软件工程第三次上机

U201516980 白云 软工1501

Group PM:白云 U201516980
Group QA:黄建霖

Introduction

In a box bounded by [-1,-1],given m ballons(they can't overlap) with variable radio r and position mu,and some tiny blocks are in the box at given position:{d};ballons can't overlap with these blocks,find optimal value of r and mu which maximmize sum r^2

Algorithm

- 算法思路:贪心算法.每次找正方形中剩余空间中,内切圆最大的那个空间,放入该空间的内切圆.
- 方法:构造所有的相切的圆,然后按照半径排序,假设内切圆中包含block，则删除该种情况，否则求出该点在圆上的情况.
- 构造方法:
  设所求圆半径为r,圆心为x,y,已知圆的半径为r0,圆心为x0,y0,以第一象限为例
  1.一圆两个正方形边界所构成的区域内切圆为
  构造出一个内切圆(内切上述区域)
  则有方程 r+r0 = sqrt((x-x0)^2+(y-y0)^2) (圆外切性质)
  r = 1-x
  r = 1-y
  2.两圆一个正方形边界所构成的区域内切圆为
  构造出两个内切圆(内切上述区域) 且关于 y=x 这条直线对称
  以 x = 1为正方形边界为例
  则有方程 r = 1-x
  r+r0 = sqrt((x-x0)^2+(y-y0)^2)
  r+r1 = sqrt((x-x1)^2+(y-y1)^2)
  同理可得 以 y = 1为正方形边界时的解
  3.三圆构成的区域内切圆为
  根据外切圆的性质
  则有方程 r+r0 = sqrt((x-x0)^2+(y-y0)^2)
  r+r1 = sqrt((x-x1)^2+(y-y1)^2)
  r+r2 = sqrt((x-x2)^2+(y-y2)^2)
  构造出所有的圆之后,根据四个象限的对称性,可得当m取确定的的值时,
  使得sigma(ri^2) (i from 1 to m) 取得最大值时,所有圆的坐标和半径.
- 具体实现看如下代码:

