

## 7.25 课堂笔记 26

### 本周作业 (7.31 之前提交)

1. 洛谷 P1540
2. 洛谷 P1071
3. 洛谷 P1125
4. 洛谷 P1097
5. 洛谷 P1051

### 课堂内容

1. P3951

赛瓦维斯特定理：已知  $a, b$  为大于 1 的正整数， $\gcd(a, b) = 1$ ，  
则使不定方程  $ax + by = C$  无负数解的最大整数  $C = ab - a - b$

不知道定理：找规律

留意数据范围应使用 long long

```
#include<iostream>
using namespace std;
long long a, b;

int main() {
    cin >> a >> b;
    cout << a*b - a - b;
    return 0;
}
```

2. P3958

函数传入参数为两个空洞的球心坐标，通过空间中两点距离公式求出球心距离，与空洞半径进行对比，如果两个空洞相通返回 true，不相通则返回 false

```

bool isTangency(LL x1, LL y1, LL z1, LL x2, LL y2, LL z2) {
    if((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2) + (z1 - z2)*(z1 - z2) <= 4 * r*r) {
        return true;
    }
    else{
        return false;
    }
}

```

并查集查询操作与合并操作，实现功能分别为查询某个空洞最终连通的空洞是哪一个与合并两个相连通的空洞

```

int find(int x) {
    if(x == fa[x]) {
        return x;
    }
    else{
        return find(fa[x]);
    }
}

void merge(int x, int y) {
    fa[find(x)] = find(y);
}

```

读入输入数据，建立并查集数组，初始化的时候预先将所有和底部连通与和顶部连通的空洞处理好

```

int main() {
    cin >> t;
    while (t--) {
        memset(x, 0, sizeof(x));
        memset(y, 0, sizeof(y));
        memset(z, 0, sizeof(z));
        cin >> n >> h >> r;
        for(int i = 0; i <= n + 1; i++) {
            fa[i] = i;
        }
        for (int i = 1; i <= n; i++) {
            cin >> x[i] >> y[i] >> z[i];
        }
        for (int i = 1; i <= n; i++) {
            if (h - z[i] <= r) merge(i, n + 1); //将与顶部相接触的洞与顶部合并
            if (z[i] <= r) merge(i, 0); //与底部合并
        }
    }
}

```

按顺序判断每一个空洞，如果找到一个与顶部连通的空洞那么输出 yes

```

        for (int i = 1; i <= n; i++) {
            for (int j = i + 1; j <= n; j++) {
                if (isTangency(x[i], y[i], z[i], x[j], y[j], z[j])) {
                    merge(i, j);
                }
            }
        }
        if (find(0) == find(n + 1)) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
}

```

完整代码：

```

#include<bits/stdc++.h>
typedef long long LL;
using namespace std;
int fa[1010];
long long x[1010], y[1010], z[1010];
int t, n;
long long h, r;

bool isTangency(LL x1, LL y1, LL z1, LL x2, LL y2, LL z2) {
    if((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2) + (z1 - z2)*(z1 - z2) <= 4 * r * r) {
        return true;
    }
    else {
        return false;
    }
}

int find(int x) {
    if(x == fa[x]) {
        return x;
    }
    else {
        return find(fa[x]);
    }
}

void merge(int x, int y) {
    fa[find(x)] = find(y);
}

int main() {
    cin >> t;
    while (t--) {
        memset(x, 0, sizeof(x));
        memset(y, 0, sizeof(y));
        memset(z, 0, sizeof(z));
        cin >> n >> h >> r;
        for(int i = 0; i <= n + 1; i++) {
            fa[i] = i;
        }
        for (int i = 1; i <= n; i++) {
            cin >> x[i] >> y[i] >> z[i];
        }
        for (int i = 1; i <= n; i++) {
            if (h - z[i] <= r) merge(i, n + 1); //将与顶部相接触的洞与顶部合并
            if (z[i] <= r) merge(i, 0); //与底部合并
        }
        for (int i = 1; i <= n; i++) {
            for (int j = i + 1; j <= n; j++) {
                if (isTangency(x[i], y[i], z[i], x[j], y[j], z[j])) {
                    merge(i, j);
                }
            }
        }
        if (find(0) == find(n + 1)) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
    return 0;
}

```

### 3. P5019

贪心策略:  $\text{if}(a[i+1] > a[i]) \{ \text{ans} += a[i+1] - a[i]; \}$

显然最深的坑需要填满, 在填满最深的坑的时候最深的坑左右的坑也被填满了, 这个代价是免费的, 范围是直到出现下降趋势即因为中间的某个坑已经填满不能再填外侧的坑, 所以从左往右扫描, 如果坑成上升趋势那么只需要支付最测最深的坑的代价就能一直填下去, 直到某个坑深度变深, 需要支付新的代价。

```

#include<bits/stdc++.h> //包含了所有C++头文件的头文件
//#include<iostream>
//#include<algorithm>
//#include<cstring>
//poj
using namespace std;

int main()
{
    int n;
    int a[100000+1];
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
    int ans=a[0];
    for(int i=0;i<n-1;i++)
    {
        if(a[i+1]>a[i])
        {
            ans+=a[i+1]-a[i];
        }
    }
    cout<<ans<<endl;

    return 0;
}

```

#### 4. P5020

题意为删除给出的数中所有可以由其他数组成的数，然后剩下的数数量即为输出

每个数只能由比它小的数字组成而不能被比它大的数字组成。

那么很显然，我们可以首先对数组排个序，然后对于每一个数考虑能不能被它前面的数字所组成。我们知道如果  $x$  能被前  $i$  个数组成且组成  $x$  的数当中包含  $a_i$  那么  $(x-a_i)$  也必然能被前  $i$  个数来组成，那么我们很容易想到定义  $dp[j]$  表示  $j$  能否被组成，那么根据上面的想法显然有  $dp[j]=dp[j]||dp[j-a[i]]$

```

#include<bits/stdc++.h>
using namespace std;
int t,n;
int a[105];
int dp[25010];

int main(){
    cin>>t;
    for(int i=1;i<=t;i++){
        cin>>n;
        int ans=n;
        for(int i=1;i<=n;i++) cin>>a[i];
        sort(a+1,a+1+n);
        memset(dp,0,sizeof(dp));
        dp[0]=1; //自己可以表示
        for(int i=1;i<=n;i++){
            if(dp[a[i]]){
                ans--;
                continue;
            }
            for(int j=a[i];j<=a[n];j++){ //删除无效货币
                dp[j]=dp[j]||dp[j-a[i]];
            }
        }
        cout<<ans<<endl;
    }
    return 0;
}

```

#### 5. P5657

观察输入数据输出数据找规律

```
#include <bits/stdc++.h>
using namespace std;
int n;
unsigned long long c;
int a[1001];

int main() {
    cin >> n >> c;
    c ^= (c / 2); //异或，找规律
    for(int i = 1; i <= n; i++) {
        a[i] = c % 2;
        c = c / 2;
    }
    for(int i = n; i >= 1; i--) {
        cout << a[i];
    }
    return 0;
}
```

## 初赛答案

S-9

一

D D B D A  
B C D A B  
B D C B A

二

√ √ × × B A  
× √ C A B D  
√ × × √ B C A

三

A A C B A  
B C A B B