Dynamic Graph Matching

一. 考察内容：

动态规划 状态压缩

二. 题目分析：

[题目大意]

题目要求实现动态图匹配，对于一个有n个节点（n<=10）的图，支持加边删边操作，每次操作之后求出图上匹配数为1~n/2的匹配方案数。

[写题思路]

题目要求对每次操作求出所有匹配的方案数，显然需要维护很多信息，用一些线段树或者二分一类的“奇技淫巧”显然是不行的，我们考虑一种最朴素的算法，状态压缩动态规划，其实状压DP等同于枚举，将每种情况记录下来，用之前的答案得到之后的答案。

设状态f[i][j]为执行完前i个操作之后，匹配点集为j时的方案数。我们来考虑每次操作影响了什么。

首先考虑加边操作，添加一条边，我们不妨将这条边设为a—b，很显然，如果有一个不包含a、b节点的匹配点集S，那么加上这条边加入S，原本匹配点集S的方案同样可以用于匹配点集S+a+b，所以f[i][S+a+b]+=f[i-1][S]；

我们再考虑删除操作，同理，我们不妨将这条边设为a—b，则我们删除这条边，原来匹配点集S的方案不再能用于S+a+b（当然，可能a和b之间还有别的边，但是由于匹配边不同的方案属于不同的匹配，所以该转移仍然成立），我们需要减去之前加上的方案数f[i][S+a+b]-=f[i-1][S]。

通过这两种转移，我们可以得到对于每种匹配点集的方案数，我们先预处理出每个匹配点集所对应的匹配数，将这些方案数归纳入各自对应的匹配数中，最后输出即可。

另外由于这道题对格式要求相当严格，所以每行最后一个字符不能有多余的空格。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 09 01
*文件类型：源代码文件
*题目来源：HDU
*当前状态：已通过
*备忘录：动态规划
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 11
#define MAXM 30001
#define MOD 1000000007
int n,m,f[2][1<<MAXN],ans[MAXN],num[1<<MAXN];
struct OP
```

```
{
    int op,a,b;
}op[MAXM];
/* Variable explain:

*/
void read()
{
    char ls1[10];
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;++i)
    {
        scanf("%s%d%d",ls1,&op[i].a,&op[i].b);
        op[i].op=ls1[0]=='+'?1:-1;
        --op[i].a,--op[i].b;
    }
    return;
}
int add(int a,int b)
{
    return ((long long)a+b+MOD)%MOD;
}
void dp()
{
    int U=(1<<n)-1;
    for(int i=0;i<=U;++i)f[0][i]=f[1][i]=0;
    for(int i=1;i<=n;++i)ans[i]=0;
    //设状态f[T][i]:第T个操作之后，将集合i中的点设为匹配点的方案数
    //实现时考虑采用滚动数组
    f[0][0]=1;
    for(int i=1;i<=m;++i)
    {
        int node=(1<<op[i].a)|(1<<op[i].b);
        for(int j=1;j<=n;++j)ans[j]=0;
        for(int j=0;j<=U;++j)f[i&1][j]=f[(i&1)^1][j];
        for(int j=0;j<=U;++j)
        {
            if((j&node)==0) f[i&1][j^node]=add(f[i&1][j^node],f[(i&1)^1]
[j]*(op[i].op==1?1:-1));  //加上原来不包含的a,b的集合的匹配数
            ans[num[j^node]]=add(ans[num[j^node]],f[i&1][j^node]);
        }
        for(int i=2;i<=n;i+=2)printf("%d%c",ans[i],i==n?'\n':' ');
    }
}
void init()
{
    for(int i=1;i<=1024;++i)
    {
        int _i=i;
        while(_i)
        {
            if(_i&1)++num[i];
            _i>>=1;
        }
    }
}
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    int T;
    scanf("%d",&T);
    init();
    // printf("%d\n",num[15]);
    while(T--)
    {
        read();
        dp();
    }

    return 0;
}
```