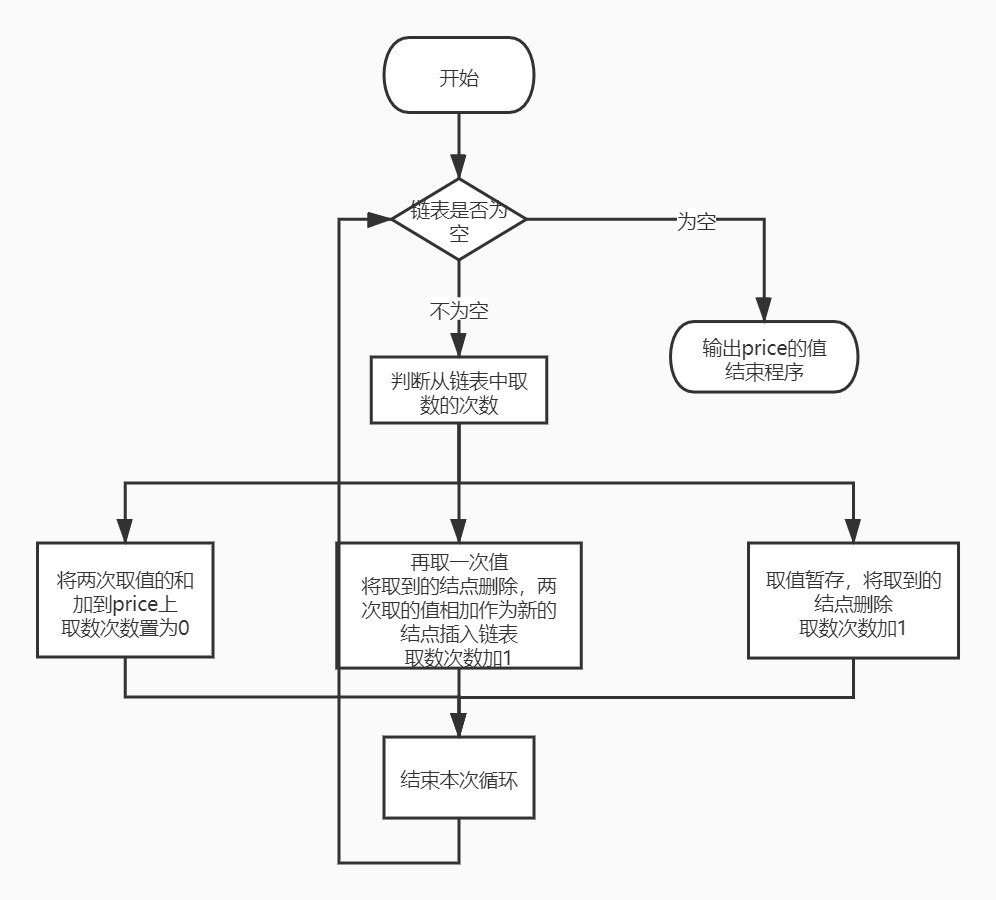
自定义cutprice类

cutprice：

私有成员： linklist s;

公有成员函数：

1. cutprice() {}
2. ~cutprice() {}
3. **void** calprice()
4. {
5. **int** N;
6. cin >> N;//输入要分割的木头块数
7. **int** length, price = 0, count = 0;
8. **while** (count < N)//循环输入每一块的权值，即长度
9. {
10. cin >> length;
11. s.insert(length);
12. count++;
13. }
14. **int** num = 0, getnum = 0, NUM = 0;
15. **while** (s.isempty() != 0)//循环，条件为链表s不为空
16. {
17. **if** (getnum == 2)//当从链表中取数的次数达到两次时
18. {
19. price = price + NUM;//price的值加上NUM
20. num = 0;//三个暂存空间的值全部清零
21. NUM = 0;
22. getnum = 0;
23. }
24. **else** **if** (getnum == 1)//当取值次数为1时
25. {
26. s.findmin(num);//继续取一次值
27. s.del(num);//将取完的该结点删除
28. NUM = NUM + num;//NUM的值极为这两次取值之和
29. s.insert(NUM);//将值为NUM的结点插入链表
30. getnum++;
31. }
32. **else** {
33. s.findmin(num);//getnum值为0时取链表中的最小值
34. s.del(num);//将结点删除
35. NUM = num;
36. getnum++;
37. }
38. }
39. //输出price的值即为要求的最小价格
40. cout << price << endl;
41. }
42. 功能实现
    1. 流程图



* 1. 结果显示

以4 5 1 2 1 3 1 1为例

