实验三

1. **小希的数表**

**#include<bits/stdc++.h>**

**using namespace std;**

**typedef long long ll;**

**int n;**

**int a[110];**

**int s[110];**

**bool vis[110];**

**int main()**

**{**

**while(cin>>n&&n!=0)**

**{**

**memset(a,0,sizeof(a));**

**memset(s,0,sizeof(s));**

**int num=n\*(n-1)/2;**

**for(int i=1;i<=num;i++) cin>>s[i];**

**int cc=s[2]-s[1];//cc=a[3]-a[2]**

**for(int i=3;i<=num;i++)**

**{**

**//遍历可能的a[2]+a[3]**

**if((cc+s[i])%2) continue;**

**memset(vis,false,sizeof(vis));**

**bool flag=true;**

**vis[1]=vis[2]=vis[i]=true;**

**a[3]=(cc+s[i])/2;**

**a[1]=s[2]-a[3];**

**a[2]=s[1]-a[1];**

**for(int j=4;j<=n;j++)**

**{**

**int pos=0;**

**//找到现有的s中最小的 确定a[j]**

**for(int k=1;k<=num;k++)**

**{**

**if(vis[k]==false)**

**{**

**pos=k;**

**break;**

**}**

**}**

**if(pos)**

**{**

**vis[pos]=true;**

**a[j]=s[pos]-a[1];**

**//排除掉a[1]+a[j],a[2]+a[j],a[3]+a[j]~~**

**for(int k=1;k<j;k++)**

**{**

**int temp=a[j]+a[k];**

**bool ff=false;**

**for(int m=1;m<=num;m++)**

**{**

**if(s[m]==temp)**

**{**

**ff=true;**

**if(!vis[m])**

**{**

**vis[m]=true;**

**break;**

**}**

**}**

**}**

**if(!ff)**

**{**

**//如果s里面没有a[k]+a[j]这一元素**

**//就说明假设的这个a[2]+a[3]是错的**

**flag=false;**

**break;**

**}**

**}**

**//假设错误 立即跳出**

**if(!flag) break;**

**}**

**}**

**//假设正确 答案唯一 立即跳出**

**if(flag) break;**

**}**

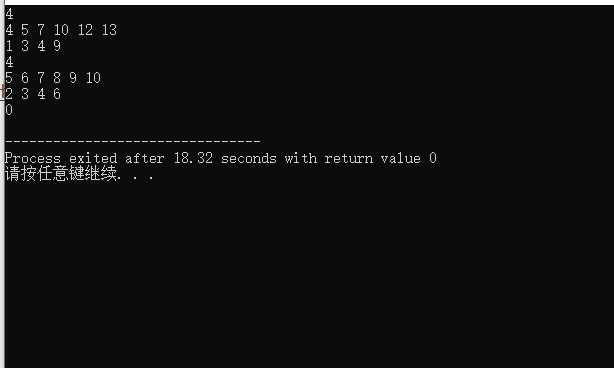
**for(int i=1;i<=n;i++) cout<<a[i]<<" ";**

**cout<<endl;**

**}**

**return 0;**

**}**



s[1]=a[1]+a[2]，s[2]=a[1]+a[3]，但是之后的 s[i] 就有很多种组合的可能了。

OK，现在我们知道a[1]+a[2]，和a[1]+a[3]，那就差一个a[2]+a[3]便可求出a[1]、a[2]、a[3]，知道这三个数后便可推理出其他数：删除掉s数组中的a[1]+a[2]，a[1]+a[3]，a[2]+a[3]，再找出s中的最小的数，这个数便是a[1]+a[4]，推理出a[4]。推理出a[4]后，再在s数组中删除a[1]+a[4]，a[2]+a[4]，a[3]+a[4]，再找最小元素，推出a[5]。

**2.变形课**

#include <iostream>

#include<string.h>

#include<stdio.h>

using namespace std;

char a[100][2];

int v[100];

int dfs(int n,int j)

{

int i;

if(a[j][1]=='m')

return 1;

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

{

if(a[i][0]==a[j][1]&&v[i]!=1)

{

v[i]=1;

if(dfs(n,i))

return 1;

}

}

return 0;

}

int main()

{

char b[1000];

int str;

int n,i=0,j;

while((scanf("%s",b))!=EOF)

{

memset(v,0,sizeof(v));

i=0;

if(strcmp(b,"0")!=0)

{

n=strlen(b);

a[i][0]=b[0];

a[i][1]=b[n-1];

i++;

}

if(strcmp(b,"0")==0)

continue;

while(1)

{

scanf("%s",b);

if(strcmp(b,"0")!=0)

