

# АНАЛИЗ И РАЗРАБОТКА АЛГОРИТМОВ

**Комбинаторика**  
**Задача о рюкзаке**

Беляков А.Ю.

# ВОПРОСЫ

Жадный алгоритм

Рекурсивный алгоритм

Бинарные маски

Динамическое программирование

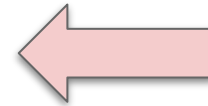
# Постановка задачи



# РЕШАЮЩИЕ АЛГОРИТМЫ

## Жадный алгоритм

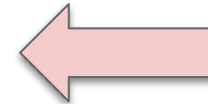
дискретный



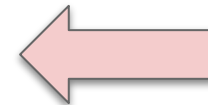
непрерывный

## Порождение подмножеств

рекурсия

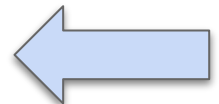


бинарные маски



## Порождение перестановок (рекурсия)

## Динамическое программирование



# ЖАДНЫЙ АЛГОРИТМ

# ЖАДНЫЙ АЛГОРИТМ. ПРЕДЕЛЫ ПРИМЕНИМОСТИ

**Пример задачи. Задача о заполнении рюкзака.**

Есть контейнер для перевозки вещей.

Для контейнера есть ограничение по максимально возможной массе вещей, которые он может взять.

Есть совокупность предметов, для каждого известны масса и ценность.

Требуется максимизировать прибыль от контейнерной перевозки. То есть набрать из предоставленных предметов максимально возможную стоимость, которую можно перевезти за один приём.

# ЖАДНЫЙ АЛГОРИТМ

какие  
принципы  
сортировки?

№	Масса	Ценность	
1	50	100	
2	40	90	
3	30	80	
4	20	70	

вместимость контейнера  
100 кг

# ЖАДНЫЙ АЛГОРИТМ

190

№	Масса	Ценность	
1	50	100	
2	40	90	
3	30	80	
4	20	70	

сортируем по стоимости



# ЖАДНЫЙ АЛГОРИТМ

240

№	Масса	Ценность	Ц / М
1	50	100	2,00
2	40	90	2,25
3	30	80	2,66
4	20	70	3,50

сортируем по относительной  
стоимости

# ЖАДНЫЙ АЛГОРИТМ

250

	№	Масса	Ценность	Ц / М
★	1	50	100	2,00
	2	40	90	2,25
★	3	30	80	2,66
★	4	20	70	3,50

оптимальное решение

# Жадный алгоритм

реализация

# РЕКУРСИВНЫЙ АЛГОРИТМ

# ТЕСТ

```
1 def F(n):  
2     if n > 0:  
3         F(n // 4)  
4         print(n, end=' ')  
5         F(n - 1)
```

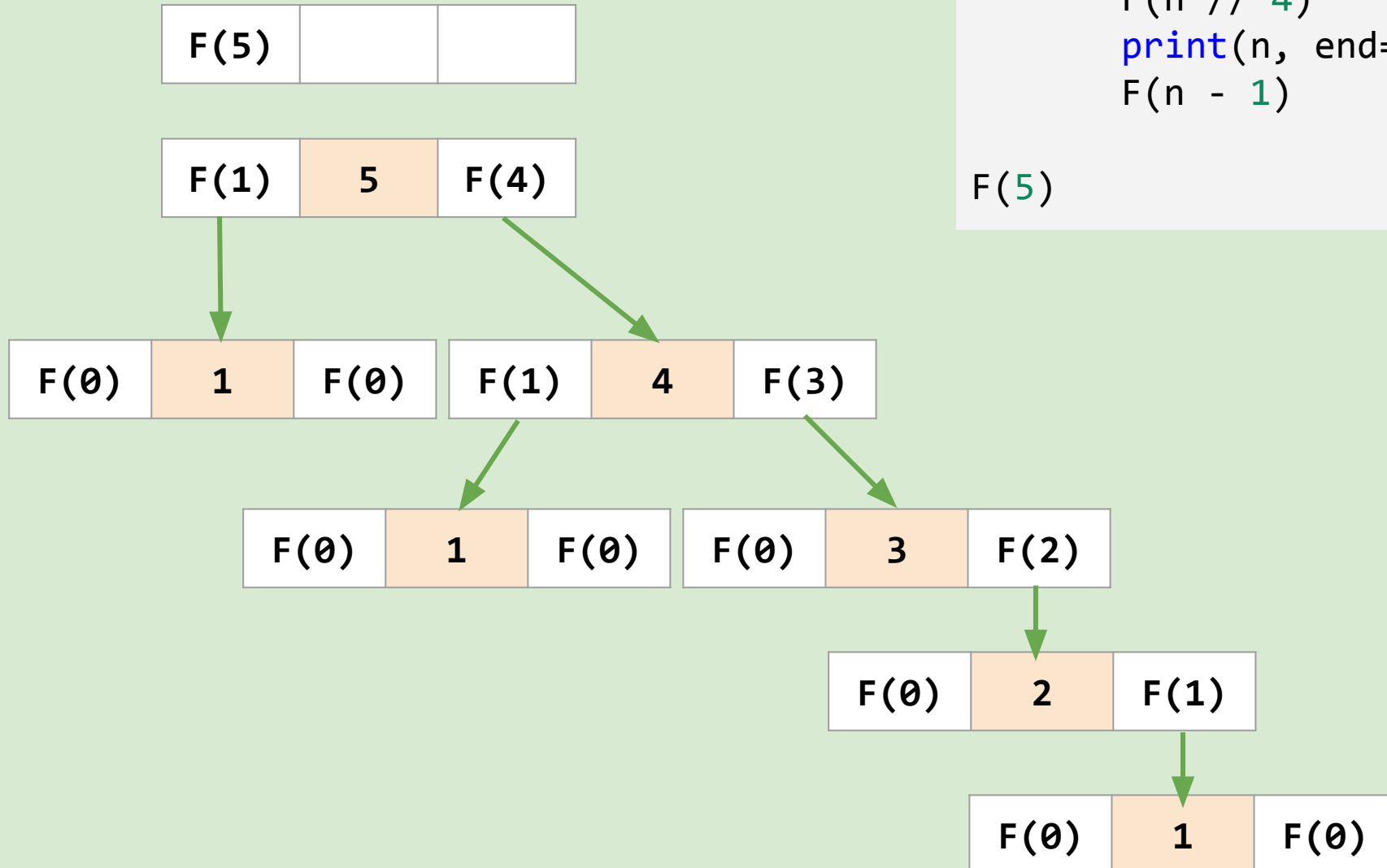
```
10 F(5)
```

что выведет на экран ?

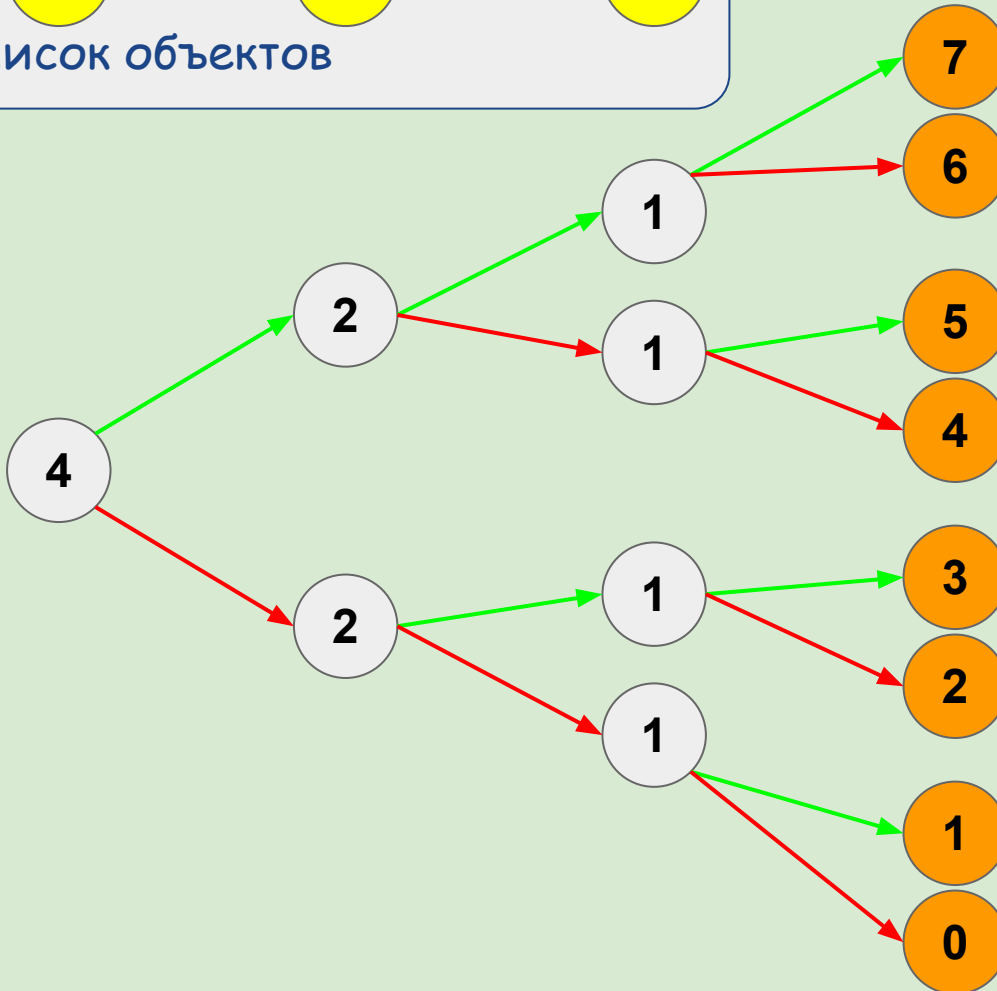
# TECT

```
def F(n):  
    if n > 0:  
        F(n // 4)  
        print(n, end=' ')  
        F(n - 1)
```

F(5)

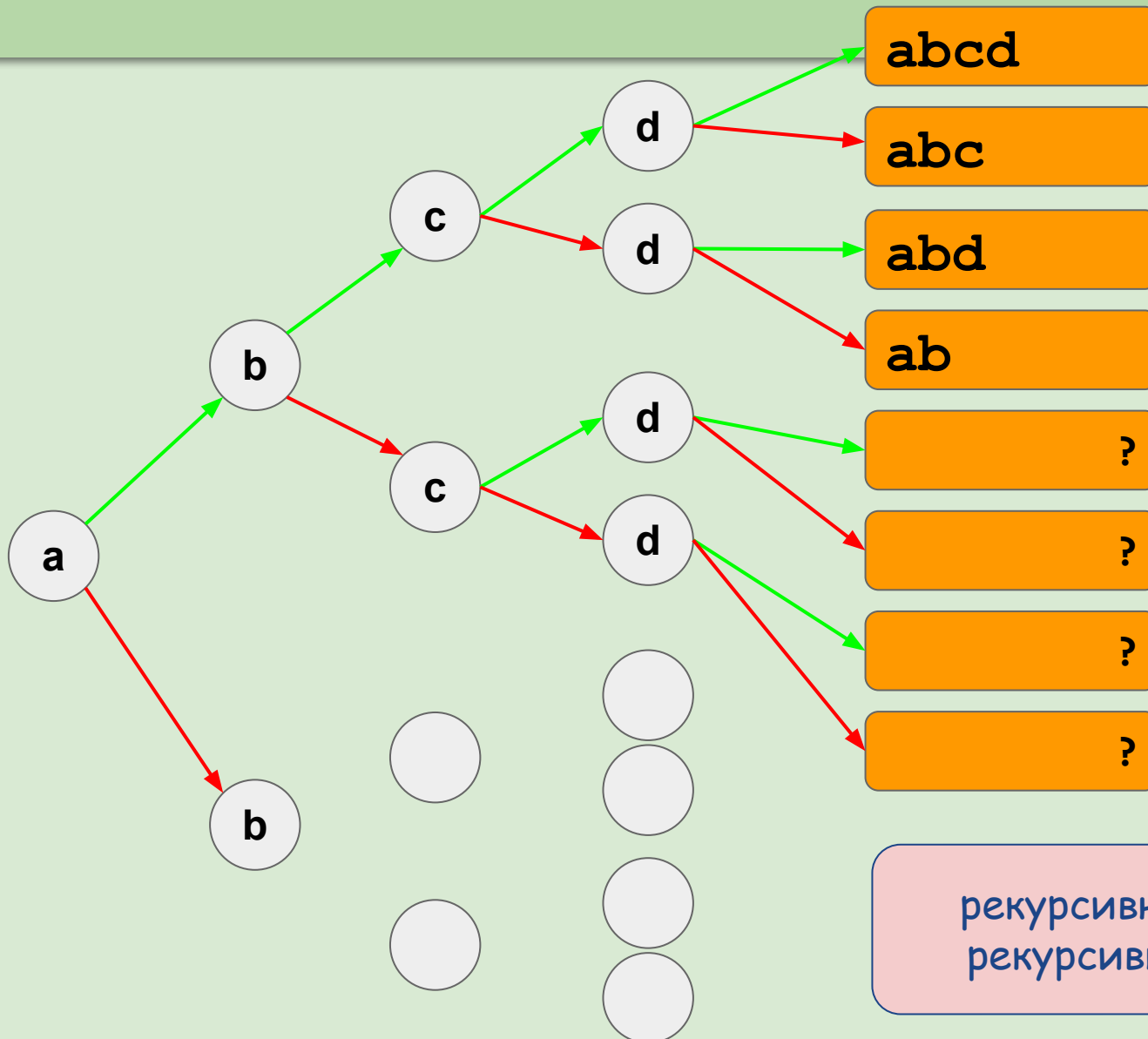


# Рекурсивно перебрать бинарное дерево



бинарный поиск

# Рекурсия



рекурсивный поиск или  
рекурсивный генератор



# Как добавлять в конец списка и убирать

СПИСОК КАК СТЕК

```
1 lst = []
2
3 for i in 1,2,3,4,5:
4     lst.append(i)
5
6 print(*lst)
7
8 lst.pop()
9
10 print(*lst)
11
12
13
14
```

1	2	3	4	5
1	2	3	4	

# Как добавлять в конец списка и убирать

СПИСОК КАК СТЕК

```
1 lst = []
2 for i in 1,2,3,4,5:
3     lst.append(i)
4
5 lst.pop()
6
7 for i in 6,7,8,9:
8     if i%2==0:
9         lst.append(i)
10    else:
11        lst.pop()
12
13 print(*lst)
14
```

?

# Рекурсия

```
1 lst = ['a', 'b', 'c', 'd']
2 tmp = []
3
4
5 def getSubsetsRec(k):
6     global tmp
7     if k == len(lst):
8         print(*tmp)
9     else:
10        tmp.append(lst[k])
11        getSubsetsRec(k+1)
12        tmp.pop()
13        getSubsetsRec(k+1)
14
```

СПИСОК КАК СТЕК

# Рекурсия

```
1 lst = ['a', 'b', 'c', 'd']
```

исходный список

```
2 tmp = []
```

место сборки комбинации

```
3  
4  
5 def getSubsetsRec(k):  
6     global tmp
```

```
7     if k == len(lst):  
8         print(*tmp)
```

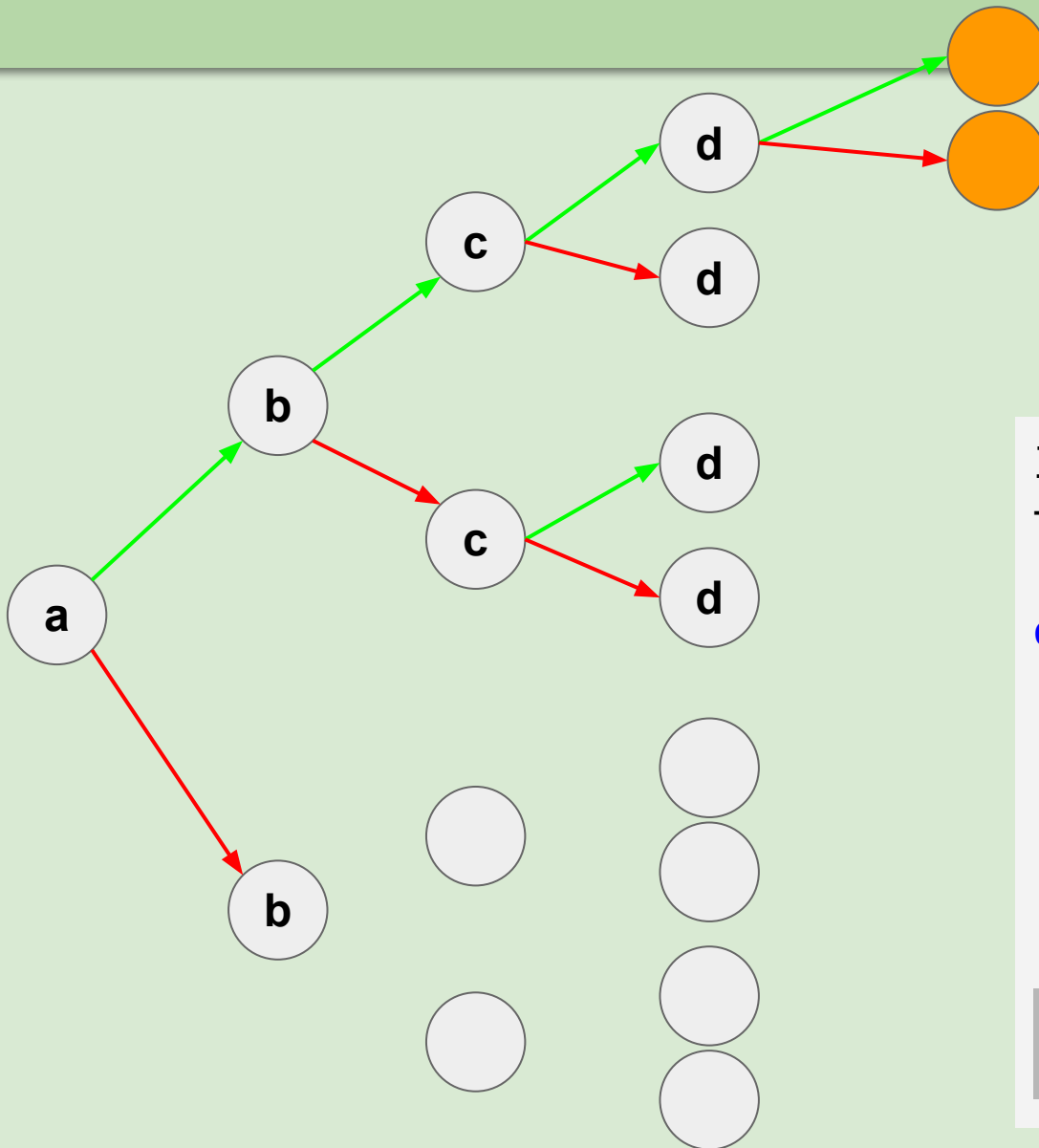
базис рекурсии

```
9     else:
```

```
10         tmp.append(lst[k])  
11         getSubsetsRec(k+1)  
12         tmp.pop()  
13         getSubsetsRec(k+1)  
14
```

шаг рекурсии

# Рекурсия



```
lst = ['a', 'b', 'c', 'd']  
tmp = []
```

```
def getSubsetsRec(k):  
    global tmp  
    if k == len(lst):  
        print(*tmp)  
    else:  
        tmp.append(lst[k])  
        getSubsetsRec(k+1)  
        tmp.pop()  
        getSubsetsRec(k+1)
```

# РЕКУРСИВНЫЙ АЛГОРИТМ

Реализация

# БИНАРНЫЕ МАСКИ

# БИНАРНЫЕ МАСКИ

	a	b	c	d
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
14	1	1	1	0
15	1	1	1	1



# БИНАРНЫЕ МАСКИ

Реализация

# ДИНАМИКА

# ДИНАМИКА

1

	a	b	c	d
0	[]	[]	[]	[]
10	[]	[]	[]	[]
20	a	a	a	a
30	a			
40	a			
50	a			
60	a			
70	a			
80	a			
90	a			
100	a			

# ДИНАМИКА


2

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]			
40	[20, 70]			
50	[20, 70]			
60	[20, 70]			
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			

# ДИНАМИКА

3

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[_ , _]		
40	[20, 70]			
50	[20, 70]			
60	[20, 70]			
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			



# ДИНАМИКА

4

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]			
50	[20, 70]			
60	[20, 70]			
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			

# ДИНАМИКА

5

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]			
60	[20, 70]			
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			

# ДИНАМИКА

6

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]	[50, 150]	[50, 150]	[50, 150]
60	[20, 70]			
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			



# ДИНАМИКА

7

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]	[50, 150]	[50, 150]	[50, 150]
60	[20, 70]	[50, 150]	[60, 160]	[60, 160]
70	[20, 70]			
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			

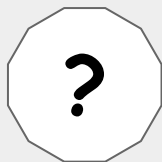
# ДИНАМИКА

8

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]	[50, 150]	[50, 150]	[50, 150]
60	[20, 70]	[50, 150]	[60, 160]	[60, 160]
70	[20, 70]	[50, 150]	[70, 170]	[70, 170]
80	[20, 70]			
90	[20, 70]			
100	[20, 70]			

weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]	[50, 150]	[50, 150]	[50, 150]
60	[20, 70]	[50, 150]	[60, 160]	[60, 160]
70	[20, 70]	[50, 150]	[70, 170]	[70, 170]
80	[20, 70]	[50, 150]	[70, 170]	[80, 180]
90	[20, 70]	[50, 150]	[__, __]	[__, __]
100	[20, 70]	[50, 150]	[__, __]	[__, __]

# ДИНАМИКА



weight	20/70	30/80	40/90	50/100
0	[0, 0]	[0, 0]	[0, 0]	[0, 0]
10	[0, 0]	[0, 0]	[0, 0]	[0, 0]
20	[20, 70]	[20, 70]	[20, 70]	[20, 70]
30	[20, 70]	[30, 80]	[30, 80]	[30, 80]
40	[20, 70]	[30, 80]	[40, 90]	[40, 90]
50	[20, 70]	[50, 150]	[50, 150]	[50, 150]
60	[20, 70]	[50, 150]	[60, 160]	[60, 160]
70	[20, 70]	[50, 150]	[70, 170]	[70, 170]
80	[20, 70]	[50, 150]	[70, 170]	[80, 180]
90	[20, 70]	[50, 150]	[90, 240]	[__, __]
100	[20, 70]	[50, 150]	[__, __]	[__, __]

# ДИНАМИКА

Реализация