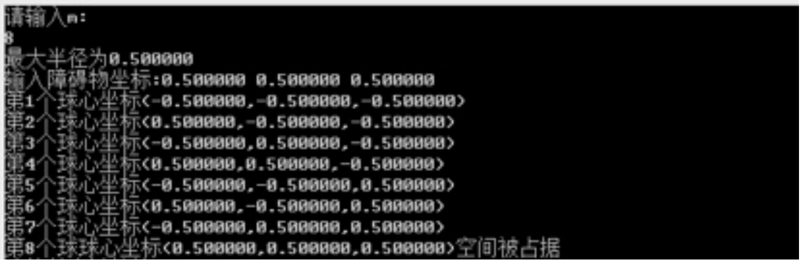
```
\#include \<iostream\>
\#include \<complex\>
```

```
\#include \<algorithm\>
\#include \<cstdio\>
\#include \<cstring\>
\#include \<vector\>
\#include \<queue\>
\#include \<cmath\>
//\#define SHOW\_CONSTRUCT
\#define ITERATION
//\#define NOLIMIT
\#define LIMIT
using namespace std;
const double eps = 1e-6;
const double a = 2;
const int x[4] = {1,-1,1,-1};
const int y[4] = {1,1,-1,-1};
double ans = 0;
struct Balloon{
 double r;
 pair\<double,double\> mu;
 Balloon(){
 r = 0;
 mu.first =0;
 mu.second =0;
 }
 Balloon(double \_r,pair\<double,double\> \_mu){
 r = \_r;
 mu = \_mu;
 }
 //按照半径降序
 bool operator\<(const Balloon& b) const{
 return this-\>r\>b.r;
 }
};
Balloon getFirstSitiuation(double boundX,double boundY,Balloon a){ //第一种情况求解内切圆
 Balloon ans;
 double r = (sqrt(2)-sqrt(2)\*a.mu.first-a.r)/(1+sqrt(2));
 double x,y;
 x = y = 1-r;
 ans.r = r;
 ans.mu.first = x;
 ans.mu.second = y;
 return ans;
};
void iterationMethod(double &r,double &y,Balloon a,Balloon b){
 r = 0;
 y = 0;
 double tmpr = 0;
 double tmpy = 0;
 int num = 0;
 while (num\<1000){
 tmpy = y;
 tmpr = r;
 //r = (y-b.mu.second)\*(y-b.mu.second)/(b.r+1-b.mu.first)+b.mu.first+1-b.r;
 y = sqrt((r+a.r)\*(r+a.r)-(1.-r-a.mu.first)\*(1.-r-a.mu.first))+a.mu.second;
 r = ((b.mu.first-1.)\*(b.mu.first-1)-b.r\*b.r+(y-b.mu.second)\*(y-b.mu.second))/(b.r-b.mu.first+1.);
 r = 0.5*r;
 //cout\<\<y\<\< " "\<\<r\<\<endl;
 if(fabs(r-tmpr)\<=eps&&fabs(y-tmpy)\<=eps){
 break;
 }
 num++;
 }
}
Balloon getSecondSitiuation(double bound,Balloon a,Balloon b){ //第二种情况求解内切圆
 Balloon ans;
 double r,y;
#ifdef ITERATION
 iterationMethod(r,y,a,b);
 ans.mu.first = 1.-r;
 ans.mu.second = y;
 ans.r = r;
 if(isnan(ans.r)||isnan(ans.mu.first)||isnan(ans.mu.second)){
 ans.r = 0;
 ans.mu.first = 0;
 ans.mu.second = 0;
 }
 // cout\<\<r\<\<" --\>r"\<\<endl;
#else
 ans.r = 0;
 ans.mu.first = 0;
 ans.mu.second = 0;
#endif
 return ans;
}
Balloon getThirdSitiuation(Balloon a,Balloon b,Balloon c){ //第三种情况求解内切圆
 Balloon ans;
 ans.r = 0;
 ans.mu.first = 0;
 ans.mu.second = 0;
 return ans;
}
vector\<Balloon\> res;
vector\<Balloon\> conv; // 构造序列
vector\<pair\<double,double\> \>limPoint;
void construct(int m){
 // conv.clear();
#ifdef NOLIMIT
 conv.push_back(Balloon(1,make_pair(0.,0.)));
 Balloon preFisrtSitiuation = conv[0];
 for(int i = 0;i\<=m;i++){
 Balloon tmpFirst = getFirstSitiuation(1,1,preFisrtSitiuation);
 preFisrtSitiuation = tmpFirst;
 conv.push_back(tmpFirst);
 Balloon tmpSecond = getSecondSitiuation(1,tmpFirst,preFisrtSitiuation);
 conv.push_back(tmpSecond);
