专题3

目录

1000 Sticks

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int cmp(int x, int y)
{
    return x > y;/*改变sort方向*/
}
int a[70];
int vis[70];/*存放是否使用过的状态*/
int n;/*总量*/
int ans;/*各组长度和*/
int dfs(int cnt, int cur, int pos);
/*已选择数；当前组已选择长度和；当前位置*/
int main()
{
    while (scanf("%d", &n), n) {
        int sum = 0;
        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]);
            sum += a[i];
        }
        sort(a, a + n, cmp);
        for (int i = n; i >= 1; i--) {
            if (sum % i == 0 && sum / i >= a[0]) {
                ans = sum / i;/*剪枝*/
                memset(vis, 0, sizeof(vis));
                if (dfs(0, 0, 0)) {
                    printf("%d\n", ans);
                    break;
                }
            }
        }
    }
    return 0;
}
int dfs(int cnt, int cur, int pos)
{
    if (cnt == n)return 1;
    for (int i = pos; i < n; i++) {
        if (vis[i])continue;
        if (cur + a[i] < ans) {
            vis[i] = 1;
            if (dfs(cnt + 1, cur + a[i], i + 1))return 1;/*不要将i+1误写成pos+1*/
            vis[i] = 0;
            if (cur == 0)return 0;/*剪枝：往后棍子更少更凑不起来*/
            while (a[i + 1] == a[i] && i + 1 < n)i++;/*剪枝：试过的不再试*/
        }
        else if (cur + a[i] == ans) {
            vis[i] = 1;
            if (dfs(cnt+1, 0, 0))return 1;
            vis[i] = 0;/*不要漏写了回溯*/
            return 0;
        }
    }
    return 0;
}
```

1001 N皇后问题

AC代码

```
#include<bits/stdc++.h>
using namespace std;
#include<bits/stdc++.h>
using namespace std;
int a[15];/*列*/
int c[30];/*副对角线*/
int d[30];/*主对角线*/
int num[15];
int n;
int cnt;
int text[11];
void dfs(int i);
int main()
{
    for (n = 1; n <= 10; n++) {
        cnt = 0;
        dfs(1);
        text[n] = cnt;/*需要打表储存一下，不然会超时*/
    }
    while (scanf("%d", &n), n) {
        printf("%d\n", text[n]);
    }
    return 0;
}
void dfs(int i)
{
    if (i == n + 1) {
        cnt++;
        return;
    }
    for (int j = 1; j <= n; j++)
    {
        if (a[j] == 0 && c[i + j - 1] == 0 && d[n - i + j] == 0) {
            num[i] = j;
            a[j] = 1; c[i + j - 1] = 1; d[n - i + j] = 1;
            dfs(i + 1);
            a[j] = 0; c[i + j - 1] = 0; d[n - i + j] = 0;
        }
    }
    return;
}
```

1002 Sudoku Killer

未AC

1003 Card Game Cheater

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int con[110][110];
int vis[110];
int link[110];
int n;
int t;
int find(int x)
{
    for (int i = 1; i <= n; i++){
        if (vis[i] == 0 && con[x][i] == 1){
            vis[i] = 1;
            if (link[i] == -1 || find(link[i])){
                link[i] = x;
                return 1;
            }
        }
    }
    return 0;
}
int hungary()
{
    int res = 0;
    memset(link, -1, sizeof(link));
    for (int x = 1; x <= n; x++) {
        memset(vis, 0, sizeof(vis));
        if (find(x))res++;
    }
    return res;
}

int getScore(string str) {
    int ans = 0;
    if (str[0] >= '2' && str[0] <= '9')
        ans += (str[0] - '0') * 10;
    else if (str[0] == 'T')
        ans += 100;
    else if (str[0] == 'J')
        ans += 110;
    else if (str[0] == 'Q')
        ans += 120;
    else if (str[0] == 'K')
        ans += 130;
    else if (str[0] == 'A')
        ans += 140;
    if (str[1] == 'C')
        ans += 1;
    else if (str[1] == 'D')
        ans += 2;
    else if (str[1] == 'S')
        ans += 3;
    else if (str[1] == 'H')
        ans += 4;
    return ans;
}

int main()
{
    scanf("%d", &t);
    while (t--) {
        memset(con, 0, sizeof(con));
        scanf("%d", &n);
        string str;
        int scoreAdam[2000], scoreEve[2000];
        for (int i = 1; i <= n; i++) {
            cin >> str;
            scoreAdam[i] = getScore(str);
        }
        for (int i = 1; i <= n; i++) {
            cin >> str;
            scoreEve[i] = getScore(str);
        }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                if (scoreEve[i] > scoreAdam[j])
                    con[i][j] = 1;
        printf("%d\n", hungary());
    }
    return 0;
}
```

