ACM程序设计竞赛算法讲座

第一讲

沈　　俊

852679238@QQ.COM

2020.9.5

上海大学ACM集训队

一、模拟

学习目标

1. 熟练应用模拟法解决一些实际问题。

2. 体验模拟法的审题分析和细节测试。

模拟

在程序设计竞赛中，有一类问题是模拟一个游戏的对弈过程，或者模拟一项任务的操作过程，进行统计计分、判断输赢等。这些问题很难建立数学模型用特定算法解决，一般只能采用“模拟”法。用模拟法解题必须关注以下几个问题：审题要仔细，游戏规则不能错；分析要全面，各种特例不能漏；编程要细心，测试运行要到位。

例1、蚱蜢

【问题描述】

有一天，一只蚱蜢像往常一样在草地上愉快地跳跃，它发现了一条写满了英文字母的纸带。

蚱蜢只能在字母(A、E、I、O、U、Y)间跳跃，一次跳跃所需的能力是两个位置的差。纸带所需的能力值为蚱蜢从纸带开头的前一个位置根据规则跳到纸带结尾的后一个位置的过程中能力的最大值。

蚱蜢想知道跳跃纸带所需的能力值(最小)是多少。如图 9.3-1 所示的纸带所需的能力值(最小)是 4。

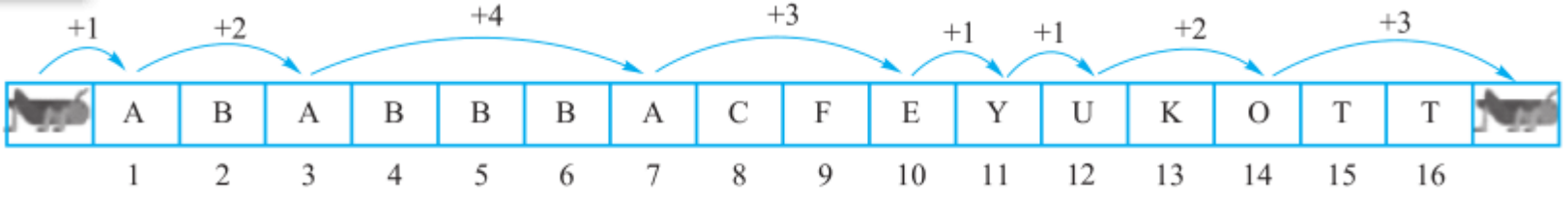

图 9.3-1　蚱蜢跳跃示意图

例1、蚱蜢

【输入格式】

一行一个字符串，字符串长不超过 100。

【输出格式】

一行一个整数，代表(最小)能力值。

【输入样例】

KMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKL

PMNFVCKMLPTGFHNBVCDRFGHNMBVXWSQFDCV

BNHTJKLPMNFVC

【输出样例】

85

例1、蚱蜢

【问题分析】

从头到尾枚举纸带的每一个字母，按照规则模拟蚱蜢在元音字母之间跳跃的过程，打擂台记录能力值。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int pre=-1,ans = 0;
    string s;
    cin >> s;
    for(int i = 0; i < s.length(); i++){
        if (s[i]=='A'||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='Y'){
            if (i-pre > ans)   ans = i-pre;
            pre = i;
        }
    }
    if (s.length()-pre > ans)     ans = s.length()-pre;
    cout << ans << endl;
    return 0;
}
```

例2、乒乓球

【问题描述】

国际乒联立志推行一系列改革，以推动乒乓球运动在全球的普及。其中 11 分制改革引起了很大的争议，有一部分球员因为无法适应新规则只能选择退役。华华就是其中一位，他退役之后走上了乒乓球研究工作，意图弄明白 11 分制和 21 分制对选手的不同影响。在开展研究之前，他首先需要对自己多年比赛的统计数据进行一些分析，所以需要一些帮忙。

华华通过以下方式进行分析，首先将比赛每个球的胜负列成一张表，然后分别计算在 11 分制和 21 分制下，双方的比赛结果(截至记录末尾)。

例2、乒乓球

比如，现在有这么一份记录，(其中 W 表示华华获得一分，L 表示华华对手获得一分)：

WWWWWWWWWWWWWWWWWWWWWWLW。在 11 分制下，此时比赛的结果是华华第一局 11 比 0 获胜，第二局 11 比 0 获胜，正在进行第三局，当前比分 1 比 1。而在 21 分制下，此时比赛结果是华华第一局 21 比 0 获胜，正在进行第二局，比分 2 比 1。如果一局比赛刚开始，则此时比分为 0 比 0。

本题就是要对于一系列比赛信息的输入(WL 形式)，输出正确的结果。

【输入格式】

输入文件包含若干行字符串(每行至多 20 个字母)，字符串由大写的 W、L 和 E 组成。其中E 表示比赛信息结束，程序应该忽略 E 之后的所有内容。

例2、乒乓球

【输出格式】

输出由两部分组成，每部分有若干行，每一行对应一局比赛的比分(按比赛信息输入顺序)。

其中第一部分是 11 分制下的结果，第二部分是 21 分制下的结果，两部分之间由一个空行分隔。

【输入样例】

WWWWWWWWWWWWWWWWWWWWWWLWE

【输出样例】

11：0
11：0
1：1

21：0
2：1

