**5-21子集树问题**

1、问题描述：是设计一个用回溯法搜索子集数空间的函数。该函数的参数包扣节点可行性判定函数和上界函数等必要函数，并将此函数用于解装载问题。  
有一批共n 个集装箱要装上艘载重量为c 的轮船，其中集装箱i 的重量为wi。找出一种最优装载方案，将轮船尽可能装满，即在装载体积不受限制的情况下，将尽可能重的集装箱装上轮船。

2、程序代码如下：

//load.h

#ifndef LOAD\_H

#define LOAD\_H

#include "load.template"

#endif//load.template

template <class Type>

class Loading

{

friend Type Maxloading(Type[], Type, int, int[]);

private:

void Backtrack(int i);

int n, //集装箱数

\*x, //当前解

\*bestx; //当前最优解

Type \*w, //集装箱重量数组

c, //第一艘轮船的在载重量

cw, //当前载重量

bestw, //当前最优装载重量

r; //剩余集装箱重量

};

template <class Type>

void Loading<Type> ::Backtrack(int i)

{

//搜索第i层节点

if (i > n) //到达叶节点

{

if (cw > bestw)

{

for (int j = 1; j <= n; j++)

bestx[j] = x[j];

bestw = cw;

}

return;

}

//搜索子树

r -= w[i];

if (cw + w[i] <= c) //搜索左子树

{

x[i] = 1;

cw += w[i];

Backtrack(i + 1);

cw -= w[i];

}

if (cw + r > bestw) //搜索右子树

{

x[i] = 0;

Backtrack(i + 1);

}

r += w[i]; //回溯

}

template <class Type>

Type Maxloading(Type w[], Type c, int n, int bestx[])

{

//返回最优载重量

Loading <Type> X;

//初始化X

X.x = new int[n + 1];

X.w = w;

X.c = c;

X.n = n;

X.bestx = bestx;

X.bestw = 0;

X.cw = 0;

//初始化r

X.r = 0;

for (int i = 1; i <= n; i++)

{

X.r += w[i];

}

X.Backtrack(1);

delete[] X.x;

return X.bestw;

}

//2018\_04\_28

//使用回朔法解决装载问题

//main.cpp

//=============================================================================

#include <iostream>

#include "load.h"

using namespace std;

int main(void)

{

int n = 0; //集装箱数量；

int c = 0; //第一艘船的载重重量

int bestx[4]; //最优解

int \*w = new int[n + 1]; //集装箱重量数组

int m = 0; //最优载重量

cout << "请输入集装箱数量: ";

cin >> n;

cout << endl;

cout << "请输入第一艘船的载重重量: ";

cin >> c;

cout << endl;

cout << "请输入集装箱的重量数组 ：";

for (int i = 0; i <= n; i++)

{

cin >> w[i];

}

m = Maxloading(w, c, n, bestx); //求解问题

cout << "输出最优解如下：" << endl;

for (int i = 1; i < n; i++)

{

cout << bestx[i] << " ";

}

cout << "请输出最优载重量： " << endl;

cout << m;

system("pause");

return 0;

}