307-06-期望dp

概述

做一件事情成功：p，失败：$\overline{p}=1-p$。

和性：$E[x+y]=E[x]+E[y]$

期望的意义就是对于做一件事情，期望多少次这件事情可以做成功。

P1291 [SHOI2002]百事世界杯之旅

题目描述

“.....在2002年6月之前购买的百事任何饮料的瓶盖上都会有一个百事球星的名字。只要凑齐所有百事球星的名字，就可参加百事世界杯之旅的抽奖活动，获得球星背包，随声听，更克赴日韩观看世界杯。还不赶快行动！”

你关上电视，心想：假设有n个不同的球星名字，每个名字出现的概率相同，平均需要买几瓶饮料才能凑齐所有的名字呢?

题解

首先定义转移状态$f[i]$为收集到i个不同的球星的期望要买多少个。则$f[i]$这个状态可以从两个状态转移过来，一种是对于$f[i-1]$这种状态买了不同的一个球星，另外一种就是$f[i]$，但是买的和原来的一样。

所以我们可以粗略的得到以下的转移方程：

$$f[i]=f[i-1]\frac{n-(i-1)}{n}+f[i]\frac{i}{n} \tag{1}$$

但是我们会非常遗憾地发现，上面这个式子当中的概率总和$P_S=\frac{n-(i-1)}{n}+\frac{i}{n}=\frac{n-1}{n}$，并不为1。

所以我们要重新考虑这样做的意义何在：

- 从上面一个转移过来的情况肯定不存在问题，关键是第二个转移情况，转移失败怎么办
- 第二种情况之所以会不等于1，$P_S=1+\frac{1}{n}$，要思考的是这$\frac{1}{n}$从何而来。
- 由于对于这这道题而言，当购买结束后，就不需要购买，那么我们可以略微修改一下转移方程的意义：“买了一个，买的是自己又的的概率”==》“买一个，买的是自己有的且不是自己的概率”

- 那么这样的转换就可以消去多余的$\frac{1}{n}$

从而，我们可以得到正确的转移方程：

$$f[i] = f[i-1]\frac{n-(i-1)}{n} + f[i]\frac{i-1}{n} \tag{2}$$

最后进行以下化简，然后简单地dp就可以了。最后，这道题需要考虑分数线等神奇的情况，此为题目细节，这里不过多赘述。下面为AC代码：

```cpp
#include <iostream>
#define int long long

using namespace std;

const int maxn = 36;
int n, f_son[maxn], f_mum[maxn];

int count_digit(int n) {
    int digit = 0;
    while (n > 0) {
        n /= 10; digit ++;
    }
    return digit;
}

int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}

signed main() {
    cin >> n;
    f_son[1] = 1; f_mum[1] = 1;
    for (int k = 2; k <= n; k ++) {
        int mum = f_mum[k - 1] * (n - k + 1);
        int son = f_son[k - 1] * (n - k + 1) + n * f_mum[k - 1];
        int t = gcd(son, mum);
        mum /= t; son /= t;
        f_son[k] = son; f_mum[k] = mum;
    }
    int int_part = f_son[n] / f_mum[n];
    int new_son = f_son[n] - f_mum[n] * int_part;
    int t = gcd(new_son, f_mum[n]);
    new_son /= t; f_mum[n] /= t;
    for (int i = 1; i <= count_digit(int_part); i ++)
        cout << " ";
    cout << new_son << endl << int_part;
    for (int i = 1; i <= count_digit(f_mum[n]); i ++)
        cout << "-";
```

```
    cout << endl;
    for (int i = 1; i <= count_digit(int_part); i ++)
        cout << " ";
    cout << f_mum[n] << endl;
    return 0;
}
```

P4284 [SHOI2014]概率充电器

题目描述

著名的电子产品品牌SHOI 刚刚发布了引领世界潮流的下一代电子产品——概率充电器：

“采用全新纳米级加工技术，实现元件与导线能否通电完全由真随机数决 定！SHOI 概率充电器，您生活不可或缺的必需品！能充上电吗？现在就试试看 吧！”

