具体数学初级班第六周参考答案

517 老师

2020 年 6 月 4 日

0.1 A 题

详见例题一

0.2 B 题

详见例题二

0.3 C 题

注意这个是 poj 的题，不能用万能头文件

设 $dp[i][j][k]$ 表示第 i 个队伍在前 j 道题种解出 k 道题的概率

转移方程为 $dp[i][j][k] = dp[i][j-1][k-1] * p[i][j] + dp[i][j-1][k] * (1 - p[i][j])$

$s[i][k]$ 表示第 i 队做出的题小于等于 k 的概率

利用 dp 数组可以求出 s 数组

每个队至少做出一个题的概率为 $p_1 = (1 - s[1][0]) * (1 - s[2][0]) * ... * (1 - s[T][0])$

每个队做出的题数都在 1 到 $N-1$ 的概率为 $p_2 = (s[1][N-1] - s[1][0]) * (s[2][N-1] - s[2][0]) * ... * (s[T][N-1] - s[T][0])$

最终答案就是 $p_1 - p_2$

C 题

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
double dp[1005][40][40];
```

```
double p[1005][40];
double s[1005][40];
int main()
{
  int m,t,n;
  double ans,sum;
  while(scanf("%d%d%d",&m,&t,&n)==3) {
    memset(dp,0,sizeof(dp));
    memset(s,0,sizeof(s));
    ans=1; sum=1;
    if(m==0&&t==0&&n==0) break;
    for(int i=1;i<=t;i++) {
      for(int j=1;j<=m;j++) {
        scanf("%lf",&p[i][j]) ;
      }
    }
    for(int i=1;i<=t;i++) {
      dp[i][1][0]=1-p[i][1];
      dp[i][1][1]=p[i][1];
      for(int j=2;j<=m;j++)
        dp[i][j][0]=dp[i][j-1][0]*(1-p[i][j]);
      for(int j=2;j<=m;j++) {
        for(int k=1;k<=j;k++) {
          dp[i][j][k]=dp[i][j-1][k]*(1-p[i][j])+dp[i][j-1][k-1]*p[i][j];
        }
      }
      s[i][0] = dp[i][m][0];
      for(int k=1;k<=n-1;k++)
        s[i][k]=s[i][k-1]+dp[i][m][k];
    }
    for(int i=1;i<=t;i++) {
      ans *= (1-s[i][0]);
      sum*=s[i][n-1] - s[i][0];
    }
    printf("%.3lf\n",ans-sum);
```

```
  }
  return 0;
}
```

0.4 D 题

设 $dp[i]$ 为分数为 i 的时候到达目标的期望步数

设 $p[k]$ 表示投出 k 分的概率，$p[0]$ 为投出 a, b, c 的概率

p 的求法如下

p 的求法

```
for (int i = 1; i <= k1; i++) {
  for (int j = 1; j <= k2; j++) {
    for (int k = 1; k <= k3; k++) {
      p[i + j + k] += 1.0/(k1*k2*k3);
    }
  }
}
```

dp 转移方程

$dp[i] = \sum(p[k] * dp[i+k]) + dp[0] * p[0] + 1$

我们尴尬的发现每个 $dp[i]$ 里面都有 $dp[0]$, 但是 $dp[0]$ 也是个未知数，我们要求的答案就是 $dp[0]$

因此此题无法像之前的概率 dp 一样简单的递推出来

可以发现 $dp[i]$ 是关于 $dp[0]$ 的一个一次方程的形式

我们可以设 $dp[i] = A[i] * dp[0] + B[i]$ ($A[i], B[i]$)

如果最终能解出所有的 $A[i], B[i]$, 那么 $dp[0]$ 就可以解出来了

将这种形式代入 dp 的转移方程可以得到

$dp[i] = \sum(p[k] * (A[i+k] * dp[0] + B[i+k])) + dp[0] * p[0] + 1$

$dp[i] = \sum(p[k] * A[i+k] * dp[0] + p[k] * B[i+k]) + dp[0] * p[0] + 1$

$dp[i] = (\sum(p[k] * A[i+k]) + p[0]) * dp[0] + \sum(p[k] * B[i+k]) + 1$

根据这个式子我们可以得到 $A[i], B[i]$ 的递推方法

D 题

```
#include<stdio.h>
```

```
#include<string.h>
#include<iostream>
#include<algorithm>
using namespace std;

double A[600],B[600];
double p[100];
int main()
{
    int T;
    int k1,k2,k3,a,b,c;
    int n;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d%d%d%d%d%d%d",&n,&k1,&k2,&k3,&a,&b,&c);
        double p0=1.0/k1/k2/k3;
        memset(p,0,sizeof(p));
        for(int i=1;i<=k1;i++)
          for(int j=1;j<=k2;j++)
            for(int k=1;k<=k3;k++)
              if(i!=a||j!=b||k!=c)
                p[i+j+k]+=p0;
        memset(A,0,sizeof(A));
        memset(B,0,sizeof(B));
        for(int i=n;i>=0;i--)
        {
            A[i]=p0;B[i]=1;
            for(int j=1;j<=k1+k2+k3;j++)
            {
                A[i]+=A[i+j]*p[j];
                B[i]+=B[i+j]*p[j];
            }
        }
        printf("%.16lf\n",B[0]/(1-A[0]));
    }
```

```
    return 0;
}
```

0.5 E 题

$dp[i][j][0]$ 表示当前袋子里还有 i 只白的，j 只黑的，轮到小猫抓了，小猫赢的概率

$dp[i][j][1]$ 表示当前袋子里还有 i 只白的，j 只黑的，轮到大猫抓了，小猫赢的概率

转移的时候如果当前是小猫，直接抓了白的就结束了，抓了黑的话就要继续算子结构的概率

如果当前是大猫，分抓了黑的，跑了白的或者抓了黑的，跑了黑的两种情况进行计算

E 题

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
const int maxn = 1010;
double dp[maxn][maxn][2];
int n, m;
double dfs(int W, int B, int flag) {
  double & ret = dp[W][B][flag];
  if (ret != -1.0)
    return ret;
  ret = 0;
  if (W == 0)
    return ret = 0;
  if (B == 0)
    return ret = !flag;

  if (!flag) {
    ret += 1.0*W / (W + B);
    ret += 1.0*B / (W + B) * dfs(W, B - 1, 1);
```

```
    } else {
    //抓了一只黑的，跑了一只白的
        ret += 1.0*B / (W + B) * W / (W + B - 1) * dfs(W - 1, B - 1, 0);
        if (B > 1) //抓了一只黑的，跑了一只黑的
            ret += 1.0*B / (W + B) * (B - 1) / (W + B - 1) * dfs(W, B - 2, 0);
    }
    return ret;
}
int main() {
    while (cin >> n >> m) {
        for(int i=0;i<=n;i++)for(int j=0;j<=m;j++) dp[i][j][0]=dp[i][j][1]=-1;
        dfs(n, m, 0);
        printf("%.10lf\n", dp[n][m][0]);
    }
    return 0;
}
```