15.2. Азартная игра

$f_m(j) = 2j$

$f_i(j) = max[2j, \sum_{k=1}^{n} p_k f_{i+1}(k)]$, i = 1, 2, ..., m - 1

In [6]:

```
import numpy as np
```

Пример 15.2-1

n = 5, m = 4,

$p_1 = 0.3, p_2 = 0.25, p_3 = 0.2, p_4 = 0.15, p_5 = 0.1$

x = 5.

In [11]:

```
f = {}

n = int(input()) #сколько чисел
m = int(input()) #сколько вращений
p = [0]
p += list(map(float, input().split())) #вероятности
x = int(input()) #цена игры

def Ef(i):
    res = 0
    for k in range(1, n + 1) :
        res += p[k] * f[i, k]
    return res

for j in range(1, n + 1):
    f[m, j] = 2 * j

for i in range(m - 1, -1, -1):
    if i == 0:
        f[i, 0] = Ef(i + 1)
        break

    print('Вращение', i, ':')
    for j in range(1, n + 1):
        f[i, j] = max(2 * j, Ef(i + 1))
        print('     Если выпало', j, 'то выгодней', end=' ')
        if (2 * j > Ef(i + 1)):
            print('закончить')
        else:
            print('продолжить')

prof = f[0, 0] - x
print('Ожидание прибыли:', prof)
```

```
5
4
0.3 0.25 0.2 0.15 0.1
5
Вращение 3 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней закончить
     Если выпало 4 то выгодней закончить
     Если выпало 5 то выгодней закончить
Вращение 2 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней закончить
     Если выпало 5 то выгодней закончить
Вращение 1 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней закончить
     Если выпало 5 то выгодней закончить
Ожидание прибыли: 2.309375
```

Упражнение 15.2-1

n = 8, m = 5,

$p_1 = p_2 = \ldots = p_8 = \frac{1}{8}$

x = 5.

In [10]:

```
f = {}

n = int(input()) #сколько чисел
m = int(input()) #сколько вращений
p = [0]
p += list(map(float, input().split())) #вероятности
x = int(input()) #цена игры

def Ef(i):
    res = 0
    for k in range(1, n + 1) :
        res += p[k] * f[i, k]
    return res

for j in range(1, n + 1):
    f[m, j] = 2 * j

for i in range(m - 1, -1, -1):
    if i == 0:
        f[i, 0] = Ef(i + 1)
        break

    print('Вращение', i, ':')
    for j in range(1, n + 1):
        f[i, j] = max(2 * j, Ef(i + 1))
        print('     Если выпало', j, 'то выгодней', end=' ')
        if (2 * j > Ef(i + 1)):
            print('закончить')
        else:
            print('продолжить')

prof = f[0, 0] - x
print('Ожидание прибыли:', prof)
```

```
8
5
0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
5
Вращение 4 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней продолжить
     Если выпало 5 то выгодней закончить
     Если выпало 6 то выгодней закончить
     Если выпало 7 то выгодней закончить
     Если выпало 8 то выгодней закончить
Вращение 3 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней продолжить
     Если выпало 5 то выгодней продолжить
     Если выпало 6 то выгодней закончить
     Если выпало 7 то выгодней закончить
     Если выпало 8 то выгодней закончить
Вращение 2 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней продолжить
     Если выпало 5 то выгодней продолжить
     Если выпало 6 то выгодней продолжить
     Если выпало 7 то выгодней закончить
     Если выпало 8 то выгодней закончить
Вращение 1 :
     Если выпало 1 то выгодней продолжить
     Если выпало 2 то выгодней продолжить
     Если выпало 3 то выгодней продолжить
     Если выпало 4 то выгодней продолжить
     Если выпало 5 то выгодней продолжить
     Если выпало 6 то выгодней продолжить
     Если выпало 7 то выгодней закончить
     Если выпало 8 то выгодней закончить
Ожидание прибыли: 8.3828125
```