```
#include<bits/stdc++.h>
using namespace std;
char ch,s[62510];
int a,b,n;
int main(){
    while(cin>>ch)
        if(ch=='E')break;
        else s[++n]=ch;
    for(int i=1;i<=n;i++){
        if(s[i]=='W')    a++;
        else if(s[i]=='L')  b++;
        if((a>=11 || b>=11) && abs(a-b) > 1){
            cout<<a<<":"<<b<<endl;
            a=0;        b=0;
        }
      }
```

```
        cout<<a<<":"<<b<<endl<<endl;
        a=0;        b=0;
        for(int i=1;i<=n;i++){
            if(s[i]=='W')a++;
            else if(s[i]=='L')b++;
            if((a >= 21 || b >= 21) && abs(a-b) > 1){
                cout<<a<<":"<<b<<endl;
                a=0;        b=0;
            }
        }
        cout<<a<<":"<<b<<endl;
        return 0;
}
```

二、递推

学习目标

1. 理解递推关系和递推法。

2. 体会递推法解题的三个重点和两类模型。

3. 熟练应用递推法解决一些实际问题。

递推

“递推”是计算机解题的一种常用方法。利用“递推法”解题首先要分析归纳出“递推关系”。如经典的斐波那契数列问题，用 f (i)表示第 i 项的值，则 f (1) =0，f(2) =1，在 n>2 时，存在递推关系：f (n) = f(n-1) + f(n-2)。

在递推问题模型中，每个数据项都与它前面的若干个数据项（或后面的若干个数据项）存在一定的关联，这种关联一般是通过一个“递推关系式”来描述的。求解问题时，需要从初始的一个或若干数据项出发，通过递推关系式逐步推进，从而推导计算出最终结果。这种求解问题的方法叫“递推法”。其中，初始的若干数据项称为“递推边界”。

递推

解决递推问题有三个重点：

1、建立正确的递推关系式；

2、分析递推关系式的性质，确定边界值；

3、根据递推关系式编程求解。

递推法分为“顺推”和“倒推”两类模型。

所谓顺推，就是从问题的边界条件(初始状态)出发，通过递推关系式依次从前往后递推出问题的解。

所谓倒推，就是在不知道问题的边界条件(初始状态)下，从问题的最终解(目标状态或某个中间状态)出发，反过来推导问题的初始状态。

例3、铺瓷砖

【问题描述】

用红色的 1×1 和黑色的 2×2 两种规格的瓷砖不重叠地铺满 n×3 的路面，求出有多少种不同的铺设方案。

【输入格式】

一行一个整数 n，0<n<1000。

【输出格式】

一行一个整数，为铺设方案的数量模12345的结果。

【输入样例】

2

【输出样例】

3

例3、铺瓷砖

【问题分析】

用 f(n) 表示 n×3 的路面有多少种不同的铺设方案。把路面看成 n 行 3 列，则问题可以分成两种情况考虑，一种是最后一行用 3 块 1×1 的瓷砖铺设；另一种是最后两行用 1 块 2×2 和 2 块 1×1 的瓷砖铺设(最后两行就有两种铺法)，第一种铺法就转换为 f(i-1) 的问题了，第二种铺法就转换成 f(i-2) 的问题了。根据加法原理，得到的递推关系式为 f(i) = f(i-1) +f(i-2)×2，边界为 f(0)=1，f(1)=1。

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,a[1005];
    cin >> n;
    a[0] = 1; a[1] = 1;
    for(int i = 2; i <= n; i++)
            a[i] = (a[i-1] + a[i-2]*2) % 12345;
    cout << a[n] << endl;
    return 0;
}
```

