计蒜客 2018 蓝桥杯模拟赛
第一场题解

by islands

A组第一题 and B组第一题

- 结果填空：年龄

- 讲解：简单题，直接暴力枚举每一个人可能的年龄。然后查看是否符合条件。答案其实正好只有一组，两人年龄分别为 18 和 27

。

```cpp
#include <iostream>
using namespace std;

int main() {
    int ans = 0;
    for (int i = 10; i < 100; ++i) {
        for(int j = 10; j < 100; ++j) {
            int a = i / 10;
            int b = i % 10;
            int c = j / 10;
            int d = j % 10;
            if (((a + b) * 3 == c * 10 + d) && ((c + d) * 2 == 10 * a +
                ans++;
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```

B 组第二题

- 结果填空：开关灯
- 讲解：简单题，用数组模拟一遍。

答案是 571。

```cpp
#include <iostream>
using namespace std;
bool f[1010];
int main() {
    int ans = 0;
    for (int i = 3; i <= 1000; i += 3) {
        f[i] = !f[i];
    }
    for (int i = 5; i <= 1000; i += 5) {
        f[i] = !f[i];
    }
    for (int i = 7; i <= 1000; i += 7) {
        f[i] = !f[i];
    }
    for (int i = 1; i <= 1000; ++i) {
        if (!f[i]) {
            ++ans;
        }
    }
    cout << ans << endl;
    return 0;
}
```

A组第二题目 and B组第三题

- 结果填空：U型数字
- 讲解：暴力 + 简单模拟，从 1 到 100000 枚举每一个数字，然后查看每一个数字，是否符合条件。从前往后找到一个不递减的位置，然后检查这个位置是否递增到最后。
- 答案为 8193

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    int ans = 0;
    for (int i = 1; i <= 100000; i++) {
        string s = to_string(i);
        bool down = 0, up = 0;
        int j;
        for (j = 1; j < s.size(); j++) {
            if ( s[j - 1] > s[j] ) {
                down = 1;
            } else {
                break;
            }
        }
        for (; j < s.size(); j++) {
            if ( s[j - 1] < s[j] ) {
                if (j == s.size()-1)
                    up = 1;
            } else {
                break;
            }
        }
        if (down && up) {
            ans++;
        }
    }
    cout << ans << endl;
    return 0;
}
```

A组第三题

- 结果填空：加减乘

- 讲解：dfs，把每个空格使用加减乘除进行填写，但是要处理好优先级，我这里使用的是每次停留一下看下一个符号是什么。如果是加号或者是减号就把上一个计算给计算了，如果是乘号就把 b 和当前数相乘。继续保留上一个符号继续往后看。

- 答案为 63。

```c
int a_b(int a, char c, int b) {
    if (c == '+') {
        return a+b;
    }
    return a - b;
}
int dfs(int a, char c, int b, int cnt) {
    if (cnt == 11) {
        return a_b(a, c, b) == 0;
    }
    int res = 0;
    res += dfs(a_b(a,c,b), '+', f[cnt], cnt + 1);
    res += dfs(a_b(a, c, b), '-', f[cnt], cnt + 1);
    res += dfs(a, c, b * f[cnt], cnt + 1);
    return res;
}
```

A组第四题 and B组第四题

- 代码填空：LIS

讲解：经典算法，这里填 f[i] = max(f[i], f[j]+1);

A组第五题 and B组第五题

- 代码填空：全排列

- 讲解：全排列用 dfs 实现的，这里需要填空的代码是 str[i] == str[j] && vis[j]。
- 涉及到有重复元素的全排列，后面相同的元素只需要和一个交换就可以了，我们这样写，相同的元素，只和最后一个交换。

B组第六题

- 结果填空：数独

- 讲解：数独用 dfs 暴搜可以解决，标记每行，每列，和每个小方格用过的元素。当然对于更快的解法，可以用 DLX 来求解。这个数独的难度比较大，有很多解，手算有点难。给出一组解。

```
1 2 6 7 3 4 5 9 8
3 7 8 5 9 2 6 1 4
4 9 5 1 6 8 2 3 7
7 3 9 4 2 5 1 8 6
8 6 1 3 7 9 4 2 5
2 5 4 8 1 6 3 7 9
5 4 7 2 8 1 9 6 3
6 1 3 9 5 7 8 4 2
9 8 2 6 4 3 7 5 1
```

