

课堂内容

1、P1311

枚举、模拟、暴力的解法（60分）

```
#include<bits/stdc++.h>
using namespace std;
int n,k,p;
int color[200010],price[200010];
int cnt;
int c;

int main(){
    cin >> n >> k >> p;
    for(int i=1; i<=n; i++){
        cin >> color[i] >> price[i];
    }

    //遍历客栈b, c表示b前面离它最近的最低消费客栈（包括它自己）
    //那么客栈a一定在 c前面（包括c），统计颜色与b相同的a即可
    for(int i=1; i<=n; i++){
        if(price[i]<=p){//b与c重合,统计之前的同色个数即可
            for(int j=1; j<i; j++){
                if(color[i] == color[j]){
                    cnt++;
                }
            }
        }
        else{
            for(int j=i-1; j>=1; j--){
                if(price[j] <= p){
                    c = j;
                    break;
                }
            }
            for(int j=1; j<=c; j++){
                if(color[j] == color[i]){
                    cnt++;
                }
            }
            c = 0;
        }
    }
    cout << cnt << endl;
    return 0;
}
```

100 分代码

```
#include<bits/stdc++.h>
using namespace std;
int n,k,p;
int color[200010],price[200010];
int cnt[60];
int temp;
int res;

int main(){
    cin >> n >> k >> p;
    for(int i=1; i<=n; i++){
        cin >> color[i] >> price[i];
    }
    //遍历客栈b, c表示b前面离它最近的最低消费客栈（包括它自己）
    //那么客栈a一定在 c前面（包括c）
    for(int i=1; i<=n; i++){
        if(price[i]<=p){
            for(int j=temp+1; j<i; j++){
                cnt[color[j]]++; //上一次统计结束的位置"到b之间的各颜色客栈数量
            }
            res += cnt[color[i]];
            temp = i; //更新统计到的位置
            cnt[color[i]]++; //b客栈的此位置额外统计一次
        }
        else{
            res += cnt[color[i]];
        }
    }
    cout << res << endl;
    return 0;
}
```

2、P1314

算法：二分、前缀和

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define maxn 200005
4 #define ll long long
5 int n,m;
6 ll S,a;
7 int w[maxn],v[maxn],l[maxn],r[maxn];
8 ll sum[maxn],cnt[maxn];
9
10 bool judge(int W){
11     for(int i=1;i<=n;i++){
12         if(w[i]>=W){
13             sum[i] = sum[i-1] + v[i]; //矿石的重量大于参考值，我们才要求其价值之和
14             cnt[i] = cnt[i-1] + 1;
15         }
16         else{
17             sum[i] = sum[i-1]; //矿石的重量小于参考值
18             cnt[i] = cnt[i-1];
19         }
20     }
21     ll tmp=0;
22     for(int i=1;i<=m;i++){
23         tmp += (cnt[r[i]]-cnt[l[i]-1])*(sum[r[i]]-sum[l[i]-1]); //tmp就是检验值Y
24     }
25     a = llabs(S-tmp); //保存 |S-tmp|
26     if(tmp < S) return true;
27     else return false;
28 }
29 }
```

```

30 int main(){
31     cin >> n >> m >> S;
32     int mx=0;
33     for(int i=1;i<=n;i++){
34         cin >> w[i] >> v[i];
35         mx=max(w[i],mx); //保存最大重量
36     }
37     for(int i=1;i<=m;i++){
38         cin >> l[i] >> r[i];
39     }
40     int l=0,r=mx+1; //矿石全选和全不选的情况
41     ll ans=0x3f3f3f3f3f3f3f3f; //Long Long 类型的无穷大
42     while(l<=r){ //更新参考重量W
43         int mid=(l+r)/2;
44         if(judge(mid))r=mid-1;//注意二分的取值
45         else l=mid+1;
46         ans=min(ans,a);// 更新|S-tmp|的最小值
47     }
48     cout << ans;
49     return 0;
50 }

```

3、P1080

算法：高精度乘法、贪心

高精度乘法：

