Информатика. 27 задание. Теория.

Пояснение: нижеописанное будет сказано для питона, но во многом применимо в других языках своими схожими синтаксическими конструкциями

Первый прототип, который будет разобран имеет примерно следующую формулировку:

`База-прототип: Дано целое число n, а затем подается последовательность из n натуральных чисел.`
`Требуется найти (количество, сумму, разность, макс, мин) среди пар чисел, которые идут либо подряд (a[i], a[i+1]), либо идут как угодно.`

Собственно, давайте для начала решим следующую задачу таким методом:

Дано целое число n, а затем подается последовательность из n натуральных чисел. Требуется найти максимальную сумму, состоящую из пары чисел (парой являются любые два различных элемента), которая будет четна.

Пример входных данных:

5
2
10
5
17
3

Выходные данные для приведённого выше примера: 22

Решение.

```
n = int(input())
a = []
summ = 0
for i in range(n):
    a += [int(input())] #Аналог a.append
for i in range(n):
```

```
    for j in range(i+1, n):
        if (a[i]+a[j]) % 2 == 0:
            summ = max(summ, a[i]+a[j])
print(summ)
```

Данное решение является неэффективным и при больших N время работы программы точно превысит время проведения экзамена.

Что значит эффективное решение? Для задач 27 эффективным решением считается то, которое не хранит все изначальные данные и работает за время, равное *O(N)*.

Хорошо, но что такое это *O(N)*. О-большое называется асимптотикой функции, то есть эта функция, которая определяет, что оказывает наибольший рост функции при подстановке любого n. Так, для функции $y = 100n+123456$ асимптотикой или О-большим будет является n. То есть для данной функции значительное значение для ее роста оказывает переменная n и записывается как O(N).

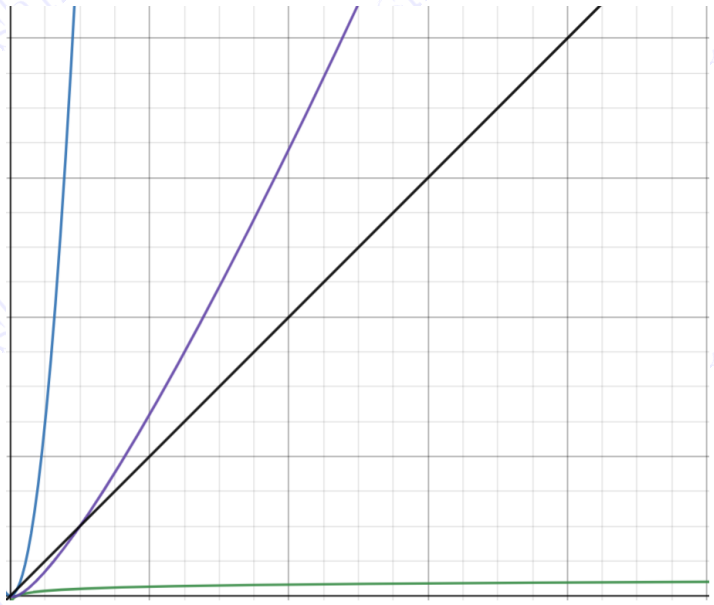
Для функции $y = n^2+10000n-100$ асимптотикой или О-большим будет является n^2. Даже несмотря на то, что есть часть 10000n все равно с ростом n функция n^2 превзойдет n, умноженную на сколь большую константу. Следовательно, данная функция имеет асимптотику $O(N^2)$.

Итак, как определить, является ли решение эффективным или нет? Давайте посмотрим на примере выше:

Можно заметить, что у нас есть цикл, размерностью n, а внутри него точно такой же. Получается, для каждого элемента массива, будет перебираться каждый последующий после него, то есть относительно каждого элемента будут рассмотрены все элементы после. Сколько нужно операций, чтобы выбрать первый элемент? N. А чтобы второй? По сути N. А значит общая сложность алгоритма (асимптотика) равна $O(N*N) = O(N^2)$. Следовательно, решение является неэффективным.

Также можно заметить, что мы сохранили все элементы последовательности в массив, чем заняли лишнюю память, чего можно было избежать.