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 107;
const int inf = 0x3f3f3f3f;
struct node {
    int x, y;
}g[90];
int f[11][11];
int flag, num;
bool check(int a, int b, int n){
    for (int i=0; i<9; ++i) {
        if (f[a][i] == n || f[i][b] == n) {
            return false;
        }
    }
    for (int i = a / 3 * 3; i < a / 3 * 3 + 3; ++i) {
        for (int j = b / 3 * 3; j < b / 3 * 3 + 3; ++j) {
            if (f[i][j] == n) {
                return false ;
            }
        }
    }
    return true;
}
void dfs(int cnt) {
    if (flag) {
        return;
    }
    if (cnt == num) {
        flag = 1;
        return;
    }
    for (int k = 1; k <= 9; ++k) {
        if ( check(g[cnt].x, g[cnt].y, k)) {
            f[g[cnt].x][g[cnt].y]=k;
```

```cpp
    }
    for (int k = 1; k <= 9; ++k) {
        if ( check(g[cnt].x, g[cnt].y, k)) {
            f[g[cnt].x][g[cnt].y]=k;
            dfs(cnt + 1);
            if(flag == 1) return;
            f[g[cnt].x][g[cnt].y] = 0;
        }
    }
}
int main() {
    int i, j, k, n, m, t, tt = 0, x;
    char ch;
    while (cin >> ch) {
        memset(f, 0, sizeof f);
        flag = 0,num = 0;
        if (ch != '*') {
            f[0][0] = ch - '0';
        }
        else {
            g[num].x = 0;
            g[num++].y = 0;
        }
        for (i=0; i < 9; ++i) {
            for (j = 0; j < 9; ++j) {
                if (i == 0 && j == 0) {
                    continue;
                }
                cin >> ch;
                if(ch != '*') {
                    f[i][j]=ch-'0';
                } else {
                    g[num].x = i;
                    g[num++].y = j;
                }
```

```cpp
            }
        }
    }
    dfs(0);
    if (tt++) {
        cout << endl;
    }
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 8; j++) {
            cout << f[i][j] << " ";
        }
        cout << f[i][8] << endl;
    }
  }
  return 0;
}
```

A组第六题

- 结果填空：铺瓷砖
- 讲解：方法 1 ，状态压缩动态规划，每一行的状态压缩成为一个二进制数，然后进行转移。对于 n 很大的时候，还可以用矩阵优化。
- 方法2，轮廓线动态规划。
- 答案为 258584046368。

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1 << 12;
long long dp[2][maxn];
int cur, n, m;

void add(int a, int b) {
    if (b & (1 << m)) {
        dp[cur][b ^ (1 << m)] += dp[cur ^ 1][a];
    }
}

int main() {
    while (scanf("%d%d", &n, &m) != EOF) {
        if (n < m) {
            swap(n, m);
        }
        int all = (1 << m) - 1;
        memset(dp, 0, sizeof dp);
        dp[0][all] = 1;
        cur = 0;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < m; ++j) {
                cur ^= 1;
                memset(dp[cur], 0, sizeof dp[cur]);
                for (int sta = 0; sta <= all; ++sta) {
                    if (dp[cur ^ 1][sta] == 0) {
                        continue;
                    }
                    add(sta, sta << 1);  //  这个点不放,sta<<1让sta的最后
                    if (i && !(sta & (1 << (m-1)))) {  //  竖着放,不是第-
                        add(sta, (sta << 1) ^ (1 << m) ^ 1);
                    }
                    if (j && !(sta & 1)) {
```

```cpp
                    }
                    add(sta, sta << 1);  //  这个点不放,sta<<1让sta的最后
                    if (i && !(sta & (1 << (m-1)))) {  //  竖着放,不是第-
                        add(sta, (sta << 1) ^ (1 << m) ^ 1);
                    }
                    if (j && !(sta & 1)) {
                        add(sta, (sta << 1) ^ 3);
                    }
                }
            }
        }
        printf("%lld\n", dp[cur][all]);
    }
    return 0;
}
```

