CF 663 div2

A.Suborrays

题意：略

题解：输出1,2,...,n即可

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
#include<algorithm>
#include<vector>
#include<queue>
using namespace std;
int main()
{
    int t;
    ios::sync_with_stdio(0);
    cin >> t;
    while(t--)
    {
        int n, m;
        cin >> n;
        for (int i = 1; i <= n;i++)
            cout << i << " ";
        cout << endl;
    }
    return 0;
}
```

B.Fix You

题意：给定矩阵，有很多字母,代表在这个格子上只能往某个方向上移动（只有向右和向下）。现要求棋盘上所有的点都能到达右下角的终点,问最少修改几个可以达成。输出次数。

题解：既然是全都向右或者向下，那么每个格子走一步一定离终点更近一步，不会出现环什么的，唯一的情况就是右边界不能再往右走，下边界不能再往下走，统计一下即可。

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
```

```
#include<algorithm>
#include<vector>
#include<queue>
using namespace std;
char pic[256][256];
int main()
{
    int t;
    ios::sync_with_stdio(0);
    cin >> t;
    while(t--)
    {
        int n, m;
        cin >> n >> m;
        //getchar();
        for (int i = 1; i <= n;i++)
            cin >> pic[i];
        int ans = 0;
        for (int i = 0; i < m;i++)
            if (pic[n][i] == 'D')
                ans++;
        for (int i = 1; i <= n;i++)
            if (pic[i][m-1] == 'R')
                ans++;
        cout << ans << endl;
    }
    return 0;
}
```

C.Cyclic Permutations

题意：对于一个长度为n的排列，里面的每一个元素，向他左边最近且比他大的元素连边，以及右边最近且比他大的元素连边。问所有长度为n的排列里，至少包含一个简单环的排列有多少个。

题解：本题初看没什么思路，然后画了下图，发现如果有无环出现，需要元素递增至n再降下去，所以最后答案就是$n! - 2^{n-1}$

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
#include<algorithm>
#include<vector>
#include<queue>
using namespace std;
typedef long long ll;
```

```
const ll MOD = 1e9 + 7;
int main()
{
    int n;
    cin >> n;
    ll ans = 1,tem = 1;
    for (int i = 1; i < n;i++)
    {
        ans *= i;
        ans %= MOD;
        tem *= 2;
        tem %= MOD;
    }
    ans = (ans * n) % MOD;
    cout << (ans - tem + 2 * MOD) % MOD;
    return 0;
}
```

D.505

题意：给定一个01矩阵，要求所有边长为偶数的正方形子矩阵里面所包含的1的个数都是奇数，问最少修改多少给定矩阵的元素，才可达到要求。

题解：发现对于4*4的矩阵是没有解的，于是只要给定矩阵的两边长都大于4，则无解，否则，短边长度只有可能是2或者3，所以可以用类似状压dp的方法求解答案$dp[i][j]$表示考虑到长边的$1-i$位，最后一排的状态是j□复杂度$O(\max(m,n)\ * 2^3)$

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
#include<algorithm>
#include <stdio.h>
#include<vector>
#include<queue>
using namespace std;
const int MAX = 1e6 + 20;
int n, m;
char pic[8][MAX];
int dp[MAX][12];
int main()
{
    cin >> n >> m;
    memset(dp, 0x3f, sizeof(dp));
    if ((n >= 4 && m >= 4))
    {
        puts("-1");
        while(getchar()!=EOF);
        return 0;
```

```
    }
    else if (n == 1 || m == 1)
    {
        puts("0");
        while(getchar()!=EOF);
        return 0;
    }
    if (n < m)
    {
        for (int i = 1; i <= n;i++)
            scanf("%s", pic[i] + 1);
    }
    else
    {
        getchar();
        for (int i = 1; i <= n;i++)
        {
            for (int j = 1; j <= m;j++)
            {
                pic[j][i] = getchar();
            }
            getchar();
        }
        swap(n, m);
    }
    int maxdp = n == 3 ? 7 : 3;
    for (int i = 0; i <= 8;i++)
        dp[0][i] = 0;
    for (int i = 1; i <= m;i++)
    {
        for (int j = 0; j <= maxdp;j++)
        {
            int tem = 0;
            for (int k = 1; k <= n;k++)
                tem = (tem << 1) + pic[k][i] - '0';
            tem ^= j;
            int temans = 0;
            while(tem)
            {
                temans+=tem&1;
                //tem &= -tem;
                tem >>=1;
            }
            if (maxdp == 3)
            {
                if (j == 0)
                    dp[i][j] = min(dp[i - 1][2], dp[i - 1][1]) + temans;
                if (j == 1)
                    dp[i][j] = min(dp[i - 1][3], dp[i - 1][0]) + temans;
```

```
                if (j == 2)
                    dp[i][j] = min(dp[i - 1][3], dp[i - 1][0]) + temans;
                if (j == 3)
                    dp[i][j] = min(dp[i - 1][1], dp[i - 1][2]) + temans;
            }
            if (maxdp == 7)
            {
                if (j == 0)
                    dp[i][j] = min(dp[i - 1][2], dp[i - 1][5]) + temans;
                if (j == 1)
                    dp[i][j] = min(dp[i - 1][4], dp[i - 1][3]) + temans;
                if (j == 2)
                    dp[i][j] = min(dp[i - 1][0], dp[i - 1][7]) + temans;
                if (j == 3)
                    dp[i][j] = min(dp[i - 1][1], dp[i - 1][6]) + temans;
                if (j == 4)
                    dp[i][j] = min(dp[i - 1][1], dp[i - 1][6]) + temans;
                if (j == 5)
                    dp[i][j] = min(dp[i - 1][0], dp[i - 1][7]) + temans;
                if (j == 6)
                    dp[i][j] = min(dp[i - 1][4], dp[i - 1][3]) + temans;
                if (j == 7)
                    dp[i][j] = min(dp[i - 1][5], dp[i - 1][2]) + temans;
            }
        }
    }
    int ans = 0x3f3f3f3f;
    for (int i = 0; i <= maxdp;i++)
        ans = min(ans, dp[m][i]);
    cout << ans << endl;
    return 0;
}
```

E

题意：给定一个联通图，无重边无自环，保证以下两个条件只满足一个

1. 存在一条简单路径（路径上每个点只经过一次），这条简单路径覆盖了至少n/2向上取整个点
2. 可以从图中选出至少n/2向上取整个点，将其两两分组，任意两组点(A,B),(C,D)□原图中这四个点的诱导子图，至多有两条边。

如果满足第一个则输出路径上的点，满足第二个则输出分组。

题解：非常神奇的，图的**dfs树**的应用，也是求连通分量的tarjan算法的基础思想，以前基本都是当作黑盒算法用的。