 conv.push_back(Balloon(tmpSecond.r,make_pair(tmpSecond.mu.second,tmpSecond.mu.first)));
 Balloon tmpThird = getThirdSitiuation(tmpFirst,tmpSecond,preFisrtSitiuation);
 conv.push_back(tmpThird);
 }
#endif
#ifdef LIMIT
```

```
 int times = 0;
 for(int i = 0;i\<conv.size()&&times\<20;i++){
 Balloon tmpfirst = getFirstSitiuation(1,1,conv[i]); //与四个边界进行构造
 Balloon tmpsecond = getFirstSitiuation(-1,-1,conv[i]);
 conv.push\_back(tmpfirst);
 times++;
 // conv.push\_back(tmpsecond);
 }
\#endif
 sort(conv.begin(),conv.end());
}
double getSumrArea(vector\<Balloon\> vec){
 double ans = 0;
 for(int i = 0;i\<vec.size();i++){
 ans+=vec[i].r\*vec[i].r;
 }
 return ans;
}
void solve(){
 res.clear();
 res.push\_back(Balloon(1,make\_pair(0.,0.))); //当m==1时,为正方形的内切圆
}
void showConstruct(){
 for(int i = 0;i\<conv.size();i++){
 cout\<\<"r = "\<\<conv[i].r\<\<" pos ( "\<\<conv[i].mu.first\<\<" , "\<\<conv[i].mu.second\<\<" ) "\<\<endl;
 }
}
void showAns(int m){
 ans = 0;
 if(m\>=1){
 cout\<\<"r = "\<\<conv[0].r\<\<" pos ( "\<\<conv[0].mu.first\<\<" , "\<\<conv[0].mu.second\<\<" ) "\<\<endl;
 ans+=conv[0].r\*conv[0].r;
 }
 int tmpm = m-1;
 int tmpre = tmpm%4;
 tmpm/=4;
 for(int i = 1;i\<=tmpm;i++){
 cout\<\<"r = "\<\<conv[i].r\<\<" pos ( "\<\<conv[i].mu.first\<\<" , "\<\<conv[i].mu.second\<\<" ) "\<\<endl;
 cout\<\<"r = "\<\<conv[i].r\<\<" pos ( "\<\<-conv[i].mu.first\<\<" , "\<\<conv[i].mu.second\<\<" ) "\<\<endl;
 cout\<\<"r = "\<\<conv[i].r\<\<" pos ( "\<\<conv[i].mu.first\<\<" , "\<\<-conv[i].mu.second\<\<" ) "\<\<endl;
 cout\<\<"r = "\<\<conv[i].r\<\<" pos ( "\<\<-conv[i].mu.first\<\<" , "\<\<-conv[i].mu.second\<\<" ) "\<\<endl;
 ans+=4\*conv[i].r\*conv[i].r;
 }
 for(int i = 0;i\<tmpre;i++){
 ans+=conv[tmpm+1].r\*conv[tmpm+1].r;
 cout\<\<"r = "\<\<conv[tmpm+1].r\<\<" pos ( "\<\<x[i]\*conv[tmpm+1].mu.first\<\<" , "\<\<y[i]\*conv[tmpm+1].mu.second\<\<" ) "\<\<endl;
 }
}
void inputLimit(int n){
 conv.clear();
 pair\<double,double\> lim;
 Balloon limBallon;
 for(int i = 0;i\<n;i++){
 cin\>\>lim.first\>\>lim.second;
 limPoint.push\_back(lim);
 limBallon = Balloon(eps,lim);
 conv.push\_back(limBallon);
 }
}
int main() {
 //ios\_base::sync\_with\_stdio(false);
 //cin.tie(NULL);
 int m,n;
 cout\<\<"input m the number of balloon and n the limit points "\<\<endl;
 while (cin\>\>m\>\>n){
 cout\<\<"m = "\<\<m\<\<endl;
 inputLimit(n);
 construct(m);
\#ifdef SHOW\_CONSTRUCT
 showConstruct();
\#endif
 showAns(m);
 //cout\<\<"the max sum r^2 is "\<\<ans\<\<endl;
 printf("the max sum r^2 is %.10lf\\n", ans);
 cout\<\<"\\ninput m the number of balloon and the n the limit points"\<\<endl;
 }
 return 0;
}
```

