**5\_23排列树问题**

问题描述:试设计一个用回溯法搜索排列空间树的函数。该函数的参数包括结点可行性判定函数和上界函数等必要的函数，并将此函数用于解圆排列问题。  
圆排列问题描述如下:给定n个大小不等的圆c1，c2 ..., cn，现要将这n个圆排进一一个矩形框中，且要求各圆与矩形框的底边相切。圆排列问题要求从n个圆的所有排列中找出有最小长度的圆排列。例如，当n=3，且所给的3个圆的半径分别为1，1，2时，这了个圆的最小长度的圆排列是1,2,1， 其最小长度为2 + 4\*sqr(2)。

编程任务:对于给定的n个圆，编程计算最小长度排列。  
功能分析:输入由多组测试数据组成。每组测试数据输入的第一行是1个正整数n,表示有n个圆。第2行有n个正数，分别表示n个圆的半径。  
对应每组输入，输出的每行是计算出的最小长度，保留3位小数。  
 例如:输入: 3   输出: 7.65685  
       112  
二、算法过程设计.

1.排列树问题  
排列树问题是通过设置相关数组，调用库函数来解决问题。

1. 程序调试及结果
2. 程序代码：

#include <iostream>

#include <cmath>

#include <vector>  
      using namespace std;

float Min;  
      vector <float> X;

vector <float> r;

int n;  
      float Center(int t)  
      float temp = 0;  
      for(int j = 0;j< t; j++)

{  
       float valuex = x[j] + 2.0 \* sqrt(r[t]\*[j]);

if (valuex > temp)  
      temp = valuex;  
     }  
     return temp;  
}  
void ComputerO  
      {  float low = 0,high = 0;  
      for(int i=0;i< n; i++)

{  
       if (x[i] - r[i] < low)  
      low = x[i] - r[j];

if (x[i] + r[i] > high)  
      high = x[i] +  r[i] ;

}  
      if (high - low < Min)  
      Min = high - low;  
void Backtrack(int t)  
void Backtrack(int t)f

{

if (t> n-1)  
Computer();  
else  
for(int j = t;j< n; j++)  
swap(r[t],r[j]);  
float centerx = Center(t);  
if (centerx +r[t] + r[O] < Min)

{  
x[t] = centerx;

Backtrack(t+1);

}  
swap(r[t],r[j]);

int main(  
while(cin >> n)

{  
Min9999;

x.resize(n);

r.resize(n);  
for(int i=0; i< n; i++)

{  
cin >> r[i];

x[i] = 0;  
}  
Backtrack(0);  
printf("%.3fn",Min);

}  
return0;

}