{

n=strlen(b);

a[i][0]=b[0];

a[i][1]=b[n-1];

i++;

}

if(strcmp(b,"0")==0)

{

for(j=0; j<i; j++)

{

if(a[j][0]=='b')

{

v[j]=1;

str=dfs(i,j);

if(str==1)

{

cout<<"Yes."<<endl;

break;

}

}

}

if(j==i&&str==0)

{

cout<<"No."<<endl;

break;

}

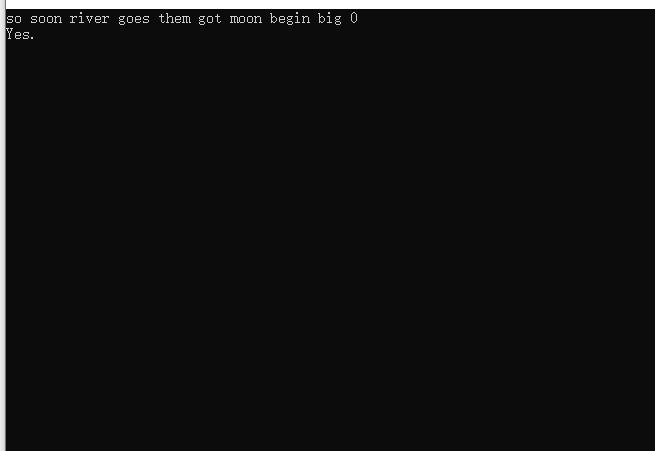
}

}

}

return 0;

}



使用递归的深度优先搜索算法（DFS）来查找字符串匹配关系。在函数 dfs 中，对于给定的字符串 b，首先检查是否以字符 'b' 开头，如果是，则开始进行 DFS。

在 DFS 中，它尝试查找下一个字符串是否以当前字符串的尾字符结束，如果是，则返回 1 表示匹配成功，否则递归地寻找下一个匹配的字符串。

循环处理： 在主循环中，不断读取输入，直到遇到单独的 "0" 表示结束。

读取一行字符串，并将首尾字符存储到 a 数组中，直到遇到 "0"。

针对以 'b' 开头的字符串，执行深度优先搜索 dfs。

如果在所有字符串中找不到以 'b' 开头的匹配序列，则输出 "No."，表示无法匹配。

3：

#include <iostream>

#include <cstring>

#include <algorithm>

using namespace std;

int maxXY; //X,Y中的较大数

int minXY; //X,Y中的较小数

int flagMax; //maxXY是否能得到

int flagMin; //minXY是否能得到

int vis[105]; //数字是否用过

//深搜，搜索minXY和maxXY是否能同时得到

int DFS(int n)

{

if (n == 1)

{

if (flagMax == 0)

{

flagMin = 1;

flagMax = 1;

if (DFS(maxXY))

return 1;

else

{

flagMax = 0;

return 0;

}

}

return 1;

}

for (int i = 2; i <= min(100, n); i++)

{

if (n % i == 0 && vis[i] == 0)

{

vis[i] = 1;

if (DFS(n / i))

return 1;

vis[i] = 0;

}

}

return 0;

}

int main()

{

int x, y;

while (cin >> x >> y)

{

memset(vis, 0, sizeof(vis));

flagMax = 0;

flagMin = 0;

maxXY = max(x, y);

minXY = min(x, y);

int flag = DFS(minXY);

if (flagMin == 0)

cout << maxXY << endl;

else if (flag == 1)

cout << maxXY << endl;

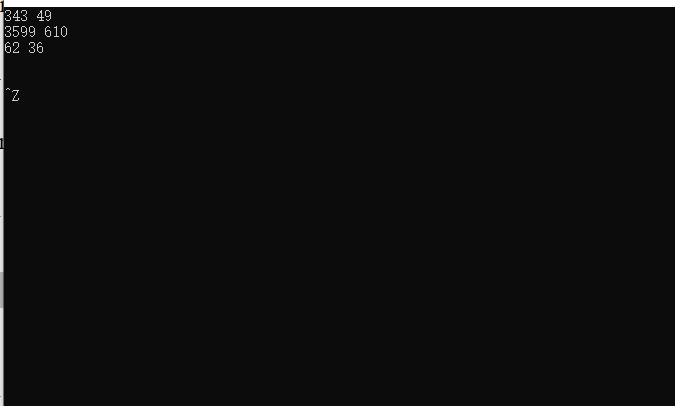
else

cout << minXY << endl;

}

return 0;

}



使用深度优先搜索（DFS）的方式，探索能够同时整除 minXY 和 maxXY 的最小公倍数。

如果当前数字 n 等于 1，表示找到了一个解。如果 flagMax 为 0，说明这是第一次找到解，标记 flagMin 和 flagMax 为 1，并继续搜索 maxXY。