Простой график, демонстрирующий отличия ассимптотик наглядно (функции сверху вниз — n^2, $nlg(n)$, n, $lg(n)$):



Итак, замечательно. Решим предыдущую задачу эффективно:

Дано целое число n, а затем подается последовательность из n натуральных чисел. Требуется найти максимальную сумму, состоящую из пары чисел (парой являются любые два различных элемента), которая будет четна.

Рассуждения

Что значит найти четную сумму, складывая два элемента? Когда сумма их остатков при делении на 2 будет равна 0. А такое возможно, когда оба числа имеют остаток 0, либо оба числа имеют остаток 1. Давайте на каждом шаге цикла for будем смотреть, является ли сумма вновь считанного числа и предыдущего максимального четного или нечетного максимальна. Если да — то заменяем сумму. Если нет, то сумма остается прежней. Если x является наибольшим числом в своей категории остатка на два, то обновляем максимум своей категории.

Решение.

```
n = int(input())
summ = 0
max_kr2 = 0
max_nekr2 = 0
for i in range(n):
    x = int(input())
    if x % 2 == 0:
        summ = max(summ, x+max_kr2)
        max_kr2 = max(x, max_kr2)
    else:
        summ = max(summ, x+max_nekr2)
        max_nekr2 = max(x, max_nekr2)
 print(summ)
```

Как можно видеть, никаких вложенных циклов от n у нас нет, так еще мы и не запоминаем все числа за ненадобностью, а лишь храним два максимума и итоговый ответ.

Перед решением задач (все впредь будем решать эффективно), стоит понимать, как именно можно решить и с каким подходом. Итак, есть статический подход и динамический. Первый подразумевает накопление всех основных данных для решения задачи и подсчет ответа после основного цикла for (как правило с помощи простых комбинаторных формул). В то время как второй считает ответ на каждом шаге цикла for и после его выполнения мы сразу получим ответ.

Давайте рассмотрим следующие задачи:

Задача 1

Дано целое число n, затем последовательность из n натуральных чисел. Требуется найти максимальное произведение, состоящее из пары чисел, которое будет кратно 6.

Пример входных данных:

5
2
10
5
17
3

Выходные данные для приведённого выше примера: 30

Рассуждения. Решение статикой.

Что требуется? Найти максимальное произведение, кратное 6. Тогда, глобально существует три случая:

a*b % 6 == 0, когда:

1) a — кратно 6 и b — кратно 6

2) a — кратно 6, b — обладает любой кратностью (3, 2, некратно ничему вышеперечисленному)

3) a — кратно 3, тогда b обязательно кратно 2. (и наоборот)

Тогда будем хранить максимумы в каждой из категории делимости, а для кратных 6 будем хранить не только максимум, но и предмаксимум (первый случай). Ответ подсчитаем в конце, сравнив все произведения из вышеуказанных.

Решение.

```
n = int(input())
max_kr_6 = 0
pr_kr_6 = 0
kr_3 = 0
kr_2 = 0
nekr = 0
for i in range(n):
    x = int(input())
```

```
        if x % 6 == 0:
            if x > max_kr_6:
                pr_kr_6 = max_kr_6
                max_kr_6 = x
            else:
                pr_kr_6 = max(x, pr_kr_6)
        elif x % 3 == 0:
            kr_3 = max(x, kr_3)
        elif x % 2 == 0:
           kr_2 = max(kr_2, x)
       else:
            nekr = max(nekr, x)
    print( max(max_kr_6*pr_kr_6, \
            max_kr_6*kr_3, max_kr_6*kr_2, max_kr_6*nekr, \
             kr_3*kr_2) )
```

Теперь динамика.

Рассуждения. Решение динамикой.

