FZOI NOIP2018模拟赛题解

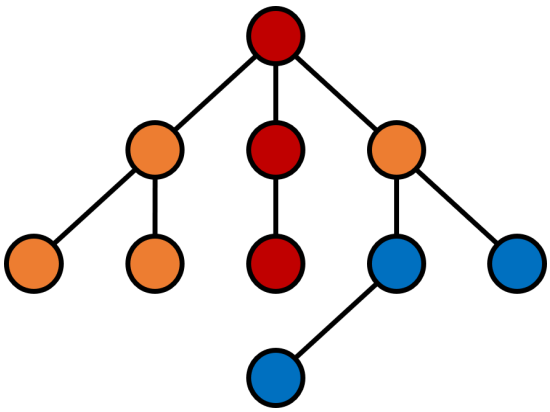
DAY1

[牛客网NOIP2018赛前集训营]中国式家长(parents)

直接暴力模拟，就可以得到$100pts$了。

[牛客网NOIP2018赛前集训营]随机生成树(tree)

题目说得很玄学，但是事实上给个例子就能够很明白了：



这棵树有5个联通块，如果你知道为什么的话，说明你已经理解了题意。

我们发现，如果把一个点连到和它颜色相同的点上取话，是不会增加联通块的个数的，所以我们要尽量使得一个点的颜色和它的父亲与它的儿子不同。

而我们又发现:1号点一定是根节点，因为它没有父亲。

所以我们可以将与1号点颜色不同的点连在1的下面。

现在剩下与1颜色相同的点，我们要将他们尽可能连在1的儿子上，暴力枚举是否有合法的儿子让这些点连即可。

纯暴力$O(n^2)$只能得到$60pts$，但是联想到素数的欧式筛法，我们可以这样子来枚举：

```
for(int i=1;i<=n;++i)
    for(int j=i;j<=n;j+=i)
```

这样枚举的复杂度为$O(\sum_{i=1}^{n}\frac{n}{i})=O(n\ log\ n)$，就可以得到$100pts$了。

[牛客网NOIP2018赛前集训营]洞穴(cave)

一种很显然的想法是直接暴力模拟，这样可以得到$50pts$。

现在来考虑正解：

我们假设$dp[i][k][j]$表示从i点开始走k步能否走到点j，那么转移方程就是：

$$dp[i][k][j]=max(dp[i][k-1][w]\ \&\ dp[w][1][j])\ \ (1\leq w\leq n)$$

事实上这里的取max就相当于位运算中的或

这样的状态设计，时间复杂度和空间复杂度都是：$O(n^2maxl)$的，期望得分$\leq 50pts$，考虑怎么去优化。

答案就是倍增。

我们假设$dp[i][k][j]$表示从i点开始走2^k步能否走到点j，那么转移方程就是：

$$dp[i][k][j] = max(dp[i][k-1][w] \ \& \ dp[w][k-1][j]) \ \ (1 \leq w \leq n)$$

这样我们就将复杂度变成了$O(n^2 log\ maxl)$，这是一个很大的优化，我们先预处理完dp数组，然后询问的时候就利用倍增模拟这一个过程，从而就可以AC本题了。

给出代码：

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
const int MAXN=105;
int n,m,u,v,q,l;
bool dp[MAXN][32][MAXN];
bool now[MAXN],nex[MAXN];
int read(){
    int x=0,f=1;char c=getchar();
    for(;c<'0'||c>'9';c=getchar()) if(c=='-') f=-1;
    for(;c>='0'&&c<='9';c=getchar()) x=(x<<3)+(x<<1)+c-
'0';
    return x*f;
}
int main(){
    n=read();m=read();
    while(m--){
        u=read();v=read();
        dp[u][0][v]=true;
    }
    for(int i=1;i<32;++i)
        for(int j=1;j<=n;++j)
            for(int k=1;k<=n;++k)
                for(int to=1;to<=n;++to)
```

```
                    dp[j][i][to]=dp[j][i][to]|(dp[j][i-1]
[k]&&dp[k][i-1][to]);
    q=read();
    while(q--){
        l=read();u=read();v=read();
        memset(now,0,sizeof(now));
        now[u]=true;
        for(int w=0;(1ll<<w)<=l;++w){
            if((l&(1<<w))==0) continue;
            memset(nex,0,sizeof(nex));
            for(int i=1;i<=n;++i){
                if(now[i]){
                    for(int j=1;j<=n;++j)
                        nex[j]=nex[j]|dp[i][w][j];
                }
            }
            for(int i=1;i<=n;++i) now[i]=nex[i];
        }
        if(now[v]) printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```