例4、彩带

【问题描述】

一中 90 周年校庆，小林准备用一些白色、蓝色和红色的彩带来装饰学校超市的橱窗，他希望满足以下两个条件：

(1) 相同颜色的彩带不能放在相邻的位置；

(2) 一条蓝色的彩带必须放在一条白色的彩带和一条红色的彩带中间。

现在，他想知道满足要求的放置彩带的方案数有多少种。例如，如图 9.4-1 所示为橱窗宽度n=3 的所有放置方案，共 4 种。

图 9.4-1　彩带放置示意图

例4、彩带

【输入格式】
一行一个整数 n，表示橱窗宽度(或者说彩带数目)。
【输出格式】
一行一个整数，表示装饰橱窗的彩带放置方案数。
【样例输入】
3
【样例输出】
4
【数据规模】
对 30% 的数据满足：1≤n≤15。
对 100% 的数据满足：1≤n≤45。

例4、彩带

【问题分析】

用 f (i) 表示宽度为 i 的橱窗(或 i 条彩带)的合法放置方案数，则 f(1) =2，f(2) =2，f(3) =4，f(4) =6，f(5) =10，……不难发现，答案就是初始值不一样的斐波那契数列，所以，用递推法就可以很方便地求出 f (n)。

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,a[100];
    cin >> n;
    a[1] = 2; a[2] = 2;
    for(int i = 3; i <= n; i++)
            a[i] = a[i-1] + a[i-2];
    cout << a[n] << endl;
    return 0;
}
```

例5、城市路径

【问题描述】

地图上有 n 个城市，一只奶牛要从 1 号城市开始依次经过这些城市，最终到达 n 号城市。但是这只奶牛觉得这样太无聊了，所以它决定跳过其中的一个城市(但是不能跳过 1 号和 n 号城市)，使得它从 1 号城市开始，到达 n 号城市所经过的总距离最小。假设每一个城市 i 都有一个坐标(x_i，y_i)，从 (x_1，y_1) 的城市 1 到 (x_2，y_2) 的城市 2 之间的距离为 $|x_1-x_2|+|y_1-y_2|$。

【输入格式】

第 1 行 1 个正整数 n，表示城市个数。

接下来的 n 行，每行 2 个数 x_i 和 y_i，表示城市 i 的坐标。

【输出格式】

一行一个数，使得它从1号城市开始，跳过某一个城市，到达n号城市所经过的最小总距离。

【输入样例】

4

0 0

8 3

11 -1

10 0

【输出样例】

14

【样例说明】

跳过 2 号城市。

【数据规模】

对于 40% 的数据满足：n≤1000。

对于 100% 的数据满足：3≤n≤100000，-1000≤x i ， y i ≤1000。

例5、城市路径

【问题分析】

设f (i)为从城市1依次跳到城市i的距离之和，设g(i)为从城市i依次跳到城市n的距离之和，则问题的答案为 min{f (i-1)+g(i+1)+dis[i-1,i+1]}。其中，dis [i-1,i+1]表示城市 i-1 到城市 i+1的曼哈顿距离，f (i)和 g(i)都可以用递推来预处理求出：

f(i)= f(i-1)+ dis[i-1,i]，g(i) =g(i+1) +dis[I,i+1]。

也可以设 f (i)表示从城市 1 依次跳到城市 n，且跳过城市 i 的距离之和，sum 表示表示从城市 1 依次跳到城市 n 的距离之和，则 f (i)=min{sum–dis[I,i-1]-dis[i+1.i]+dis[i-1,i+1]}。

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100001;
pair<int,int> coor[MAXN];
int dist(int a, int b){
        return abs(coor[a].first-coor[b].first) + abs(coor[a].second-coor[b].second);
}
int main(){
    int n,sum=0,ans;
    cin >> n;
    for(int i = 1; i <= n; i++)          cin >> coor[i].first >> coor[i].second;
    for(int i = 2; i <= n; i++)          sum += dist(i,i-1);
    ans = sum;
    for(int i = 2; i < n; i++)
        ans = min(ans,sum-dist(i,i-1)-dist(i+1,i) + dist(i-1,i+1));
    cout << ans << endl;
    return 0;
}
```