Идея точно такая же, НО! Вместо хранения двух максимумов для кратности 6, будем хранить 1 и на каждом шаге цикла for сразу будем считать ответ и при надобности обновлять максимумы каждой из категории.

```
n = int(input())
max_kr_6, kr_3 = 0, 0
kr_2, nekr = 0, 0
ans = 0
for i in range(n):
    x = int(input())
    if x % 6 == 0:
        ans = max(ans, x*max_kr_6, x*kr_3, x*kr_2, x*nekr)
        max_kr_6 = max(x, max_kr_6)
    elif x % 3 == 0:
        ans = max(ans, max_kr_6*x, x*kr_2)
```

```
        kr_3 = max(x, kr_3)
    elif x % 2 == 0:
        ans = max(ans, max_kr_6*x, x*kr_3)
        kr_2 = max(x, kr_2)
    else:
        ans = max(ans, x*max_kr_6)
        nekr = max(x, nekr)
print(ans)
```

Ура, такой прототип решать научились считайте! Как понять, сколько максимумов нужно? Сколько делителей у числа, на которое делим, кроме 1, значит столько и максимумов. А если найти мин. произведение? Все тоже самое, только ищем минимумы и мин. сумму. Вот и все.

Задача 2

Дано целое число n, затем последовательность из n натуральных чисел. Требуется найти количество пар, произведение которых кратно 6.

Пример входных данных:

5
2
10
5
17
3

Выходные данные для приведённого выше примера: 2

Рассуждения. Решение статикой.

По задачам выше уже знаем что искать. Только вместо максимумов нужно подсчитывать на каждом шаге количество элементов, имеющих свою определенную кратность при делении на 6, 3, 2 и некр.

После чего идет самое интересное — как подсчитать ответ?

Сначала разберемся, как подсчитать количество пар, в которых оба числа являются кратными 6? Можно воспринимать это как одно множество или один граф, внутри которого находятся вершины и объединяются ребрами. Как выбрать любую вершину из k? k способами. А как ей выбрать вершину в пару? k*(k-1). Хорошо, однако если нарисовать граф, то вы заметите, что одна пара может иметь два вида: (a, b) и (b, a). Поскольку пары эквиваленты, то требуется повторы исключить, а для этого нужно k*(k-1) поделить на 2! = 2.

Отлично, а как подсчитать оставшиеся произведения? Поскольку, к числам кратным 6, к примеру ставятся в пару числа, кратные исключительно 3, то они обладают разным свойством, по которому мы сравнивали ранее. А значит лежат в разных множествах (графах). Тогда достаточно лишь перемножить количество чисел кр_6 на количество чисел кр_3. И так со всеми оставшимися связями.

Решение.

```
n = int(input())
kr_6 = 0
kr_3 = 0
kr_2 = 0
```

```
nekr = 0
ans = 0
for i in range(n):
    x = int(input())
    if x % 6 == 0:
        kr_6 += 1
    elif x % 3 == 0:
        kr_3 += 1
    elif x % 2 == 0:
        kr_2 += 1
    else:
        nekr += 1
ans += kr_6*(kr_6-1)//2 \
       + kr_6*kr_3 + kr_6*kr_2 + kr_6*nekr \
       + kr_3*kr_2
print(ans)
```

Рассуждения. Решение динамикой.

Считываем x. Считаем на каждом шаге количество пар, в зависимости от кратности x, а затем обновляем количество элементов, обладающих определенной кратностью.

Решение.

```
n = int(input())
kr_6, kr_3,kr_2, nekr, ans = 0, 0, 0, 0, 0
for i in range(n):
    x = int(input())
    if x % 6 == 0:
        ans += kr_6+kr_3+kr_2+nekr
        kr_6 += 1
    elif x % 3 == 0:
        ans += kr_6 + kr_2
        kr_3 += 1
    elif x % 2 == 0:
```

```
            ans += kr_6 + kr_3
            kr_2 += 1
        else:
            ans += kr_6
            nekr += 1
print(ans)
```

Задача 3

Дано целое число n, затем последовательность из n натуральных чисел. Требуется найти максимальную сумму, среди пар чисел, которая будет кратна 10.

Пример входных данных:

5
2
10
5
17
3

Выходные данные для приведённого выше примера: 20

Рассуждения. Решение статикой.

Есть три случая:

1) В пару ставим макс. число, кратное 10 и пред. макс. кр. 10

2) Пункт 1, но для чисел с остатком 5, при делении на 10

3) Остальные случаи, когда в пару ставятся макс. числа с остатком при делении на 10 с числом, который обладает взаимодополняющим остатком для первого числа при делении на 10.