SHOI 概率充电器由n-1 条导线连通了n 个充电元件。进行充电时，每条导 线是否可以导电以概率决定，每一个充电元件自身是否直接进行充电也由概率 决定。随后电能可以从直接充电的元件经过通电的导线使得其他充电元件进行 间接充电。

作为SHOI 公司的忠实客户，你无法抑制自己购买SHOI 产品的冲动。在排 了一个星期的长队之后终于入手了最新型号的SHOI 概率充电器。你迫不及待 地将SHOI 概率充电器插入电源——这时你突然想知道，进入充电状态的元件 个数的期望是多少呢？

输入格式

第一行一个整数：n。概率充电器的充电元件个数。充电元件由1-n 编号。

之后的n-1 行每行三个整数a, b, p，描述了一根导线连接了编号为a 和b 的 充电元件，通电概率为p%。

第n+2 行n 个整数：qi。表示i 号元件直接充电的概率为qi%。

输出格式

输出一行一个实数，为能进入充电状态的元件个数的期望，四舍五入到小 数点后6 位小数。

题解

首先我们看到这里每一个的贡献都是1，所以我们要求的**期望就是概率**

所以可以得出题目的所求：

$$\sum_{i=1}^{n} P_i \tag{3}$$

其中，P_i为第i个充电原件的点亮概率。

那么我们只需要列出方程，然后两边dfs就可以非常顺利地A掉这道题。

以下为递推方程，其中$f(i)$为第i个充电器点亮的概率，$\overline{f(i)} = 1 - f(i)$：

$$\overline{f(u)} = \overline{q_u} \prod_{v \in u.G,\ v \neq u.\pi} \left\{ \overline{f(v)} + f(v) \times \overline{p_{(u,v)}} \right\} \tag{4}$$

接下来，是第二遍自上而下的dfs的转移方程：

$$t = \frac{\overline{f(u)}}{\overline{f(v)} + f(v) \times p_{(u,v)}},\ f(v) = f(v) \times (\bar{t} + t \times \overline{p_{(u,v)}}) \tag{5}$$

这两个方程的含义不言而喻。

所以直接给出AC代码：

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

using namespace std;

const int maxn = 1000005;

int n;
struct node {
    int to, next;
    double w;
} g[maxn];

int ecnt = 2, head[maxn];

double f__[maxn];
double q[maxn];

void add_edge(int u, int v, double w) {
    g[ecnt] = (node) {v, head[u], w};
    head[u] = ecnt ++;
}

void dfs_1(int u, int p) {
    f__[u] = 1.0 - q[u];
    for (int e = head[u]; e != 0; e = g[e].next) {
        int v = g[e].to;
        if (v == p) continue;
        dfs_1(v, u);
        f__[u] *= f__[v] + (1 - f__[v]) * (1 - g[e].w);
    }
```

```cpp
}

void dfs_2(int u, int p) {
    for (int e = head[u]; e != 0; e = g[e].next) {
        int v = g[e].to;
        if (v == p) continue;
        double t = f__[u] / (f__[v] + (1 - f__[v]) * (1 - g[e].w));
        f__[v] *= (t + (1 - t) * (1 - g[e].w));
        dfs_2(v, u);
    }
}

int main() {
    cin >> n;
    for (int i = 1; i < n; i ++) {
        int u, v, w; cin >> u >> v >> w;
        add_edge(u, v, ((double) w) * 0.01);
        add_edge(v, u, ((double) w) * 0.01);
    }
    for (int i = 1; i <= n; i ++)
        cin >> q[i], q[i] *= 0.01;
    dfs_1(1, 0); dfs_2(1, 0);
    double ans = 0;
    for (int i = 1; i <= n; i ++)
        ans += 1 - f__[i];
    printf("%.6lf\n", ans);
    return 0;
}
```