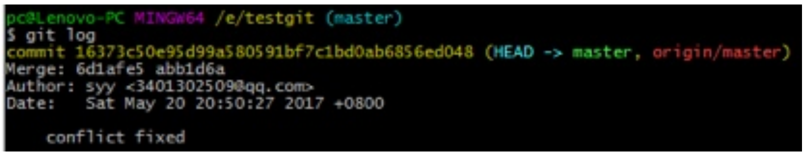
Test

- 运行截图



```
请输入m:
8
最大半径为0.500000
输入障碍物坐标:0.500000 0.500000 0.500000
第1个球心坐标<-0.500000,-0.500000,-0.500000>
第2个球心坐标<0.500000,-0.500000,-0.500000>
第3个球心坐标<-0.500000,0.500000,-0.500000>
第4个球心坐标<0.500000,0.500000,-0.500000>
第5个球心坐标<-0.500000,-0.500000,0.500000>
第6个球心坐标<0.500000,-0.500000,0.500000>
第7个球心坐标<-0.500000,0.500000,0.500000>
第8个球球心坐标<0.500000,0.500000,0.500000>空间被占据
```

- 输出结果



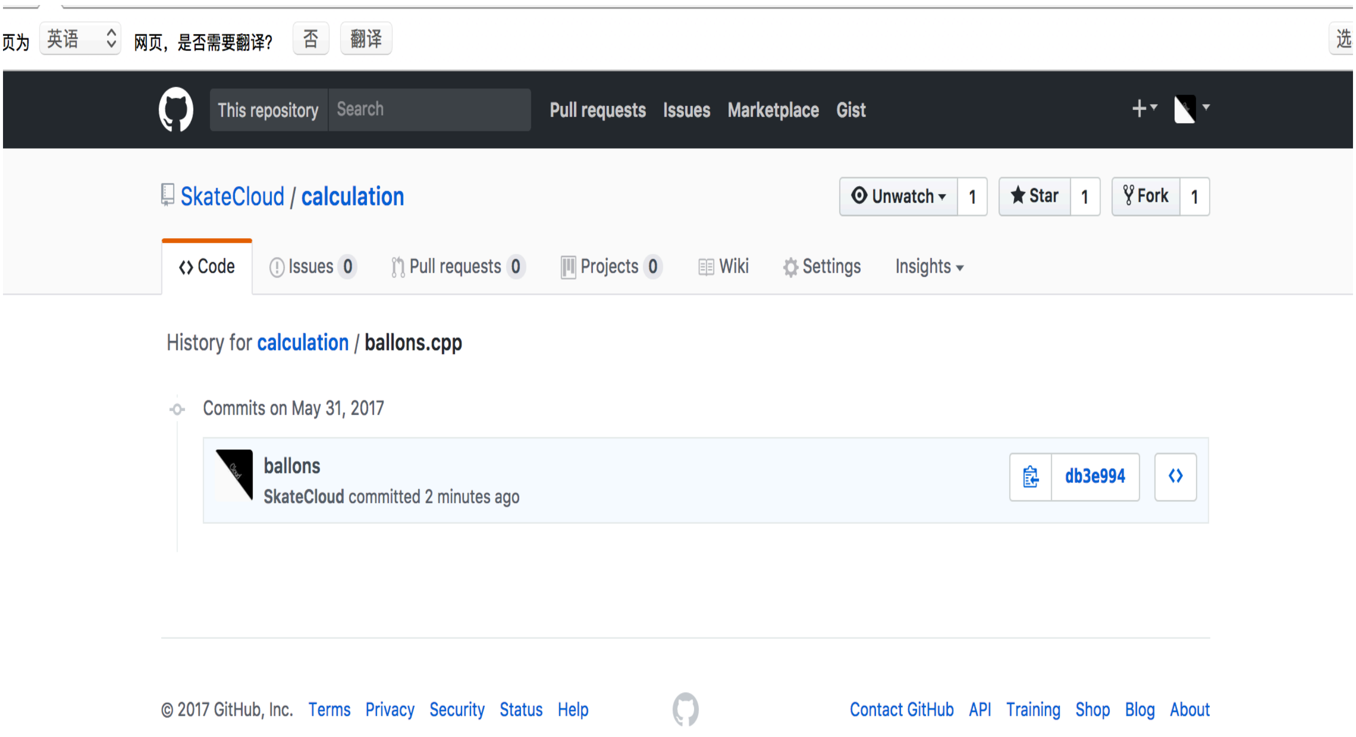
```
pc@Lenovo-PC MINGW64 /e/testgit (master)
$ git log
commit 16373c50e95d99a580591bf7c1bd0ab6856ed048 (HEAD -> master, origin/master)
Merge: 6d1afe5 abb1d6a
Author: syy <3401302509@qq.com>
Date:   Sat May 20 20:50:27 2017 +0800

    conflict fixed
```

Conclution

通过练习这个算法题，我对数学建模的认识更加深刻，规避各种不正确的情况，从而获得最终正确解。学习型团队是一个有着巨大潜力的团队，授人以鱼不如授人以渔。团队之间要相互交流，使团队之间相互学习共同提高，实现资源的共享。最后，一个团队要有一个领导核心。一个成功的团队领导者，除了专业能力要服人，更要懂得创造共同愿景，激励成员士气，并且让队员跟着你有成长的机会。领导核心是一个团队中必不可少的重要位置，是整个团

队中方向的领导者和决策者。正如大家所熟知的一个故事一样，“一头绵羊带领的一群狮子，敌不过一头狮子带领的一群绵羊”，一个组织的成败往往取决于组织的领导，领袖的魅力、魄力、预见力指引组织正确的目标和方向。我们通过这次的团队合作，充分的锻炼了项目经理的领导能力。

git log


页为 英语 网页，是否需要翻译? 否 翻译 选
This repository Search Pull requests Issues Marketplace Gist
Unwatch 1 Star 1 Fork 1
Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights
© 2017 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About

SkateCloud / calculation

History for calculation / ballons.cpp

Commits on May 31, 2017

ballons
SkateCloud committed 2 minutes ago
db3e994