在搜索过程中，尝试将 n 分解为素因数，递归搜索这些素因数的组合。标记 vis 数组表示某个素因数是否已经使用过，避免重复使用。

如果找到了解，则返回 1，否则回溯。

输出结果： 根据搜索结果输出符合条件的结果。

如果 flagMin 为 0，说明 minXY 无法得到解，直接输出 maxXY。

如果 flag 为 1，说明找到了解，直接输出 maxXY。

否则，输出 minXY。

1. **Tempter of the Bone**

**#include<iostream>**

**#include<cstring>**

**#include<cmath>**

**using namespace std;**

**int n,m,t;**

**int sx,sy,ex,ey,flag=0;**

**char mp[11][11];**

**int book[11][11];**

**int Next[4][2]={{1,0},{-1,0},{0,1},{0,-1}};**

**void dfs(int x,int y,int s)**

**{**

**if((t-s-(abs(x-ex)+abs(y-ey)))%2!=0) return;**

**/\***

**这里是本题剪枝的核心代码，可以看上面的解释多理解一下**

**\*/**

**if(x==ex&&y==ey)**

**{**

**if(s==t)**

**{**

**flag=1;**

**return ;**

**}**

**else return;**

**}**

**if(flag==1) return;//找到答案直接return出去，减少步骤。**

**for(int i=0; i<4; i++)**

**{**

**int tx=x+Next[i][0];**

**int ty=y+Next[i][1];**

**if(tx<0||tx>n-1||ty<0||ty>m-1) continue;**

**if(mp[tx][ty]=='X') continue;**

**if(mp[tx][ty]=='.'&&book[tx][ty]==0)**

**{**

**book[tx][ty]=1;**

**dfs(tx,ty,s+1);**

**book[tx][ty]=0;//注意返回标记，以便下次搜索**

**}**

**}**

**return ;**

**}**

**int main()**

**{**

**while(cin>>n>>m>>t)**

**{**

**flag=0;**

**if(n==0&&m==0&&t==0) break;**

**memset(book,0,sizeof(book));**

**memset(mp,'\0',sizeof(mp));**

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

**for(int j=0; j<m; j++)**

**{**

**cin>>mp[i][j];**

**if(mp[i][j]=='S')**

**{**

**sx=i,sy=j;**

**mp[i][j]=='.';**

**}**

**if(mp[i][j]=='D')**

**{**

**ex=i,ey=j;**

**mp[i][j]='.';**

**}**

**}//存入地图数据**

**book[sx][sy]=1;//标记起始点**

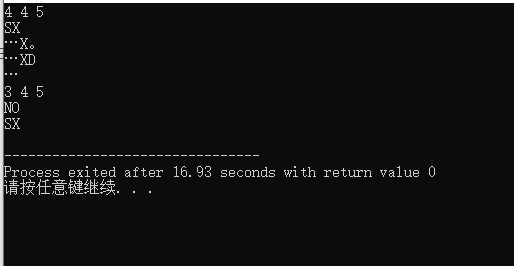
**dfs(sx,sy,0);**

**if(flag==1) cout<<"YES"<<endl;**

**else cout<<"NO"<<endl;**

**}**

**}**



DFS搜索： 使用深度优先搜索，递归地探索迷宫中的路径。搜索函数 dfs 接受当前位置 (x, y)、已经走过的步数 s 作为参数。

如果剩余时间 t - s - (abs(x - ex) + abs(y - ey)) 为奇数，说明剩余的时间无法满足到达目标点的条件，直接返回。

如果当前位置是目标点，并且已经走过的步数等于总时间，将 flag 设置为 1 表示找到解。

如果已经找到解，直接返回。

否则，尝试向四个方向移动，如果下一个位置是合法的、未被访问的空格，标记该位置为已访问，递归调用 dfs

**8.八球胜负问题**

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

int n,i,j;

int t;

int sum,v;

int r,y;

char c;

while (cin>>n,n)

{

//getchar();//过滤回车

cin.ignore();

r=y=0;

while ((c=getchar()) != '\n')

{

if (c == 'R')

r++;

else

{

if(c=='Y')

y++;

else

if(c=='B')

{

if(r != 7)

cout<<"Yellow"<<endl; //如果是红打进了黑球，要判断是否已将红球打完

else

cout<<"Red"<<endl;

}

else

{

if(y != 7) //如果是黄打进了黑球，要判断是否已将黄球打完

cout<<"Red"<<endl;

else

cout<<"Yellow"<<endl;

}

}

}

}

return 0;

}