```

15 void chengfa(int X[],int v,int Y[]){
16     int len=X[0];
17     for (int i=1; i<=len+20; i++){//Y[0]不能清空
18         Y[i]=0;
19     }
20     for (int i=1; i<=len; i++){
21         //因为a、b小于10000，故每一位不会超过范围，直接拿v与每一位相乘
22         Y[i] += (X[i]*v);
23         Y[i+1] += (Y[i]/10); //先进位
24         Y[i]%=10; //再求余
25     }
26     while(Y[len+1]>0){ //判断最后一位需不需要进位
27         len++;
28         Y[len+1]+=(Y[len]/10);
29         Y[len]%=10;
30     }
31     Y[0]=len; //保存当前长度
32 }

```

高精度除法

```
34 void chufa(int X[],int v,int Y[]){
35     int len=X[0];
36     int rest=0;
37     for (int i=len; i>=1; i--) {
38         rest = rest*10+X[i];
39         Y[i] = rest/v;
40         rest -= (v*Y[i]); //求余数
41     }
42     while (!Y[len] && len>0) len--; //删除前导0
43     Y[0]=len;
44 }
```

完整代码：

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  struct dacheng{
4      int a,b;
5  } p[1010];
6  int t[1000000]; //存每次乘法的结果
7  int d[1000000]; //暂存的中间数组
8  int ans[1000000]; //存每次除法后的最大奖励数
9  int n;
10
11 bool cmp(dacheng x, dacheng y){
12     return x.a*x.b<y.a*y.b;
13 }
14
15 void chengfa(int X[],int v,int Y[]){
16     int len=X[0];
17     for (int i=1; i<=len+20; i++){//Y[0]不能清空
18         Y[i]=0;
19     }
20     for (int i=1; i<=len; i++){
21         //因为a、b小于10000，故每一位不会超过范围，直接拿v与每一位相乘
22         Y[i] += (X[i]*v);
23         Y[i+1] += (Y[i]/10); //先进位
24         Y[i]%=10; //再求余
25     }
26     while(Y[len+1]>0){ //判断最后一位需不需要进位
27         len++;
28         Y[len+1]+=(Y[len]/10);
29         Y[len]%=10;
30     }
31     Y[0]=len; //保存当前长度
32 }
```



```

34 void chufa(int X[],int v,int Y[]){
35     int len=X[0];
36     int rest=0;
37     for (int i=len; i>=1; i--) {
38         rest = rest*10+X[i];
39         Y[i] = rest/v;
40         rest -= (v*Y[i]); //求余数
41     }
42     while (!Y[len] && len>0) len--; //删除前导0
43     Y[0]=len;
44 }
45
46 void _max(int X[],int Y[]){//比较Y和X的大小, 更新获奖最多数
47     if(Y[0]<X[0]) return;
48     if(Y[0]==X[0])
49         for(int i=Y[0]; i>=1; i--)
50             if (Y[i]<X[i]) return;
51             else if (Y[i]>X[i]) break;
52     for(int i=0; i<=Y[0]; i++) X[i]=Y[i]; //若Y大则把Y赋值给x
53 }
54
55 int main(){
56     cin >> n;
57     for(int i=0; i<=n; i++){
58         cin >> p[i].a >> p[i].b;
59     }
60     sort(p+1,p+n+1,cmp);//按照a*b从小到大排序
61     t[0]=1;//初始长度1
62     t[1]=1;//乘法计算, 起始值为1
63     for(int i=1; i<=n; i++){
64         chengfa(t, p[i-1].a, d);
65         for(int j=0; j<=d[0]; j++){
66             t[j] = d[j];//将每一次乘法计算的结果保存到t数组进行下一次计算
67         }
68         chufa(t, p[i].b, d);
69         _max(ans,d);//更新最大值
70     }
71     for(int i=ans[0]; i>=1; i--)//逆序输出
72         cout << ans[i];
73     return 0;
74 }