1004 连连看

AC代码

1005 Tempter of the Bone

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int n, m, t;//n行m列时间t
char Map[10][10];
int se[4][2] = { {-1,0},{1,0},{0,-1},{0,1} };//移动方向
int flag;
int dx, dy;/*结束点*/
int sx, sy;/*初始点*/

void dfs(int sx,int sy,int time)
{
    int temp;
    if (sx > n || sy > m || sx < 1 || sy < 1)return;/*注意sx，sy的范围，千万不能写错*/
    if (time == t && dx == sx && dy == sy) flag = 1;
    if (flag)return;
    temp = (t - time) - abs(sx - dx) - abs(sy - dy);
    if (temp < 0 || temp % 2 == 1) return;/*奇偶性剪枝，不然此题超时*/
    for (int i = 0; i < 4; i++)
    {
        if (Map[sx + se[i][0]][sy + se[i][1]] != 'X')
        {
            Map[sx + se[i][0]][sy + se[i][1]] = 'X';
            dfs(sx + se[i][0], sy + se[i][1], time + 1);
            Map[sx + se[i][0]][sy + se[i][1]] = '.';/*回溯*/
        }
    }
    return;
}
int main()
{
    while (scanf("%d%d%d", &n, &m, &t))
    {
        flag = 0;
        int wall = 0;
        if (m == 0 && n == 0 && t == 0)break;
        for(int i=1;i<=n;i++)
            for (int j = 1; j <= m; j++)
            {
                cin >> Map[i][j];
                if (Map[i][j] == 'S') {
                    sx = i; sy = j;
                }
                if (Map[i][j] == 'D') {
                    dx = i; dy = j;
                }
                if (Map[i][j] == 'X')
                    wall++;
            }
        if (m * n - wall <= t)
            printf("NO\n");
        else {
            Map[sx][sy] = 'X';
            dfs(sx,sy,0);
            if (flag)printf("YES\n");
            else printf("NO\n");
        }
    }
    return 0;
}
```

1006 Red and Black

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int w, h;
char Map[22][22];
int vis[22][22];
int dis[4][2] = { -1,0,1,0,0,1,0,-1 };
int cnt;
int st, ed;
void dfs(int x, int y);
int main()
{
    while (scanf("%d%d", &w, &h), w, h) {
        memset(vis, 0, sizeof(vis));
        for (int i = 1; i <= h; i++)
            for (int j = 1; j <= w; j++)
            {
                cin >> Map[i][j];
                if (Map[i][j] == '@') {
                    st = i, ed = j;
                }
            }
        cnt = 0;
        dfs(st, ed);
        printf("%d\n", cnt);
    }
    return 0;
}
void dfs(int x, int y)
{
    cnt++;
    vis[x][y] = 1;
    for (int i = 0; i < 4; i++) {
        int xx = x + dis[i][0];
        int yy = y + dis[i][1];
        if (!vis[xx][yy] && xx >= 1 && xx <= h && yy >= 1 && yy <= w) {
            if (Map[xx][yy] != '#')
                dfs(xx, yy);
        }
    }
}
```

1007 排列2

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int a[4];
int main()
{
    scanf("%d%d%d%d", &a[0], &a[1], &a[2], &a[3]);
    while (1) {
        if (a[0] + a[1] + a[2] + a[3] == 0)break;
        int temp = a[0];
        if (a[0]) {
            for (int i = 0; i < 4; i++)printf("%d", a[i]);
        }
        while (next_permutation(a, a + 4)) {/*对数组中对应地址的数排列*/
            if (a[0]) {
                if (a[0] != temp) {
                    if (temp)printf("\n");
                    temp = a[0];
                }
                else printf(" ");
                for (int i = 0; i < 4; i++)
                    printf("%d", a[i]);
            }
        }
        printf("\n");/*注意题目输出格式*/
        scanf("%d%d%d%d", &a[0], &a[1], &a[2], &a[3]);
        if (a[0] + a[1] + a[2] + a[3])printf("\n");/*注意题目输出格式*/
    }
    return 0;
}
```

1008 Sudoku

未AC

1009 变形课

AC代码