Упражнение 15.2-2

m = 3, n = 4

$w_1 = 1050, w_2 = 1900, w_3 = 2500, w_4 = 3000$

$p_1 = p_2 = p_3 = p_4 = \frac{1}{4}$

In [13]:

```
f = {}

m = int(input()) #сколько дней
w = [0]
w += list(map(float, input().split())) #исходы
n = len(w) - 1
p = [0.0] * (n + 1)
for i in range(1, n + 1):
    p[i] = 1 / n

print('Вероятности:', p[1:])

def Ef(i):
    res = 0
    for k in range(1, n + 1) :
        res += p[k] * f[i, k]
    return res

for j in range(1, n + 1):
    f[m, j] = w[j]

for i in range(m - 1, 0, -1):
    print('День', i, ':')
    for j in range(1, n + 1):
        f[i, j] = max(w[j], Ef(i + 1))
        print('     Если предложили', w[j], 'то выгодней', end=' ')
        if (w[j] > Ef(i + 1)):
            print('закончить')
        else:
            print('продолжить')

f[0, 0] = Ef(1)

prof = f[0, 0]
print('Ожидание прибыли:', prof)
```

```
3
1050 1900 2500 3000
Вероятности: [0.25, 0.25, 0.25, 0.25]
День 2 :
     Если предложили 1050.0 то выгодней продолжить
     Если предложили 1900.0 то выгодней продолжить
     Если предложили 2500.0 то выгодней закончить
     Если предложили 3000.0 то выгодней закончить
День 1 :
     Если предложили 1050.0 то выгодней продолжить
     Если предложили 1900.0 то выгодней продолжить
     Если предложили 2500.0 то выгодней закончить
     Если предложили 3000.0 то выгодней закончить
Ожидание прибыли: 2590.625
```

15.3. Задача инвестирования

$x_{i+1} = (1 + r_k)y_i + (x_i - y_i) = r_k y_i + x_i$, k = 1, 2, .., m

$f_i(x_i) = \max_{0 \le y_i \le x_i} [\sum_{k=1}^{m} p_k f_{i+1}(x_i + r_k y_i)]$, i = 1, 2, ..., n,

где $f_{n+1}(x_{n+1}) = x_{n+1}$, т. к. после n-го года инвестиции нет. Отсюда следует, что

$f_n(x_n) = \max_{0 \le y_n \le x_n} [\sum_{k=1}^{m} p_k(x_n + r_k y_n)] = x_n \sum_{k=1}^{m} p_k(1 + r_k) = x_n(1 + p_1 r_1 + p_2 r_2 + \ldots + p_m r_m)$

Но тогда по индукции можно доказать, что:

$f_i(x_i) = x_i(1 + p_1 r_1 + p_2 r_2 + \ldots + p_m r_m)^{n-i+1}$

Пример 15.3-1

C = $10 000, n = 4, m = 3,

$p_1 = 0.4, p_2 = 0.2, p_3 = 0.4,$

$r_1 = 2, r_2 = 0, r_3 = -1.$

Вопрос: Правильно ли я понял, что выгодно или каждый год инвестировать все деньги или не инвестировать вообще?

In [8]:

```
d = {}
# f(i, x) - максимальное кол-во денег в конце игры если в начале i-го года: x руб

C = int(input()) #start capital
n = int(input()) #years
m = int(input()) #conditions
p = [0] #вероятности
p += list(map(float, input().split()))
r = [0] #процентные ставки
r += list(map(float, input().split()))

const = 1 + np.dot(p, r)

def f(i, x):
    if i == n + 1:
        return x
    elif (i, x) in d:
        return d[i, x]
    else:
        d[i, x] = x * const ** (n - i + 1)
        return d[i, x]

if const < 1:
    print('Выгодней вообще не инвестировать')
else:
    print('Выгодней инвестировать каждый год все имеющиеся деньги')

print('')

def k(i):
    return const ** (n - i + 1)

for i in range (1, n + 1):
    print('Коэффициент увелечения C после', i, 'года:', k(i))

print('')
print('После инвестирования:', f(1, C))
```

```
10000
4
3
0.4 0.2 0.4
2 0 -1
Выгодней инвестировать каждый год все имеющиеся деньги

Коэффициент увелечения C после 1 года: 3.8415999999999992
Коэффициент увелечения C после 2 года: 2.7439999999999993
Коэффициент увелечения C после 3 года: 1.9599999999999997
Коэффициент увелечения C после 4 года: 1.4

После инвестирования: 38415.99999999999
```