Заведем два массива — массив максимумов (a) и пред. максимумов(b). Заполним, а потом посчитаем ответ.

Решение.

```
n = int(input())
a = [0]*10
b = [0]*10
for i in range(n):
    x = int(input())
    if x % 10 == 0:
       if x > a[0]:
           b[0] = a[0]
           a[0] = x
       else:
           b[0] = max(x, b[0])
```

```
    elif x % 10 == 5:
        if x > a[5]:
            b[5] = a[5]
            a[5] = x
        else:
            b[5] = max(x, b[5])
    else:
        a[x%10] = max(a[x%10], x)
ans = max(a[0]+b[0], a[5]+b[5])
for i in range(1, 5):
    ans = max(ans, a[i]+a[10-i])
print(ans)
```

Рассуждения. Решение динамикой.

Считываем x. Считаем на каждом шаге макс. сумму с максимальным взаимодополняющим числом по остатку, обновляем массив.

Решение.

```
n = int(input())
a = [0] * 10
ans = 0
for i in range(n):
    x = int(input())
    ans = max(ans, x+a[(10-x)%10])
    a[x%10] = max(a[x%10], x)
print(ans) #Вот и все решение!
```

Задача 4

Дано целое число n, затем последовательность из n натуральных чисел. Требуется найти количество пар чисел, сумма которых будет кратна 91.

Пример входных данных:

5
2
89
5
17
3

Выходные данные для приведённого выше примера: 1

Рассуждения. Решение статикой.

Также есть два случая (не 3, потому что делим на нечетное). Вместо хранения максимумов храним в одном массиве количества элементов, обладающих определенной кратностью. Считаем ответ.

Решение.

```
n = int(input())
a = [0]*91
for i in range(n):
    a[int(input()) % 91] += 1
ans = a[0]*(a[0]-1)//2
for i in range(1, 46):
    ans += a[i]*a[91-i]
print(ans)
```

Рассуждения. Решение динамикой.

Считываем x. Считаем на каждом количество пар, обновляем массив.

Решение.

```
n = int(input())
ans = 0
a = [0]*91
for i in range(n):
    x = int(input())
```

```
    ans += a[(91-x)%91]
    a[x%91] += 1
print(ans)
```

Перед решением следующих задач просто подмечу вот что — во-первых, решения статикой элегантны своей взаимосвязью с комбинаторикой и всеми знаниями, которые так или иначе вы приобрели за курс информатики. Однако, чаще всего куда проще пользоваться динамикой. Во-вторых, следующие задачи будут содержать лишь комбинированные условия предыдущих. Порядок решения следующий — вычленяете самые простые условия, затем условия сложнее (как правило самое просто это произведение, еще проще, когда одно из чисел должно быть больше чего-то в паре, затем сумма). Создаете массивы, олицетворяющие эти условия и работаете по ранее протоптанным дорожкам.

Задача 5

Дано целое число n, затем последовательность из n натуральных чисел. Требуется найти количество пар чисел, сумма которых будет кратна 8, произведение кратно 7.

Пример входных данных:

5
7
25
5
17
3

Выходные данные для приведённого выше примера: 2

Решение.

```
#Статика
n = int(input())
kr_7 = [0]*8
nekr = [0]*8
for i in range(n):
    x = int(input())
    if x % 7 == 0:
        kr_7[x%8] += 1
    else:
        nekr[x%8] += 1
ans = kr_7[0] * (kr_7[0]-1) // 2 +\
```

```
      kr_7[4] * (kr_7[4]-1) // 2 + \
      kr_7[0] * nekr[0] + \
      kr_7[4] * nekr[4]
for i in range(1, 8):
    ans += (kr_7[i] * kr_7[8-i])//2 + kr_7[i] * nekr[8-i]
print(ans)
```

Решение.

```
#Динамика
n = int(input())
kr_7 = [0]*8
nekr = [0]*8
ans = 0
for i in range(n):
    x = int(input())
    if x % 7 == 0:
        ans += kr_7[(8-x%8)%8] + nekr[(8-x%8)%8]
        kr_7[x%8] += 1
    else:
        ans += kr_7[(8 - x % 8) % 8]
        nekr[x % 8] += 1
print(ans)
```

Теперь посмотрим на задачи, в которых добавлено одно маленькое, но немаловажное условие — пары находятся на расстоянии не меньше, чем какое-то (то есть разница в индексах по модулю больше или равна чему-то).

База-прототип:
Дано целое число n затем последовательность из n натуральных чисел.
Требуется найти
количество/макс./мин. сумму пар,
чья сумма/разность/произведение кратна(о) чему-то,
а расстояние между элементами пары не меньше k.

Решим следующую задачу неэффективно:

Дано целое число n затем последовательность из n натуральных чисел. Требуется найти количество пар, чья сумма кратна 8, а расстояние между элементами пары не меньше 3.

Пример входных данных:

7
17
15
1
5
123
1
38

Для таких входных данных значением искомое суммы будет число 1

Решение.

```
n = int(input())
ans = 0
a = []
for i in range(n):
    a += [int(input())]
for i in range(n):
    for j in range(i+3, n):
        if (a[i]+a[j]) % 8 == 0:
```

```
            ans += 1
print(ans)
```

Что поменялось в данном решении в сравнении с предыдущими такими же, но без расстояния? Всего-лишь разница индексов i и j при рассмотрении вложенного цикла.

Хорошо, но появляется вопрос: как решить данные задачи эффективно?

Введем следующие определения: свалка, массив-очередь, очередь (последние два определения можно объединить в просто очередь). Итак, на примере прошлой задачи разница в индексах рассматриваемых элементов должна быть 3 и более. То есть, числа с индексами, чья разница меньше, рассмотрены в качестве пары быть не могут.

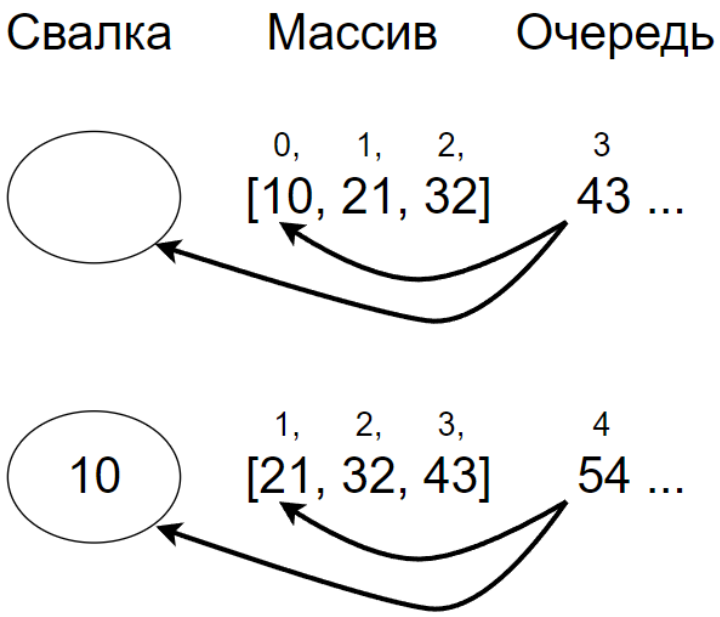
Хорошо, тогда почему бы не объединить все числа, которые не могут составлять между собой пару в один массив? И сделать так, чтобы вне массива был первый элемент, который мог бы взаимодействовать с нулевым. (необязательно так, но лично я предпочитаю такую реализацию). Схематично предлагается сделать следующее:

Пусть дано n = 6 чисел и следующие числа:

10 21 32 43 54 65

Тогда поместим в массив-очередь числа

10 21 32 (разница крайних по индексам равна 2)



Тогда, предлагается число 10 отправить на свалку (т.к. число 43 может с ним провзаимодействовать, а также с числами на свалке, т.к. они были раньше 10). После взаимодействия 43 со всеми подходящими числами сдвинем все числа последовательности на 1 вправо и уже будем рассматривать число 54 и всевозможные подходящие данному числу пары (число 21 и число 10, а также другие числа со свалки). Будем так делать, пока это возможно.

Итак, рассмотрим эффективное решение предыдущей задачи:

Дано целое число n затем последовательность из n натуральных чисел. Требуется найти количество пар, чья сумма кратна 8, а расстояние между элементами пары не меньше 3.

Рассуждения

Сначала разберемся с очередью. Создадим массив и заполним его первыми тремя элементами. После чего организуем продолжения цикла for где будем считывать новое число x. Перед тем, чтобы x взаимодействовал со свалкой отправим нулевой элемент очереди туда. Отправить элемент на свалку означает, что мы извлечем из элемента самые полезные свойства для решения нашей задачи. Затем x поставим все подходящие пары со свалки, после чего сдвинем очередь на один влево.

По условию задачи требуется найти количество пар, чья сумма будет кратна 8. Требуется на свалке создать массив из 8 счетчиков, каждый из которых будет хранить количество чисел, имеющих определенный остаток при делении на 8. Считывая x будем ставить ему в пару элементы, с взаимодополняющим остатком при делении на 8.

К реализации!

Решение.

```
n = int(input())
line = []
a = [0]*8
ans = 0
for i in range(3):
    line.append(int(input()))
for i in range(3, n):
    #1й блок: Отправляем на свалку
    #Самый крайний элемент, с которым x может взаимодействовать
```

```
    a[line[0] % 8] += 1
    #2й блок: Взаимодействие всех подходящих с x
    x = int(input())
    ans += a[(8 - x%8)%8]
    #3й блок: Сдвиг очереди
    for j in range(3-1):
        line[j] = line[j+1]
    line[2] = x
print(ans)
```

Задача 6

Дано целое число n затем последовательность из n натуральных чисел. Требуется найти количество пар чисел, чье произведение будет кратно 22, при этом расстояние между элементами пары хотя бы 7, а один из элементов пары не меньше 55.

Пример входных данных:

8
22
15
1
5
123
1
1
55

Для таких входных данных значением искомое суммы будет число 1

Рассуждения.

Первый индекс:

0 - Числа, меньшие 55, 1 - числа большие 55 (55 включительно)

Второй индекс:

0 - числа кратные 22, 1 - числа кратные только 11,

2 - числа кратные только 2, 3 - некратные ничему

Решение.

```
kr = [[0 for i in range(4)] for j in range(2)]

n = int(input())
ans = 0
line = []
for i in range(7):
    line.append(int(input()))
for i in range(7, n):
    tmp = 3 - ((line[0] % 11==0)*2 + (line[0] % 2==0))
    kr[line[0] >= 55][tmp] += 1
```

```
    x = int(input())
    if x % 22 == 0:
        for j in range(4):
            ans += kr[0][j]*(x >= 55) + kr[1][j]
    elif x % 11 == 0:
        for j in range(0, 4, 2):
            ans += kr[0][j]*(x >= 55) + kr[1][j]
    elif x % 2 == 0:
        for j in range(0, 2):
            ans += kr[0][j]*(x >= 55) + kr[1][j]
    else:
        ans += kr[0][0] * (x >= 55) + kr[1][0]
    for j in range(7-1):
        line[j] = line[j+1]
    line[6] = x
print(ans)
```

Разберем задачи следующего типа:

```
База-прототип:
Дано число n затем n строк,
каждой по паре натуральных различных чисел.
Из каждой пары берется одно число так,
чтобы итоговая сумма всех таких чисел
была кратна/некратна *целому числу* и была бы макс/мин.
```

В целом, задачи по сложности и по "идейности"нужно разделять на два блока:

1) Задачи, про сумму некратную чему-либо
2) Задачи, про сумму кратную чему-либо

Задачи первого типа:

Задача 7

Дано число n затем n строк, в каждой по паре натуральных различных чисел. Из каждой пары берется одно число так, чтобы итоговая сумма всех таких чисел была некратна 10 и была бы:

а) Максимальна

б) Минимальна

Пример входных данных:

3

10 20

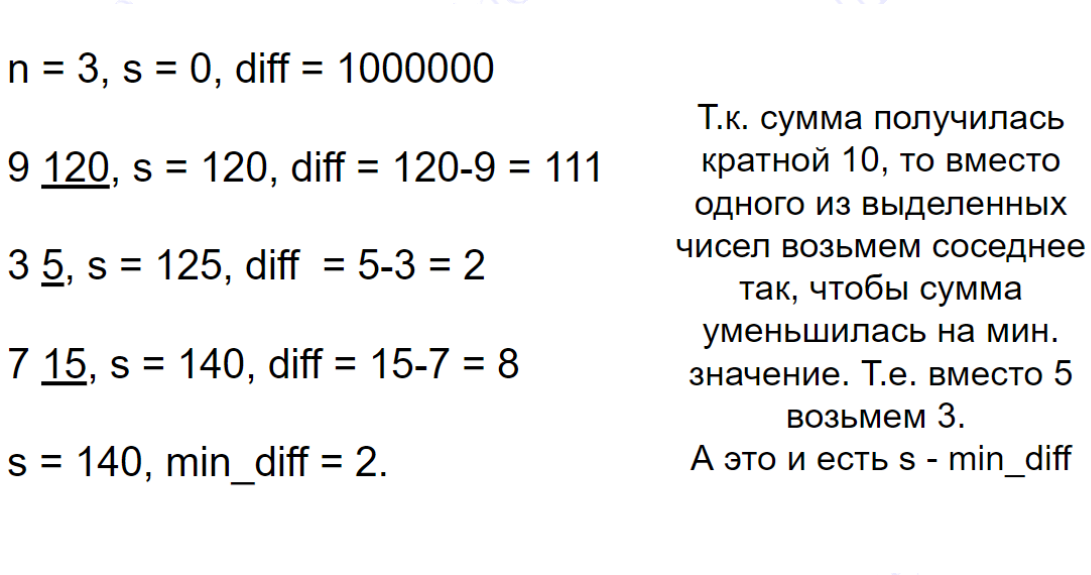
21 11

1 1

Для таких входных данных значением искомое суммы будет число 42 и 22

Рассуждения.

Посмотрим на скриншот ниже. Входные данные примера описаны слева — число 3 и 3 строки с парами. Решим задачу статически (в данном случае так проще). Воспользуемся "жадным"алгоритмом: сначала посчитаем максимальную сумму, а затем "уменьшим"ее, если она не подойдет по условию задачи. То есть, чтобы получить максимальную сумму, будем на каждом шаге прибавлять макс. элемент из каждой пары.



Но как решить проблему, когда у нас может получиться сумма кратная 10? Будем на каждом шаге вычислять разность элементов пары, некратную 10 и находить минимальную. Идея в том, что если по итогу работы жадного алгоритма получим неподходящую сумму, то мы из максимальной суммы вычтем минимальную разность, некратную 10 среди всех элементов каждой строки. Почему? Это равносильно тому, что вместо одного элемента из одной из строк мы возьмем соседний, причем итоговая сумма уменьшится на минимальное значение (гарантируется тем, что разность минимальная).

Решение. Пункт а)

```
n = int(input())
ans = 0
diff = 10000000000
for i in range(n):
    a, b = map(int, input().split())
    #Ниже альтернативный способ считать два числа:
    #a = [*что делаем,* *откуда делаем,* *при каком условии*]
    #a = [int(i) for i in input().split()]
    ans += max(a,b)
    if abs(a-b) % 10 != 0:
        diff = min(diff, abs(a-b))
print(ans - diff * (ans % 10 == 0))
```

Решение пункта б) выглядит следующим образом:

Решение. б)

```
#Мин. сумма
n = int(input())
ans = 0
diff = 10000000000
for i in range(n):
    a, b = map(int, input().split())
    ans += min(a,b)
    if abs(a-b) % 10 != 0:
        diff = min(diff, abs(a-b))
print(ans + diff * (ans % 10 == 0))
```

Разберем задачи второго типа:

Для начала вспомним два важных свойства остатков из арифметики — когда сумма и разность двух чисел a и b кратна любому целому числу? Для конкретики разберем кратность суммы и разности числу 3:

Пусть число a имеет остаток p при делении на 3. Тогда, чтобы a + b было кратно 3, число b должно иметь взаимодополняющий остаток для числа a при делении на 3, который будет равен: b % 3 == (3 - p) % 3. То есть, сумма остатков должна давать 0 в троичной системе счисления, т.к. остатки для числа 3 составляют числа: 0, 1, 2.