```cpp
#include<bits/stdc++.h>
using namespace std;
struct turn {
    char fr;/*首字母*/
    char to;/*尾字母*/
    int flag;/*是否使用过*/
}s[10000];
char a[10000];
int cnt;/*记录法则数量*/
void bfs();
int main()
{
    while (scanf("%s", a) != EOF) {
        int len = strlen(a);
        cnt = 0;
        s[++cnt].fr = a[0];
        s[cnt].to = a[len - 1];
        s[cnt].flag = 0;
        while (++cnt) {
            scanf("%s", a);
            if (a[0] == '0')break;
            len = strlen(a);
            s[cnt].fr = a[0];
            s[cnt].to = a[len - 1];
            s[cnt].flag = 0;
        }
        bfs();
    }
    return 0;
}
void bfs()
{
    queue<turn>qu;
    turn cur, nex;
    cur.to = 'b';
    cur.flag = 1;
    qu.push(cur);
    while (!qu.empty()) {
        cur = qu.front();
        qu.pop();
        if (cur.to == 'm') {
            printf("Yes.\n");
            return;
        }
        for (int i = 1; i <= cnt - 1; i++) {
            if (s[i].fr == cur.to && s[i].flag == 0) {
                nex = s[i];
                s[i].flag = 1;
                qu.push(nex);
            }
        }
    }
    printf("No.\n");
}
```

1010 Friends

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int t;
int n, m;
int con[30][2];/*朋友对应关系*/
int on[10], off[10];/*每个人线上线下朋友数*/
int ti[10];
int cnt;
/*测试：int test = 0;*/
void dfs(int k);
int check(int s1[], int s2[],int p)/*判断是否朋友数相等*/
{
    for (int i = 1; i <= p; i++)
        if (s1[i] != s2[i])return 0;
    return 1;
}
int main()
{
    scanf("%d", &t);
    while (t--) {
        cnt = 0;
        memset(on, 0, sizeof(on));
        memset(off, 0, sizeof(off));
        memset(ti, 0, sizeof(ti));
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= m; i++) {
            scanf("%d%d", &con[i][0], &con[i][1]);
            ti[con[i][0]]++;
            ti[con[i][1]]++;
        }
        int flag = 1;
        for (int i = 1; i <= n; i++) {
            if (ti[i] % 2 == 1) {
                printf("0\n");
                flag = 0;
                break;
            }
        }
        if (flag) {
            dfs(1);
            printf("%d\n", cnt);
        }
    }
    return 0;
}
void dfs(int k)/*k为已用关系数+1*/
{
    if (k == m + 1) {
        /*测试程序*/
        /*printf("%d\n", ++test);
        for (int i = 1; i <= n; i++) {
            printf("on:%d,off:%d\n", on[i], off[i]);
        }
        printf("%d\n", check(on, off, n));
        printf("\n\n");*/
        if (check(on, off, n))cnt++;
        return;
    }
    /*剪枝*/
    if (on[con[k][0]] < ti[con[k][0]] / 2 && on[con[k][1]] < ti[con[k][1]] / 2) {
        on[con[k][0]]++;/*对应朋友数加1*/
        on[con[k][1]]++;
        dfs(k + 1);
        on[con[k][0]]--;/*如果on不行，退回上一步选off*/
        on[con[k][1]]--;
    }
    if (off[con[k][0]] < ti[con[k][0]] / 2 && off[con[k][1]] < ti[con[k][1]] / 2) {
        off[con[k][0]]++;
        off[con[k][1]]++;
        dfs(k + 1);
        off[con[k][0]]--;
        off[con[k][1]]--;
    }
    return;
}
```

1011 How many ways

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int Map[105][105];
int Sum[105][105];
int t, m, n;
int si, sj;
int dfs(int si, int sj);
int main()
{
    scanf("%d", &t);
    while (t--)
    {
        memset(Sum,0,sizeof(Sum));
        scanf("%d%d", &n, &m);
        for(int i=1;i<=n;i++)
            for (int j=1; j <= m; j++)
                scanf("%d", &Map[i][j]);
        si = 1; sj = 1;
        printf("%d\n", dfs(1, 1)%10000);
    }
    return 0;
}
int dfs(int si, int sj)
{
    if (Sum[si][sj])return Sum[si][sj];
    int step = Map[si][sj];
    if (si == n && sj == m) {
        Sum[si][sj]=1; return 1;
    }
    int i, j;
    for (i = 0;i <= step; i++)
        for (j = 0; i + j <= step; j++)
        {
            if (i + j == 0)continue;
            if (si+i<=n&&sj+j<=m) {
                Sum[si][sj]=(Sum[si][sj]+dfs(si + i, sj + j))%10000;/*将该点后续情况数存放在数组里*/
            }
        }
    return Sum[si][sj];
}
```

1012 FatMouse and Cheese

AC代码

```
#include<bits/stdc++.h>
using namespace std;
int n, k;
int init[102][102];/*初始值*/
int ans[102][102];/*dfs储存值*/
int dis[4][2] = { 1,0,-1,0,0,1,0,-1 };/*方向*/
int OK(int x, int y)
{
    if (x < 0 || x >= n || y < 0 || y >= n)return 0;/*判断是否在区域边界内*/
    else return 1;
}
int dfs(int x, int y);
int main()
{
    while (scanf("%d%d", &n, &k), n != -1, k != -1) {
        memset(ans, 0, sizeof(ans));/*注意对ans数组的清零*/
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                scanf("%d", &init[i][j]);
        printf("%d\n", dfs(0,0));
    }
    return 0;
}
int dfs(int x, int y)
{
    int answer = 0;
    if (ans[x][y])return ans[x][y];/*代表该位置在其它路线上已经计算过*/
    for(int i=0;i<4;i++)
        for (int j = 1; j <= k; j++) {
            int xx = x + dis[i][0] * j;
            int yy = y + dis[i][1] * j;
            if (OK(xx, yy) && init[xx][yy] > init[x][y])
                answer = max(answer, dfs(xx,yy));/*由于题目要求求得最大值，保存最大的情况*/
        }
    ans[x][y] = answer + init[x][y];/*记忆化dfs*/
    return ans[x][y];
}
```