A组第七题 and B组第七题

- 数列求值

我们假设 A_1 的值为 x。

把公式

$$Ai = \frac{Ai-1+A_{i+1}}{2} - C_i$$

变形一下得到

$$A_{i+1} = 2(A_i + Ci) - Ai - 1$$

我们可以得到

$$A_2 = 2(A_1 + C_1) - A_0 = 2A_1 + x_2$$

$$A_3 = 2(A_2 + C_2) - A_1 = 3A_1 + x_3$$

$$A_4 = 2(A_3 + C_3) - A_2 = 4A_1 + x_4$$

$$\cdots$$

$$A_{n+1} = 2(A_n + Cn) - An - 1 = (n+1)A1 + xn + 1$$

所以我们只需要求出 x_{n+1} 就可以算出来 A_1 了。

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
double x[1010];
double C[1010];
int main() {
    int n;
    cin >> n;
    double sum = 0, A0, An1;
    cin >> A0 >> An1;
    x[0] = A0;
    x[1] = 0;
    for (int i = 1; i <= n; ++i) {
        cin >> C[i];
        x[i + 1] = 2.0 * x[i] - x[i - 1] + 2.0 * C[i];
    }
    double ans = (An1 - x[n + 1]) / (n + 1);
    printf("%.2lf\n", ans);
    return 0;
}
```

B组第八题

- 封印之门

- 讲解：首先我们发现每一位上的字符是不会相互影响的，实际上
- 本质就是计算把一个字符 a 变成 b 需要的最少步数。这里我
- 们可以借助图论中的 floyd 算法，预处理计算出任意两点之间的最短路，
- 最要对于每一位可以直接取出答案。

- 当然对于本题的数据范围，还可以对于每一位进行 dfs 搜索也是可以的。

- 标程是用 floyd 来实现的。

```cpp
#include <iostream>
#include <string>
using namespace std;
const int inf = 0x3fffffff;
int G[30][30];
int main() {
    for (int i = 0; i < 26; ++i) {
        for (int j = 0; j < 26; ++j) {
            if (i == j) {
                G[i][j] = 0;
            } else {
                G[i][j] = inf;
            }
        }
    }
    string s1, s2;
    cin >> s1 >> s2;
    int k;
    cin >> k;
    while (k--) {
        char a, b;
        cin >> a >> b;
        if (a != b) {
            G[a - 'a'][b - 'a'] = 1;
        }
    }
    for (int k = 0; k < 26; ++k) {
        for (int i = 0; i < 26; ++i) {
            for (int j = 0; j < 26; ++j) {
                G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
            }
        }
    }
    int sum = 0;
    for (int i = 0; i < s1.size(); ++i) {
        if (G[s1[i] - 'a'][s2[i] - 'a'] >= inf) {
```

```cpp
    int sum = 0;
    for (int i = 0; i < s1.size(); ++i) {
        if (G[s1[i] - 'a'][s2[i] - 'a'] >= inf) {
            sum = -1;
            break;
        } else {
            sum += G[s1[i] - 'a'][s2[i] - 'a'];
        }
    }
    cout << sum << endl;
    return 0;
}
```

A组第八题 and B组第九题

- 天上的星星

- 讲解：

用一个二维数组 S_{ij} 记录矩形 $(0,0)$ - (i,j) 的总和。

预处理的时候有 $S_{ij} = w_{ij} + S_{(i-1)j} + S_{i(j-1)} - S_{(i-1)(j-1)}$。

对于查询 x_1, y_1, x_2, y_2，答案就是 $S_{x_2 y_2} - S_{x_2 (y_2 - 1)} - S_{(x_1 -) y_2} + S_{x_1 y_1}$。

由于坐标中可能出现 0 的情况，我们把所以坐标都加上 1，这样就巧妙避免了数组访问越界了。