Упражнение 15.3-1

C = $10 000, n = 4, m = 3,

1-ый год: $r_1 = 2, r_2 = 1, r_3 = 0.5; p_1 = 0.1, p_2 = 0.4, p_3 = 0.5$

2-ой год: $r_1 = 1, r_2 = 0, r_3 = -1; p_1 = 0.4, p_2 = 0.4, p_3 = 0.2$

3-ий год: $r_1 = 4, r_2 = -1, r_3 = -1; p_1 = 0.2, p_2 = 0.4, p_3 = 0.4$

4-ый год: $r_1 = 0.8, r_2 = 0.4, r_3 = 0.2; p_1 = 0.6, p_2 = 0.2, p_3 = 0.2$

In [16]:

```
C = int(input()) #start capital
n = int(input()) #years
m = int(input()) #conditions
p = [[0]] #вероятности
r = [[0]] #процентные ставки
for i in range (1, n + 1):
    p.append([0])
    p[i] += list(map(float, input().split()))
    r.append([0])
    r[i] += list(map(float, input().split()))

const = [0] * (n + 1)
for i in range(1, n + 1):
    const[i] = 1 + np.dot(p[i], r[i])

fut = 1
for i in range(n, 0, -1):
    if const[i] <= 1:
        print(i, 'год: лучше не инвестировать')
    else:
        fut *= const[i]
        print(i, 'год: нужно инвестировать все деньги')

print('После инвестирования:', fut * C)
```

```
1000
4
3
0.1 0.4 0.5
2 1 0.5
0.4 0.4 0.2
1 0 -1
0.2 0.4 0.4
4 -1 -1
0.6 0.2 0.2
0.8 0.4 0.2
4 год: нужно инвестировать все деньги
3 год: лучше не инвестировать
2 год: нужно инвестировать все деньги
1 год: нужно инвестировать все деньги
После инвестирования: 3552.0
```

Входные данные:

10000

4

3

0.1 0.4 0.5

2 1 0.5

0.4 0.4 0.2

1 0 -1

0.2 0.4 0.4

4 -1 -1

0.6 0.2 0.2

0.8 0.4 0.2

Упражнение 15.3-2

Вопрос: что такое вероятность спроса? Как она связана с тем, сколько товара типа T у нас купят, если мы прозвели его x штук?

Примечание: вопрос решён

Упражнение 15.3-3

Дано:

- n = 3
- $p(D = 1) = 0.5, p(D = 2) = 0.3, p(D = 3) = 0.2$
- $c_i \in \{0, 1, 2, 3\}$ - произвели компьютеров за i-ый год
- price = 5, over_fine = 1, lack_fine = 2

Описание алгоритма

Пусть:

$OverCount_i$ - кол-во компьютеров, произведенных в i-ом году и нереализованных в i-ом году (избыток производства).

$LackCount_i$ - кол-во компьютеров, затребованых клиентами, но не произведенных в i-ом году (недостаток производства).

D_i - число компьютеров, которое надо произвести в i-ом году, чтобы идеально удовлетворить спрос (учитывая наши избытки и долги с прошлого года).

$\Delta D_i = \begin{cases} OverCount_i, \text{ если } D_i > c_i \\ 0, \text{ если } D_i = c_i \\ LackCount_i, \text{ если } D_i < c_i \end{cases} = D_i - c_i$ - разница между "идеальным объёмом производста" и тем, сколько компания произвела.

$p = \sum_{i=1}^{3} \Delta p_i$ - итоговая прибыль компании, которую мы хотим максимизировать.

$\Delta p_i = c_i * price - OverCount_i * \text{over\_fine} - LackCount_i * \text{lack\_fine}$ - прибыль за i-ый год.

Будем решать задачу жадно, т. е. действуя максимально эффективно на каждом шаге. Каждый год будем стараться максимизировать Δp_i - прибыль за этот год. Для этого нужно максимально возможно приблизить c_i к D_i. Но D_i может быть дробным числом, в то время как c_i обязано быть целым. Тогда пусть $f(D_i)$ - ф-ия, которая будет округлять D_i до целого числа, так чтобы Δp_i была максимальна. Также разумно считать, что $D_i = ED + \Delta D_{i-1}$, где ED - мат. ожидание спроса.

Тогда будем поддерживать следующие инварианты:

1) $D_i = ED + \Delta D_{i-1}$

2) $\text{c}_i = \begin{cases} 0, \text{ если } f(D_i) < 0 \\ f(D_i), \text{ если } 0 \le f(D_i) \le 3 \\ 3, \text{ если } f(D_i) > 3 \end{cases}$

3) $\Delta D_i = D_i - c_i$

4) $OverCount_i = max(0, \Delta D_i)$

5) $LackCount_i = max(0, -\Delta D_i)$

6) $\Delta p_i = c_i * price - OverCount_i * \text{over\_fine} - LackCount_i * \text{lack\_fine}$

Осталось определить ф-ию $f(D_i)$:

$$f(D_i) = \begin{cases} \lfloor D_i \rfloor, \text{ если } \Delta p_i(c_i = \lfloor D_i \rfloor) > \Delta p_i(c_i = \lceil D_i \rceil) \\ \lceil D_i \rceil, \text{иначе} \end{cases}$$

Неравенство $\Delta p_i(c_i = \lfloor D_i \rfloor) > \Delta p_i(c_i = \lceil D_i \rceil)$ для простоты будем проверять простой подстановкой.

In [ ]: