Динамическое программирование в задачах обработки последовательностей ЕГЭ по информатике

2 часть

Старший методист ЦИТ
ГАУ ДПО ЯО ИРО
С.Ю. Белянчева

Домашнее задание: написать решение задач

1. Заполнение рюкзака: дано N предметов массой m1,…,mN и стоимостью c1,…,cN соответственно. Ими наполняют рюкзак, который выдерживает вес не более M. Какую наибольшую стоимость могут иметь предметы в рюкзаке?

Другими словами, нужно определить набор бинарных величин (b1, b2, …, bn), при котором выполняется соотношение:
$b1 \cdot m1 + b2 \cdot m2 + \ldots + bn \cdot mn \leq M$ и $b1 \cdot c1 + b2 \cdot c2 + \ldots + bn \cdot cn$ максимальна

A(n, m) - максимальная стоимость предметов, которые можно уложить в рюкзак максимальной вместимости m, если можно использовать только первые n предметов из заданных N.

Если ни один предмет брать нельзя, т.е. n = 0, то A(0, m) = 0, для любого m.
Если вместимость рюкзака равна 0, т.е. m = 0 , то A(n, 0) = 0, для любого n.

Теперь составим рекуррентное соотношение.
A(n,m) = max(A(n-1, m), A(n-1, m-mn)+cn)

```python
N, M = map(int, input().split()) # N - количество предметов
                                 # M - предельная масса

mas = [int(i) for i in input().split()] # массы
cen = [int(i) for i in input().split()]   # стоимости
##mas = [1, 3, 4]
##cen = [1500, 2000, 3000]
mas.insert(0, 0)
cen.insert(0, 0)
A = []
for i in range(N+1):
    A.append([0]*(M+1))

for n in range(1, N+1):
    for m in range(M+1):
        A[n][m] = A[n-1][m]
        if (m >= mas[n] and A[n-1][m-mas[n]] + cen[n] > A[n][m]):
            A[n][m] =  A[n-1][m-mas[n]] + cen[n]

print(A[N][M])
```

Домашнее задание: написать решение задач

2. Дано N золотых слитков массой m1,…,mN. Ими наполняют рюкзак, который выдерживает вес не более M. Можно ли набрать вес в точности M?

Проверим, можно ли выбрать поднабор слитков с общим весом M , если имеем слитков весом m1, …, m(n-1)?
Случай 1: рюкзак вместимостью M может быть заполнен первыми n-1 слитками.
Случай 2: мы можем убрать слиток с весом m(n-1) из рюкзака, и вес оставшихся слитков составит M-m(n-1). Таким образом, рюкзак с вместимостью M-m(n-1) можно полностью заполнить первыми n-1 слитками.
В обоих случаях мы свели задачу к практически такой же, но с меньшим количеством слитков и меньшей вместимостью рюкзака.

Домашнее задание: написать решение задач

2. Дано N золотых слитков массой m1,…,mN. Ими наполняют рюкзак, который выдерживает вес не более M. Можно ли набрать вес в точности M?

Переменная p(m,i) будет иметь значение True , если существует возможность заполнить рюкзак с вместимостью m первыми i слитками, и значение False в остальных случаях. Анализ двух вышеприведённых случаев приводит нас к следующему рекуррентному соотношению для i>0:

$$p(m, i) = p(m, i-1) \text{ или } p(m - m_{i-1}, i-1)$$

Обратите внимание, что при $m_{i-1} > m$ второе выражение не имеет смысла.

p(0,0) = True, p(0,m) = False для любого m>0.

Домашнее задание: написать решение задач

2. Дано N золотых слитков массой m1,…,mN. Ими наполняют рюкзак, который выдерживает вес не более M. Можно ли набрать вес в точности M?

$$p(m,i) = \begin{cases} \text{True, если } i = 0, m = 0 \\ \text{False, если } i = 0, m > 0 \\ p(m, i-1), \text{если } i > 0, m_{i-1} > m \\ p(m, i-1) \text{ или } p(m - m_{i-1}, i-1) \text{ в остальных случаях} \end{cases}$$

ЕГЭ

Задание 18

В задачах, которые предлагаются в этом задании КИМ, нужно найти оптимальный путь для Робота, который перемещается на клетчатом поле. Робот может на каждом шаге выбирать одно из нескольких направлений движения (например, только вправо и вниз).

Для решения задачи можно применить метод динамического программирования, потому что:

- Задачу можно разбить на подзадачи того же типа.
- Два пути Робота могут пересекаться.
- В каждой клетке можно хранить определённое состояние.

Задание 18

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	99	77	66	43	73	71	28	58	9	50	80	44	75	26	27	14	56	76	29	17
2	17	20	60	74	67	58	8	75	44	12	15	12	33	39	67	35	71	14	69	12
3	26	7	48	25	22	53	35	58	75	76	30	73	8	91	78	81	37	75	25	17
4	70	79	39	36	14	79	6	11	59	25	80	36	11	61	76	50	88	74	11	13
5	72	77	61	49	14	31	42	45	66	45	39	51	52	69	61	37	48	64	28	24
6	20	49	31	6	23	63	18	13	13	8	9	19	18	67	46	50	79	41	37	12
7	18	75	64	43	59	12	10	18	14	8	19	8	8	40	59	81	50	41	21	18
8	50	67	67	45	48	48	15	8	14	12	9	7	9	70	56	50	81	61	45	11
9	13	60	60	73	67	63	11	5	13	19	13	20	11	39	60	62	35	71	14	8
10	92	55	25	71	39	37	20	13	19	12	9	8	12	22	26	16	37	78	48	21
11	74	74	92	82	58	40	12	6	9	11	18	20	5	55	52	65	49	17	63	14
12	50	42	48	29	99	61	11	15	14	6	17	15	8	28	70	30	75	26	41	25
13	83	82	30	29	71	56	20	7	18	5	10	16	18	11	14	52	59	65	41	23
14	36	38	61	33	58	78	11	62	15	31	62	42	29	55	25	36	52	18	64	16
15	42	45	94	85	89	63	17	22	77	52	44	65	32	63	11	74	53	56	28	17
16	61	32	25	85	42	79	68	64	36	25	53	56	46	50	55	71	62	44	42	21
17	51	77	100	52	100	16	50	41	62	45	20	12	79	69	63	70	51	23	34	7
18	61	37	92	99	48	38	17	45	48	54	73	63	64	66	62	32	9	77	60	71
19	33	36	88	84	73	76	38	78	16	76	36	17	16	19	13	33	20	16	37	38
20	76	73	76	29	49	28	16	73	54	30	44	44	49	12	36	71	70	56	18	14

Квадра
переме
команд
в сосед
Квадра
также м
Перед
достои
это так
В «угл
стенам
считает
включа
накопл

Определите максимальную и минимальную денежные суммы, среди всех возможных итоговых сумм, которые может собрать Робот, пройдя из левой верхней клетки в конечную клетку маршрута. В ответе укажите два числа – сначала максимальную сумму, затем минимальную.

f(i,j) — максимальная стоимость до ячейки (i,j)

Определение рекуррентных соотношений:

$f(i,j)=\max(f(i-1,j)+a[i-1][j], f(i,j-1)+a[i][j-1])$

Задание 18

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	99	77	66	43	73	71	28	58	9	50	80	44	75	26	27	14	56	76	29	17
2	17	20	60	74	67	58	8	75	44	12	15	12	33	39	67	35	71	14	69	12
3	26	7	48	25	22	53	35	58	75	76	30	73	8	91	78	81	37	75	25	17
4	70	79	39	36	14	79	6	11	59	25	80	36	11	61	76	50	88	74	11	13
5	72	77	61	49	14	31	42	45	66	45	39	51	52	69	61	37	48	64	28	24
6	20	49	31	6	23	63	18	13	13	8	9	19	18	67	46	50	79	41	37	12
7	18	75	64	43	59	12	10	18	14	8	19	8	8	40	59	81	50	41	21	18
8	50	67	67	45	48	48	15	8	14	12	9	7	9	70	56	50	81	61	45	11
9	13	60	60	73	67	63	11	5	13	19	13	20	11	39	60	62	35	71	14	8
10	92	55	25	71	39	37	20	13	19	12	9	8	12	22	26	16	37	78	48	21
11	74	74	92	82	58	40	12	6	9	11	18	20	5	55	52	65	49	17	63	14
12	50	42	48	29	99	61	11	15	14	6	17	15	8	28	70	30	75	26	41	25
13	83	82	30	29	71	56	20	7	18	5	10	16	18	11	14	52	59	65	41	23
14	36	38	61	33	58	78	11	62	15	31	62	42	29	55	25	36	52	18	64	16
15	42	45	94	85	89	63	17	22	77	52	44	65	32	63	11	74	53	56	28	17
16	61	32	25	85	42	79	68	64	36	25	53	56	46	50	55	71	62	44	42	21
17	51	77	100	52	100	16	50	41	62	45	20	12	79	69	63	70	51	23	34	7
18	61	37	92	99	48	38	17	45	48	54	73	63	64	66	62	32	9	77	60	71
19	33	36	88	84	73	76	38	78	16	76	36	17	16	19	13	33	20	16	37	38
20	76	73	76	29	49	28	16	73	54	30	44	44	49	12	36	71	70	56	18	14

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
19	33	36	88	84	73	76	38	78	16	76	36	17	16	19	13	33	20	16	37	38
20	76	73	76	29	49	28	16	73	54	30	44	44	49	12	36	71	70	56	18	14
21	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	p
22	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	p
23	.	.	.	.	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	p
24	.	n	n	n	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	p
25	.	.	.	p	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	p
26	.	.	.	p	.	.	p	.	.	.	.	.	.	.	.	.	.	p	.	p
27	.	.	.	p	.	.	p	.	.	.	.	.	.	.	.	.	.	p	.	p
28	.	.	.	p	.	.	p	.	.	.	.	.	.	.	.	.	.	p	.	p
29	.	.	.	p	.	.	p	n	n	n	n	n	n	.	.	.	.	p	.	p
30	.	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	.	p	.	p
31	.	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	.	p	.	p
32	.	.	.	p	.	.	.	.	.	.	.	.	.	.	.	.	.	p	.	p
33	.	.	.	p	.	.	.	.	n	n	n	n	.	.	.	.	.	p	.	p
34	.	.	.	p	.	.	.	.	.	.	.	p	.	.	.	.	.	p	.	p
35	.	.	.	.	.	.	.	.	.	.	.	p	.	.	.	n	n	np	.	p
36	.	.	.	.	.	.	.	.	.	.	.	p	.	.	.	.	.	.	.	p
37	.	.	.	.	.	.	.	.	.	.	.	p	.	.	.	.	.	.	.	p
38	.	.	.	.	.	.	.	.	.	.	.	p	.	.	.	.	.	.	.	p
39	.	.	.	.	.	.	.	.	.	.	.	p	.	.	.	.	.	.	.	p
40	n	n	n	n	n	n	n	n	n	n	n	np	n	n	n	n	n	n	n	np

	A	B	C
22		99	176
23		116	=MAX(B22;A23)+B2

18.txt — Блокнот

Файл Правка Формат Вид Справка

```
99 77 66 43 73 71 28 58 9 50 80 44 75 26 27 14 56 76 29 17
17 20 60 74 67 58 8 75 44 12 15 12 33 39 67 35 71 14 69 12
26 7 48 25 22 53 35 58 75 76 30 73 8 91 78 81 37 75 25 17
70 79 39 36 14 79 6 11 59 25 80 36 11 61 76 50 88 74 11 13
72 77 61 49 14 31 42 45 66 45 39 51 52 69 61 37 48 64 28 24
20 49 31 6 23 63 18 13 13 8 9 19 18 67 46 50 79 41 37 12
18 75 64 43 59 12 10 18 14 8 19 8 8 40 59 81 50 41 21 18
50 67 67 45 48 48 15 8 14 12 9 7 9 70 56 50 81 61 45 11
13 60 60 73 67 63 11 5 13 19 13 20 11 39 60 62 35 71 14 8
92 55 25 71 39 37 20 13 19 12 9 8 12 22 26 16 37 78 48 21
74 74 92 82 58 40 12 6 9 11 18 20 5 55 52 65 49 17 63 14
50 42 48 29 99 61 11 15 14 6 17 15 8 28 70 30 75 26 41 25
83 82 30 29 71 56 20 7 18 5 10 16 18 11 14 52 59 65 41 23
36 38 61 33 58 78 11 62 15 31 62 42 29 55 25 36 52 18 64 16
42 45 94 85 89 63 17 22 77 52 44 65 32 63 11 74 53 56 28 17
61 32 25 85 42 79 68 64 36 25 53 56 46 50 55 71 62 44 42 21
51 77 100 52 100 16 50 41 62 45 20 12 79 69 63 70 51 23 34 7
61 37 92 99 48 38 17 45 48 54 73 63 64 66 62 32 9 77 60 71
33 36 88 84 73 76 38 78 16 76 36 17 16 19 13 33 20 16 37 38
76 73 76 29 49 28 16 73 54 30 44 44 49 12 36 71 70 56 18 14
. . . . . . . . . . . . . . . . . . . p
. . . . . . . . . . . . . . . . . . . p
. . . . . . p . . . . . . . . . . . p
. n n n . . p . . . . . . . . . . . p
. . . p . . p . . . . . . . . . . . p
. . . p . . p . . . . . . . . . . p p
. . . p . . p . . . . . . . . . . p p
. . . p . . p . . . . . . . . . . p p
. . . p . . p n n n n n . . . . . p p
. . . p . . . . . . . . . . . . . p p
. . . p . . . . . . . . . . . . . p p
. . . p . . . . . . . . . . . . . p p
. . . p . . . . . . . . . . . . . p p
. . . p . . . n n . . . . . n n np . p
. . . . . . . . . . . . . . . . . p p
. . . . . . . . . . . . . . . . . p p
. . . . . . . . . . . . . . . . . p p
n n n n n n n n n n n np n n n n n n n np
```

```
f = open('18.txt')
a=[]*20#исходные данные
walls=[]#стены

for x in range(20):
    a.append(list(map(int, f.readline().split())))

for x in range(20):
    walls.append(f.readline().split())

ans=[]#конечные желтые ячейки

#динамическая таблица
d=[-1000000]*20
for x in range(20):
    d[x]=[-1000000]*20

d[0][0] = a[0][0]
for i in range(20):
    for j in range(20):
        if 'p' not in walls[i][j]:
            d[i][j+1]= max(d[i][j+1],d[i][j]+a[i][j+1])
        if 'n' not in walls[i][j]:
            d[i+1][j]= max(d[i+1][j],d[i][j]+a[i+1][j])
        if 'np' in walls[i][j]:
            ans.append(d[i][j])
print(max(ans))
```

```
f = open('18.txt')
a=[]*20#исходные данные
walls=[]#стены

for x in range(20):
    a.append(list(map(int, f.readline().split())))

for x in range(20):
    walls.append(f.readline().split())

ans=[]#конечные желтые ячейки

#динамическая таблица
d=[0]*20
for x in range(20):
    d[x]=[1000000]*20

d[0][0] = a[0][0]
for i in range(20):
    for j in range(20):
        if 'p' not in walls[i][j]:
            d[i][j+1]= min(d[i][j+1],d[i][j]+a[i][j+1])
        if 'n' not in walls[i][j]:
            d[i+1][j]= min(d[i+1][j],d[i][j]+a[i+1][j])
        if 'np' in walls[i][j]:
            ans.append(d[i][j])
print(min(ans))
```

Задание 23

Исполнитель преобразует число на экране.
У исполнителя есть три команды, которые обозначены латинскими буквами:

***A*. Прибавить 1**
***B*. Умножить на 2**
***C*. Возвести в квадрат**

Программа для исполнителя – это последовательность команд.
Сколько существует программ, для которых при исходном числе 2 результатом является число 20, при этом траектория вычислений не содержит числа 11?
Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. *Например*, для программы ***CBA*** при исходном числе 4 траектория будет состоять из чисел 16, 32, 33.

f(n) —количество программ, ведущих в число n.

Определение рекуррентных соотношений:

$f(n) = f(n - 1) + f(n / 2) + f(\sqrt{n})$

Начальные значения:

f(2) = 1 f(1) = 0 f(0) = 0

Задание 23

Исполнитель преобразует число на экране.

У исполнителя есть три команды, которые обозначены латинскими буквами:

***A*. Прибавить 1**

***B*. Умножить на 2**

***C*. Возвести в квадрат**

Программа для исполнителя – это последовательность команд.

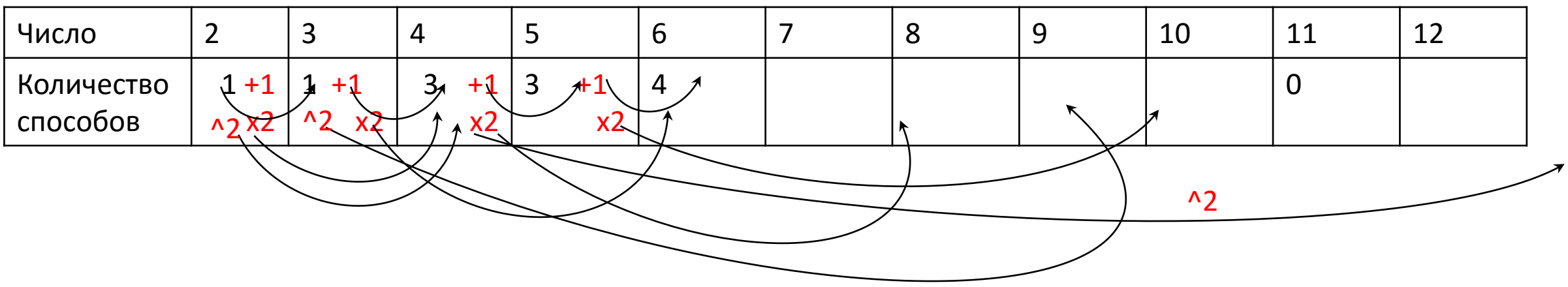
Сколько существует программ, для которых при исходном числе 2 результатом является число 20, при этом траектория вычислений не содержит числа 11?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. *Например*, для программы ***CBA*** при исходном числе 4 траектория будет состоять из чисел 16, 32, 33.

```
====
37
->
```

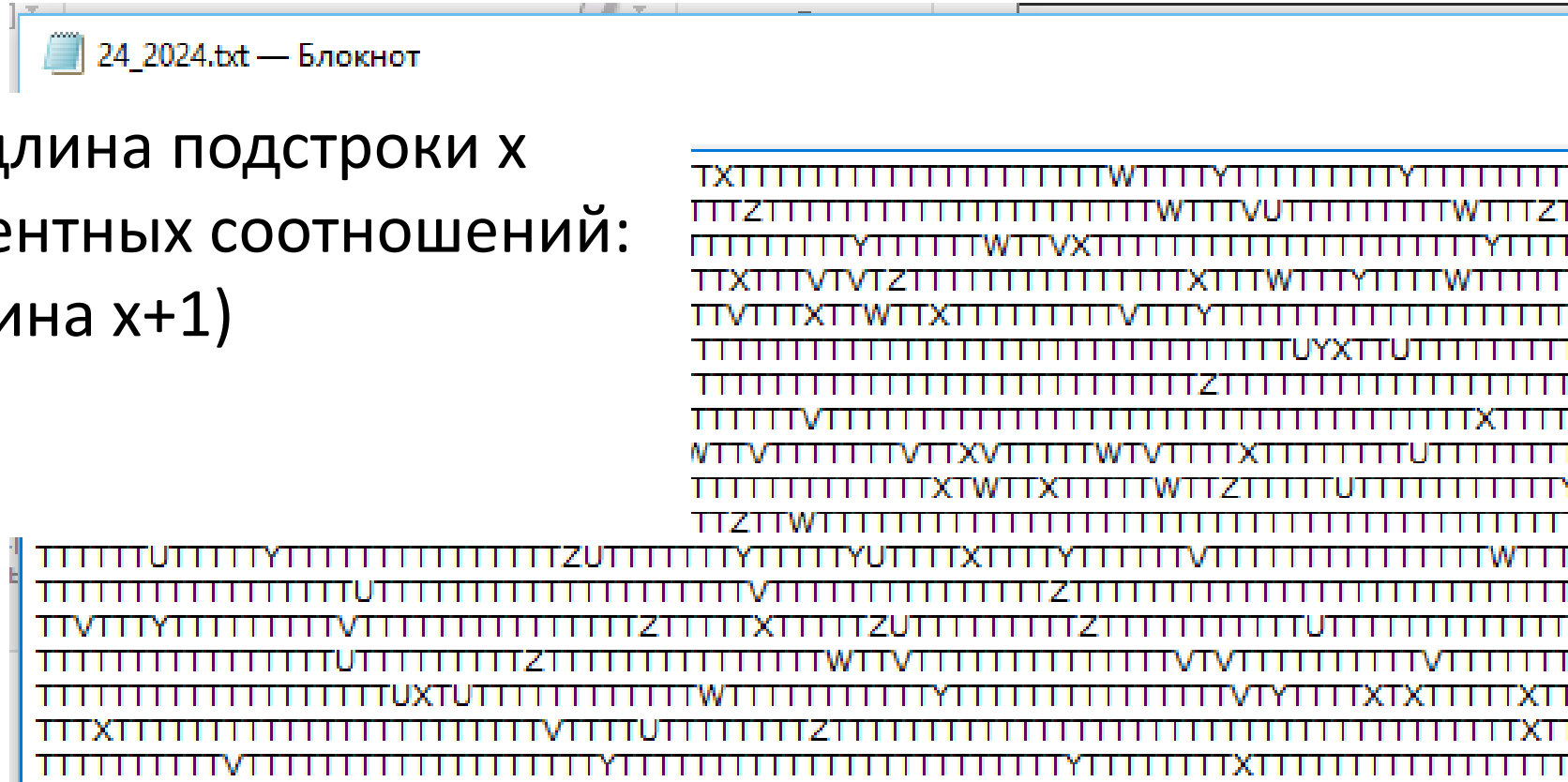
```
a=[0]*401
a[2]=1
for i in range(2,20):
    if i+1 != 11: a[i+1]+=a[i]
    if i*2 != 11: a[i*2]+=a[i]
    if i**2 != 11: a[i**2]+=a[i]
print(a[20])
```

Число	2	3	4	5	6	7	8	9	10	11	12
Количество способов	1	1	3	3	4					0	



Задание 24

Текстовый файл состоит из символов *T*, *U*, *V*, *W*, *X*, *Y* и *Z*.
Определите в прилагаемом файле максимальное количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых символ *T* встречается ровно 100 раз.
Для выполнения этого задания следует написать программу.



f(x) —максимальная длина подстроки x
Определение рекуррентных соотношений:
f(x) = max(длина x, длина x+1)
Начальные значения:
f(0) = 0

Задание 24

Текстовый файл состоит из символов *T*, *U*, *V*, *W*, *X*, *Y* и *Z*.
Определите в прилагаемом файле максимальное количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых символ *T* встречается ровно 100 раз.
Для выполнения этого задания следует написать программу.

```
f = open("24_2024.txt", "r")  #открыть файл для чтения
t = f.readline()  #читаем строку из файла
t = t.split("T")  #делим строку по символу T. Теперь это список строк
```

```
['', '', '', '', '', '', 'U', '', '', '', '', '', 'Y', '', '', 'Z', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'X', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', 'W', '', '', '', 'Y', '', '', '', '', '', '', '', '', 'Y', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
', '', '', 'U', '', 'V', '', '', '', 'XY', '', '', '', '', 'UZ', '', 'W', '', '', '']
```

Задание 24

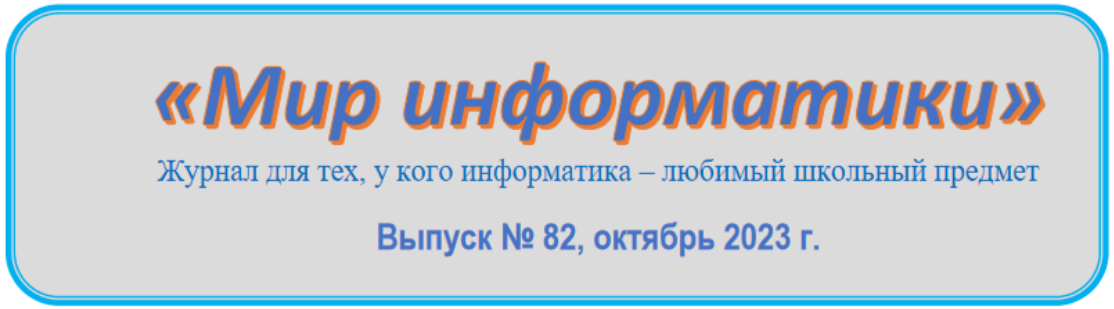
Текстовый файл состоит из символов *T*, *U*, *V*, *W*, *X*, *Y* и *Z*.
Определите в прилагаемом файле максимальное количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых символ *T* встречается ровно 100 раз.
Для выполнения этого задания следует написать программу.

```
f = open("24_2024.txt","r")#открыть файл для чтения
t = f.readline()
t = t.split("T")#делим строку по символу T. Теперь это список строк

x = 0#для хранения текущей подстроки
for i in range(100):#найдем длину первой подпоследовательности, в которой 100 T
    x = x + len(t[i]) + 1#записываем длину

f = x#максимальная длина
for i in range(100, len(t)):#перебираем подстроки
    x -= len(t(i-100))#сдвигаем начало подстроки
    x += len(t(i))#сдвигаем конец подстроки
    f = max(f, x)#записываем максимум
print(f)
```

Задание 26



https://infojournal.ru/wp-content/uploads/2023/10/mir_info-10-2023.pdf?ysclid=lq5entc53i687437057

Методика выполнения задания 26 демонстрационного варианта ЕГЭ по информатике 2024 года. Д.М. Златопольский (стр.10)

Предложен алгоритм на школьном алгоритмическом языке (система программирования КуМир). Русский синтаксис этого языка и большое число комментариев делает приводимые фрагменты программы максимально понятными и легко переводимыми на любой другой язык программирования.

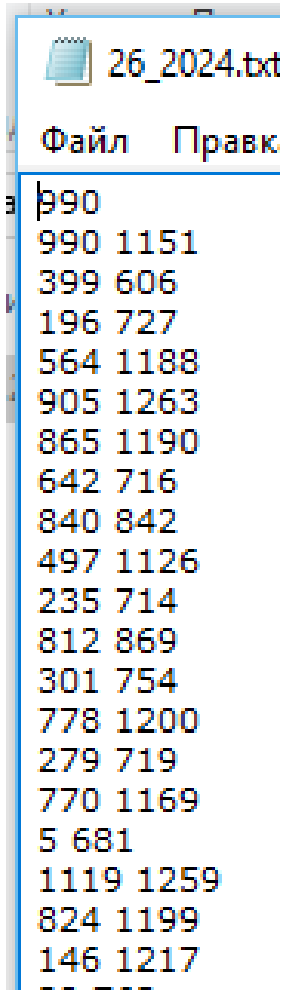
Задание 26

Входной файл содержит сведения о заявках на проведение мероприятий в конференц-зале. В каждой заявке указаны время начала и время окончания мероприятия (в минутах от начала суток). Если время начала одного мероприятия меньше времени окончания другого, то провести можно только одно из них. Если время окончания одного мероприятия совпадает со временем начала другого, то провести можно оба. Определите, какое максимальное количество мероприятий можно провести в конференц-зале и каков при этом максимально возможный перерыв между двумя последними мероприятиями.

Входные данные

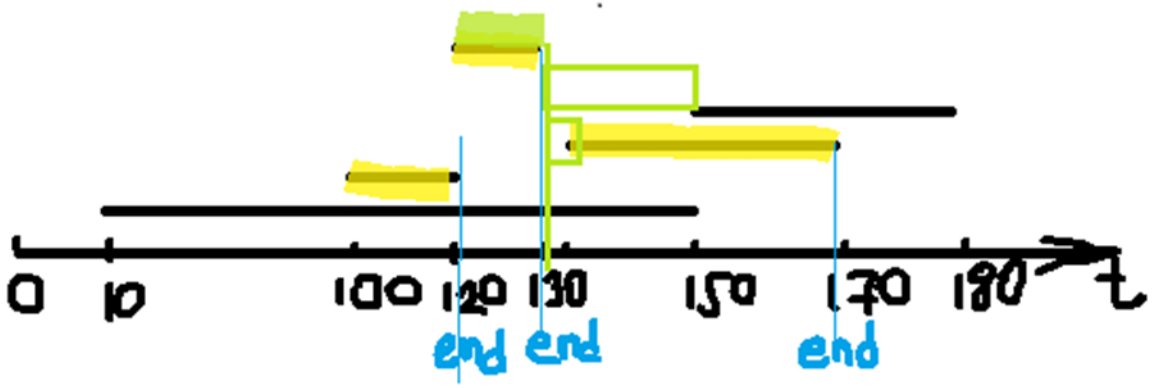
В первой строке входного файла находится натуральное число N ($N \le 1000$) – количество заявок на проведение мероприятий. Следующие N строк содержат пары чисел, обозначающих время начала и время окончания мероприятий. Каждое из чисел натуральное, не превосходящее 1440.

Запишите в ответе два числа: максимальное количество мероприятий и самый длинный перерыв между двумя последними мероприятиями (в минутах).



Задание 26

5
10 150
100 120
131 170
150 180
120 130




```
f=open('26_2024.txt')
n=int(f.readline())
a=[]
for i in range(n):
    st = f.readline().split()
    a.append((int(st[0]), int(st[1])))
```

```
= RESTART: C:/Users/svbel/Docu
[(990, 1151), (399, 606), (196
2, 716), (840, 842), (497, 112
), (279, 719), (770, 1169), (5
9, 762), (308, 704), (133, 123
87), (724, 1097), (665, 1140),
), (823, 848), (1049, 1095), (
```

Отсортируем данные по времени окончания мероприятия. Чем раньше мероприятие закончится, тем больше мы сможем их провести.

Задание 26

```
f=open('26_2024.txt')
n=int(f.readline())
a=[]
for i in range(n):
    st = f.readline().split()
    a.append((int(st[0]), int(st[1])))


# Сортируем по второму элементу
a.sort(key=lambda x: x[1])

end=0  # Окончание  мероприятия
end2=0 # Окончание предпоследнего выбранного мероприятия
k=0    # Количество мероприятий
mx_end = 0 # максимальный конец  мероприятия

# Ищем максимальное количество
for i in range(n):
    if a[i][0] >= end:
        end2 = end # Окончание предпоследнего выбранного элемента
        end =    a[i][1]
        k=k+1

# Ищем максимальное время начала последнего элемента при
# максимальном количестве
for i in range(n):
    if a[i][0] >= end2:
        mx_end=max(mx_end, a[i][0])

print(k, mx_end - end2)
```

Задание 27

К.Ю. Поляков. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В ЗАДАЧАХ ОБРАБОТКИ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ЕГЭ ПО ИНФОРМАТИКЕ (Информатика в школе • 2020 • № 5 (158))
https://kpolyakov.spb.ru/download/ege27info.pdf
А. Кабанов. Открытый курс https://kompege.ru/course
Курс Е. Джобса https://kompege.ru/jobs?theme=0&lesson=0&menu=0

Задачи на использование критериев делимости, обработка пар/троек чисел, нахождение минимальной/максимальной суммы, нахождение расстояния между элементами, использование метода частичных сумм, префиксных сумм и сдвига префиксных сумм, подпоследовательностей, использование сдвига в массиве данных, использование метода двух указателей

Задание 27

По каналу связи передаётся последовательность целых чисел – показания прибора. В течение N мин. (N – натуральное число) прибор ежеминутно регистрирует значение напряжения (в условных единицах) в электрической сети и передаёт его на сервер.
Определите три таких переданных числа, чтобы между моментами передачи любых двух из них прошло **не менее** K мин., а сумма этих трёх чисел была максимально возможной. Запишите в ответе найденную сумму.

Входные данные
Даны два входных файла (файл A и файл B), каждый из которых в первой строке содержит натуральное число K – минимальное количество минут, которое должно пройти между моментами передачи показаний, а во второй – количество переданных показаний N ($1 \le N \le 10\,000\,000$, $N > K$). В каждой из следующих N строк находится одно целое число, по модулю не превышающее 10 000 000, которое обозначает значение напряжения в соответствующую минуту.
Запишите в ответе два числа: сначала значение искомой величины для файла A, затем – для файла B.

Задание 27

По каналу связи передаётся последовательность целых чисел – показания прибора. В течение N мин. (N – натуральное число) прибор ежеминутно регистрирует значение напряжения (в условных единицах) в электрической сети и передаёт его на сервер.
Определите три таких переданных числа, чтобы между моментами передачи любых двух из них прошло **не менее** K мин., а сумма этих трёх чисел была максимально возможной. Запишите в ответе найденную сумму.

```
f = open("27_B_2024.txt","r")#открыть файл для чтения
t = f.readline()
k = int(f,readline())#минимальное количество минут
n = int(f,readline())# количество переданных показаний
d = [int(x) for x in f]# записали все числа в список
max1 = max2 = max3 = -10**10 #переменные для хранения самых больших чисел
for i in range(k*2,n):
#цикл для просмотра всех элементов, начиная с (k*2)-го до n-го
    max1 = max(max1, d[i-k*2])
    max2 = max(max2, d[i-k]+max1)
    max3 = max(max3, d[i]+max2)
print(max3)
```

Динамическое программирование в олимпиадных задачах

1. Задача о рюкзаке (Knapsack problem) - классическая задача динамического программирования, которая используется в олимпиадах. В этой задаче у нас есть набор предметов с весами и стоимостями, и нам нужно выбрать такие предметы, чтобы их общий вес не превышал заданного предела, а общая стоимость была максимальной.
2. Задачи на потоки (Flow problems) - эти задачи включают в себя нахождение максимального потока в сети, потока минимальной стоимости и т.д.
3. Задачи о перекрывающихся подмножествах (Set cover problem) - это задачи, в которых нужно найти минимальное количество подмножеств, которые покрывают все элементы данного множества.
4. Задачи об оптимальном пути (Shortest path problem) - включают в себя задачи о лабиринтах, нахождение кратчайшего пути в графе и т.п.

Динамическое программирование в олимпиадных задачах

5. Задачи коммивояжера (Travelling salesman problem) - в этих задачах коммивояжер должен посетить все города, вернувшись в исходный, найдя при этом оптимальный маршрут.
6. Задачи о разбиении (Partition problem) - здесь требуется найти такое разбиение числа на слагаемые, чтобы их сумма была наименьшей или наибольшей.
7. Задачи с использованием принципа оптимальности Беллмана (Bellman’s principle of optimality) - сюда относятся задачи с динамическим программированием по принципу Беллмана, например, задача о Ханойской башне.
8. Задачи, решаемые с помощью уравнения Вальда (Minimax theorem, или Wald’s equation) - например, классическая задача о ястребе и голубе.
9. Задачи оптимизации расписания (Job scheduling problems) - коммивояжёр с ограничением на время, планирование производства и т. д.
10. Задачи теории расписаний (Timetable scheduling problems) – оптимизация расписаний движения транспорта.