```cpp
#include <iostream>
using namespace std;
const int mmax = 2010;
long long sum[mmax][mmax];
int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        int x, y, w;
        ++x, ++y;
        cin >> x >> y >> w;
        sum[x][y] += w;
    }
    for (int i = 1; i < mmax; ++i) {
        for (int j = 1; j < mmax; ++j) {
            sum[i][j] +=  sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j -
        }
    }
    int q;
    cin >> q;
    while (q--) {
        int a, b, c, d;
        cin >> a >> b >> c >> d;
        ++a, ++b, ++c, ++d;
        cout << sum[c][d] - sum[a - 1][d] - sum[c][b - 1] + sum[a - 1][b
    }
    return 0;
}
```

A组第九题 and B组第十题

- 青出于蓝胜于蓝
- 讲解：我们对树做一次 dfs，按照 dfs 的顺序给每个点重新编号，那么，每个点
- 对应所有子结点新编号实际上正好是一段连续区间，这就是欧拉序，每
- 个点的新编号都是一个时间戳。
- 通过 dfs 处理出来每个点的子节点的编号的区间以后，用 l[i] 和 r[i] 表示每个点的子结点区间左右端点。然后我们可以借助线段树或者树状数组来维护区间问题。
- 按照武功排名从小到大处理，对于第 i 个人，l[i] 到 r[i] 之间的人数就是对应的答案，因为这时候比 i 大的点还没有处理，所以里面不会包含比 i 大的人。然后更新第 i 个点。

```cpp
#include <iostream>
#include <cstring>
using namespace std;
const int mmax = 100010;
struct node {
    int en, next;
} E[2 * mmax];
int p[mmax];
int num;
void init() {
    memset(p, -1, sizeof p);
    num = 0;
}
void add(int st, int en) {
    E[num].en = en;
    E[num].next = p[st];
    p[st] = num++;
}
int l[mmax], r[mmax];
int times;
void dfs(int u, int fa) {
    l[u] = ++times;
    for (int i = p[u]; i + 1; i = E[i].next) {
        int v = E[i].en;
        if (v != fa) {
            dfs(v, u);
        }
    }
    r[u] = times;
}
int n, f;
int C[mmax];
```


```cpp
int n, f;
int C[mmax];
```


```cpp
void up(int x, int v) {
    for (int i = x; i <= n; i += (i & -i)) {
        C[i] += v;
    }
}
int sum(int x) {
    int res = 0;
    for (int i = x; i > 0; i -= (i & -i)) {
        res += C[i];
    }
    return res;
}

int main() {
    init();
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    times = 0;
    dfs(f, - 1);
    for (int i = 1; i <= n; ++i) {
        cout << sum(r[i]) - sum(l[i]) << " \n"[i == n];
        up(l[i], 1);
    }
    return 0;
}
```

A组第十题

- 蒜头君王国

- 讲解：

用 $F(n)$ 表示 n 个点联通的概率，$G(n)$ 表示 n 个点不联通的概率。

所以 $F(n) = 1 - G(n)$。

考虑第 n 个点所在的联通块的个数为 k，那么所有的和 n 联通的点的组合为 C_{n-1}^{k-1}，然后剩下的 $n - k$ 个点不能和这 k 个点连边，概率为 $F(k)(1-p)^{k(n-k)}$。

所以转移如下

$$G(n) = \sum_{k=1}^{n-1} C_{n-1}^{k-1} F(k)(1-p)^{k(n-k)}$$

```cpp
#include<iostream>
#include<algorithm>
#include<string.h>
#include<stdio.h>
using namespace std;
double ans[30];
long long c(int n,int m) {
    long long su = 1ll;
    for (int i = m + 1; i <= n; ++i) {
        su *= i;
    }
    for (int i = 1; i <= n - m; ++i) {
        su /= i;
    }
    return su;
}
double Pow(double x, int n) {
    double su = 1.0;
    while (n) {
        if (n & 1) {
            su = su * x;
        }
        x = x * x;
        n >>= 1;
    }
    return su;
}
int main() {
    int n;
    double p;
    scanf("%d %lf", &n, &p);
    ans[1] = 1.0;
    ans[2] = p;
    for (int i = 3; i <= 20; ++i) {
```

```cpp
        x = x * x;
        n >>= 1;
    }
    return su;
}
int main() {
    int n;
    double p;
    scanf("%d %lf", &n, &p);
    ans[1] = 1.0;
    ans[2] = p;
    for (int i = 3; i <= 20; ++i) {
        ans[i] = 1.0;
    }
    for (int i = 3; i <= n; ++i) {
        for (int k = 1; k <= i - 1; ++k) {
            ans[i] -= c(i - 1, k - 1) * ans[k] * Pow(1.0 - p, k * (i - k
        }
    }
    printf("%.7lf\n",ans[n]);
    return 0;
}
```