```

4、P1083

算法：二分、差分数组、前缀和

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int _n = 0;
4  int _m = 0;
5  int _room[1000030]; //每天的可用教室数量
6  int _head[1000030]; //每次租借的起始时间
7  int _tail[1000030]; //每次租借的末尾时间
8  int _num[1000030]; //每次租借的数量
9  long long _dif[1000030]; //差分数组
10 long long _need[1000030]; //最终每天需求的教室数量
11 void ParseIn() {
12     cin >> _n >> _m;
13     for(int i = 1; i <= _n; i++) {
14         cin >> _room[i];
15     }
16     for(int i = 1; i <= _m; i++) {
17         cin >> _num[i] >> _head[i] >> _tail[i];
18     }
19 }
20
21 bool Check(int rent) {
22     memset(_dif, 0, sizeof(_dif));
23     //利用差分数组依次将每天对应需要的教室累加，然后和当天实际可用的教室对比
24     for(int i = 1; i <= rent; i++) {
25         _dif[_head[i]] += _num[i]; //差分数组头部和前面拉开的差距变大
26         _dif[_tail[i] + 1] -= _num[i]; //差分数组尾部的下一个和尾部的差距变小
27     }
28     for(int i = 1; i <= _n; i++) {
29         //求每一天教室需求总量
30         _need[i] = _need[i - 1] + _dif[i];
31         //依次和当天实际可用教室对比
32         if(_need[i] > _room[i]) {
33             return false;
34         }
35     }
36     return true;
37 }
```

二分目的：快速查找，能提高效率

```
39 //如何快速找到需要修改的订单：二分
40 void Core () {
41     int l = 1;
42     int r = _m;
43     int mid = 0;
44     int ans = 0;
45     //先算全部订单的情况，如果教室够用，则不需要二分
46     if(Check(_m)) {
47         cout << "0";
48         return;
49     }
50     while(l <= r) {
51         mid = (l + r) / 2; //mid表示教室够用的最大订单数
52         if(Check(mid)){
53             ans = mid;
54             l = mid + 1;
55         }
56         else {
57             r = mid - 1;
58         }
59     }
60     //因为二分求出来的是教室够用的最大订单数，故不满足的要 +1
61     cout << "-1" << endl << ans + 1;
62 }
63
64 int main(){
65     ParseIn();
66     Core();
67     return 0;
68 }
```

5、P1970

算法：动态规划

动态规划转移方程：

$f[i][0]$ 表示此花比前一株花矮， $f[i][1]$ 表示此花比前一株花高

$i=1$ 时 $f[i][0] = f[i][1] = 1;$

$a[i] > a[i - 1]$ 时 $f[i][1] = \max(f[i][1], f[i - 1][0] + 1);$

$a[i] < a[i - 1]$ 时 $f[i][0] = \max(f[i][0], f[i - 1][1] + 1);$

$a[i] = a[i - 1]$ 时 $f[i][0] = f[i-1][0], f[i][1] = f[i-1][1]$

完整代码:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int n, a[1000010], f[1000010][2];
4  int main() {
5      cin >> n;
6      for(int i = 1; i <= n; i++) {
7          cin >> a[i];
8          if(i == 1){
9              f[i][0] = f[i][1] = 1;
10             }
11             //f[i][0]表示此花比前一株花矮, f[i][1]表示此花比前一株花高
12             else {
13                 f[i][0] = f[i - 1][0];
14                 f[i][1] = f[i - 1][1];
15                 if(a[i] > a[i - 1]) {
16                     //选此花, 继承f[i - 1][0]并且+1
17                     //不选此花, 继承f[i - 1][1]
18                     f[i][1] = max(f[i][1], f[i - 1][0] + 1);
19                 }
20                 if(a[i] < a[i - 1]){
21                     //与上同理
22                     f[i][0] = max(f[i][0], f[i - 1][1] + 1);
23                 }
24             }
25         }
26         cout << max(f[n][1], f[n][0]) << endl;
27         return 0;
28     }
```