事实上还可以用$bitset$优化这整个过程（压掉j这一维），大家有兴趣可以自行了解。

DAY 2

[牛客网NOIP2018赛前集训营]区间(interval)

首先：$O(n^2)$带上log暴力枚举$40pts$是显然的。

当然你要$O(n^3)$加上log也是可以的。

现在考虑~~正~~解：

我们假设$Left[i]$表示从i开始往左的最长的满足区间gcd等于$arr[i]$区间的长度，$Right[i]$表示从i开始往右的最长的满足区间gcd等于$arr[i]$区间的长度，则最后答案就是$max\{Left[i]+Right[i]-1\}(1\le i\le n)$。

怎么处理出$Left$和$Right$数组呢?

以，$Left$数组为例，我们发现：假设区间$[l,r]$的gcd等于$arr[i](l\le i\le r)$的话e,则有$\forall j\in[l,r]$均有$arr[i]\mid arr[j]$，然后我们在向左枚举时，我们发现，如果$arr[j]\mid arr[i]$的话，那么$gcd(arr[w])(j-Left[j]+1\le w\le i)$一定等于$arr[i]$，，所以我们在访问到$j$的时候就可以直接跳到$j-Left[j]$继续判断了。

$Right$数组同理，这样均摊下来复杂度就是$O(n\ log\ n)$的，就可以得到$100pts$了。

然而事实上这道题目有更加优秀的$O(n)$的解法，大家可以自行探究。

[牛客网NOIP2018赛前集训营]串串(string)

直接暴力枚举S和T的组成可以得到$20pts$。

事实上我们可以想到：S就是由T加上$a-c$个0，$b-d$个1得到的。于是我们可以考虑构造出这个T来，然后再往T中加元素，那么最终的答案就是：

$$ans=\sum\text{每一个T加元素的方案}$$

但是如果直接往T中加元素，会出现重复的情况，比如说010要插入一个1，那么我有4中方法，但是最终只能得到3个不同的S串，所以我们为了使得添加的情况不重复，我们就规定，当我们不在最后面添加元素时，所有的0只能添加在1的前面，所有的1只能添加在0的前面。

所以我们可以枚举有多少个0和1被放在了最后，然后剩下的往前插，往前插的部分就是一个插板法的简单应用了。

而这样子考虑之后，我们发现每一个T加元素的方案只与0和1的个数有关，所以最终的答案就是：

$$ans=\binom{c+d}{c}\sum_{i=0}^{a-c}\sum_{j=0}^{b-d}\binom{i+j}{i}\binom{a-c-i+d-1}{a-c-i}\binom{b-d-j+c-1}{b-d-j}$$

其中$\binom{n}{m}$就表示C_n^m。

值得注意的是：有可能c或d等于0，这个时候我们就不能无脑算，它的方案数应该是0，这个情况特判一下即可。

[牛客网NOIP2018赛前集训营]旅游(travel)

这很类似于一个一笔画的问题，但关键就是有奇点，然后有一些边必须要走两次及以上，并且我们要使得重复走的边的边权总和最小，而我们可以证明，重复走的边一定在这个图的最小生成树上。

下面给出证明：

我们知道：$\sum_{i=1}^{k}2^i<2^{k+1}$，所以假设从点$u$到点$v$在最小生成树上的路径长度为$len_1=\sum_{i=1}^{n}2^{a_i}$。

假设在最小生成树外有一条路径使得$len_2 = \sum_{i=1}^{m} 2^{b_i} < len_1$，则必然存在$i, j$使得$2^{a_i} > 2^{b_j}$

而对于最小生成树，有一个性质是：对于任意两点u, v，它们在最小生成树上的路径是它们在这个图中的所有的路径中最大的边权最小的，所以我们得出的结论就与最小生成树矛盾了。

这样，我们就说明了：重复走的边一定在这个图的最小生成树上。

而重复走边的过程，就相当于加边的过程，最终要使得每个点的度都是偶数，所以我们只需要再开一个数组记录每个结点的度，最后在最小生成树上进行调整即可。

调整部分的代码如下：

```
void dfs(int nv,int father){
    int temp;
    for(edge *p=con[nv];p;p=p->nex){
        if(p->v==father) {temp=p->val;continue;}
        dfs(p->v,nv);
    }
    if(du[nv]&1){
        ++du[nv];++du[father];
        res[++cnt]=temp; //res是答案数组
    }
}
```