练习1、保龄球

【问题描述】

打保龄球是用一个滚球去打击十个站立的柱，将柱击倒。一局分十轮，每轮可滚球一次或多次，以击倒的柱数为依据计分。一局得分为十轮得分之和，而每轮的得分不仅与本轮滚球情况有关，还可能与后续一两轮的滚球情况有关。即某轮某次滚球击倒的柱数不仅要计入本轮得分，还可能会计入前一两轮得分。具体的滚球击柱规则和计分方法如下：

(1) 若某一轮的第一次滚球就击倒全部十个柱，则本轮不再滚球(若是第十轮则还需另加两次滚球，不妨称其为第十一轮和第十二轮，并不是所有的情况都需要滚第十一轮和第十二轮球)。该轮得分为本次击倒柱数 10 与以后两次滚球所击倒柱数之和。

练习1、保龄球

(2) 若某一轮的第一次滚球未击倒十个柱，则可对剩下未倒的柱再滚球一次。如果这两次滚球击倒全部十个柱，则本轮不再滚球(若是第十轮则还需另加一次滚球)，该轮得分为这两次共击倒柱数 10 与以后一次滚球所击倒柱数之和。

(3) 若某一轮两次滚球未击倒全部十个柱，则本轮不再继续滚球，该轮得分为这两次滚球击倒的柱数之和。

总之，若某一轮中一次滚球或两次滚球击倒十个柱，则本轮得分是本轮首次滚球开始的连续三次滚球击倒柱数之和(其中有一次或两次不是本轮滚球)。若一轮内二次滚球击倒柱数不足十个，则本轮得分即为这两次击倒柱数之和。下面以实例说明如下：

练习1、保龄球

轮	1	2	3	4	5	6	7	8	9	10	11	12
击球情况	/	/	/	72	9/	81	8/	/	9/	/	8/	
各轮得分	30	27	19	9	18	9	20	20	20	20		
累计总分	30	57	76	85	103	112	132	152	172	192		

现在请编写一个保龄球计分程序，用来计算并输出最后的总得分。

【输入格式】

输入一行，为前若干轮滚球的情况，每轮滚球用一到两个字符表示，每一个字符表示一次击球，字符“/”表示击倒当前球道上的全部的柱，否则用一个数字字符表示本次滚球击倒的当前球道上的柱的数目，两轮滚球之间用一个空格隔开。

练习1、保龄球

【输出格式】
输出一行一个整数，代表最后的得分。
【输入样例 1】
/ / / 72 9/ 81 8/ / 9/ / 8/
【输出样例 1】
192
【输入样例 2】
90 90 / 9/ 81 / / / 72 / /0
【输出样例 2】
170
【输入样例 3】
/ / / 72 9/ 81 8/ / 9/ 15
【输出样例 3】
169

练习2、穿越沙漠

【问题描述】

一辆卡车欲穿过 1000 千米的沙漠，卡车耗油为 1 升 / 千米，卡车总载油能力为 500 升。显然，卡车装一次油是过不了沙漠的。因此，司机必须设法在沿途建立几个贮油点，使卡车能顺利穿越沙漠。试问：司机如何建立这些贮油点？每一贮油点应存多少油，才能使卡车以消耗最少

汽油的代价通过沙漠？结果保留小数点后两位。

编程计算及打印建立的贮油点序号，各贮油点距沙漠边沿出发的距离以及存油量，格式如下：

No.	Distance(km)	Oil (litre)
1	×××.××	×××.××
2	×××.××	×××.××
3	×××.××	×××.××