6、P1351

30 分暴力求解

```
1  #include<bits/stdc++.h>
2  #define mod 10007
3  using namespace std;
4  int n, w[110], _map[110][110], sum, mx;
5  int u, v;
6  int main(){
7      cin >> n;
8      memset(_map, 0x3f, sizeof(_map));
9      for(int i=1; i<n; i++){
10         cin >> u >> v;
11         _map[u][v] = _map[v][u] = 1;
12     }
13     for(int i=1; i<=n; i++) cin >> w[i];
14     for(int k=1; k<=n; k++)
15         for(int i=1; i<=n; i++)
16             for(int j=1; j<=n; j++)
17                 if(i!=j && i!=k && j!=k){
18                     _map[i][j]=min(_map[i][j], _map[i][k]+_map[k][j]);
19                 }
```



```

20     for(int i=1;i<=n;i++)
21         for(int j=1;j<=n;j++){
22             if(_map[i][j]==2){
23                 sum=(sum+w[i]*w[j])%mod, mx=max(mx,w[i]*w[j]);
24             }
25         }
26     cout << mx << " " << sum;
27     return 0;
28 }

```

100 分解法:

枚举中转点 i

如果中转点 i 跟 a,b 两个点相连, 那么贡献值 $2ab = (a+b)^2 - (a^2+b^2)$

如果中转点 i 跟 a,b,c 三个点相连, 那么贡献值 $2ab+2ac+2bc = (a+b+c)^2 - (a^2+b^2+c^2)$

.....

所以令 t1 代表和的平方, t2 代表平方和, 联合权值之和 $ans = ans + t1 - t2$

令 max1、max2 分别代表中转点相连的最大、次大点权值

则最大联合权值 $maxx = \max(maxx, max1*max2)$

代码:

```

10 #include<bits/stdc++.h>
11 using namespace std;
12 struct edge{
13     int next;
14     int to;
15 }e[400010];
16 int numEdge,head[200010],w[200010];
17 int n,ans,maxx;
18 int u,v;
19 int max1, max2;//max1是此点相连的最大点权值, max2是此点相连的次大点权值
20 int t1, t2;//t1代表和的平方, t2代表平方和
21
22 void add(int from, int to){//邻接表存边
23     numEdge++;
24     e[numEdge].next = head[from];
25     e[numEdge].to = to;
26     head[from] = numEdge;
27 }
28
29 int main(){
30     cin >> n;
31     for(int i=1;i<n;i++){
32         cin >> u >> v;
33         add(u,v);
34         add(v,u);
35     }
36     for(int i=1;i<=n;i++){
37         cin >> w[i];
38     }

```

```

39  for(int i=1;i<=n;i++){
40      max1=0;
41      max2=0;
42      t1=0;
43      t2=0;
44      for(int j=head[i]; j!=0; j=e[j].next){
45          int y = e[j].to;
46          if(w[y]>max1){ //更新最大值和次大值
47              max2 = max1;
48              max1 = w[y];
49          }
50          else if(w[y]>max2){ //只更新次大值
51              max2 = w[y];
52          }
53          t1 = (t1+w[y])%10007; //把此中转点相连的所有点权值加起来
54          t2 = (t2+w[y]*w[y])%10007; //把此中转点相连的所有点权值平方加起来
55      }
56      t1 = t1*t1%10007;
57      ans = (ans+t1-t2+10007)%10007; //更新联合权值之和
58      if(maxx < max1*max2){ //更新最大联合权值
59          maxx = max1*max2;
60      }
61  }
62  cout << maxx << " " << ans;
63  return 0;
64  }

```