Теперь насчет разности. Здесь проще — чтобы разность была кратна некоторому числу, то достаточно лишь чтобы остатки двух чисел были одинаковыми при делении на число.

На основе вышеописанных свойств предлагается самостоятельно разобраться в следующей задаче:

Задача 8

Дано число n затем n строк, в каждой по паре натуральных различных чисел. Из каждой пары берется одно число так, чтобы итоговая сумма всех таких чисел была кратна 2 и максимальна.

Пример входных данных:

3
10 20
21 11
1 1

Для таких входных данных значением искомое суммы будет число 42

Решение.

```
n = int(input())
ans = 0
diff = 10000000000
for i in range(n):
    a = [int(i) for i in input().split()]
    ans += max(a)
    if abs(a[0]-a[1]) % 2 == 1:
        diff = min(diff, abs(a[0]-a[1]))
```

```
print(ans - diff * (ans % 2 == 1))
```

По сути, решение идентично подобной же задаче на некратность.

А что делать, если скажем нужно найти минимальную сумму, кратную 3? Какие есть подводные камни? Именно это понимание и приведет нас в дальнейшем к идее универсального алгоритма.

Задача 9

Дано число n затем n строк, в каждой по паре натуральных различных чисел. Из каждой пары берется одно число так, чтобы итоговая сумма всех таких чисел была кратна 3 и минимальна.

Пример входных данных:

3

13 20

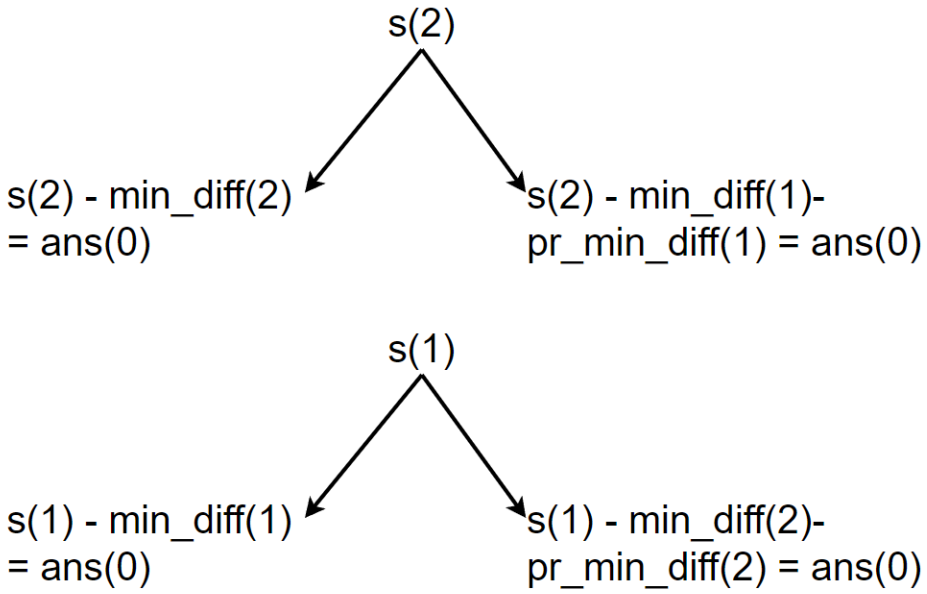
21 11

1 1

Для таких входных данных значением искомое суммы будет число 25

Рассуждения.

Предположим, что после работы жадного алгоритма мы получили сумму, с остатком 2 при делении на 3. Итак, казалось бы все просто — можно вычесть мин. дифф с остатком 2. Но не тут то было — так то оно так, однако этого недостаточно. Ведь еще можно вычесть из подобной суммы два мин. диффа с остатком 1. И ведь действительно: S(2) - diff(2) = (0) или S(2) - diff(1) - post_diff(1) = (0). (вычитаем если находим макс. сумму). Тоже самое и для суммы, с остатком 1. Собственно, нужно хранить четыре разности: первые две с остатком 1, вторые с остатком 2.



Решение.

```
n = int(input())
ans = 0
#инд 0 и 1 отвечают за мин дифы с ост. 1
#2 и 3 за мин дифы с ост. 2
diff = [1000000]*4
for i in range(n):
    a, b = map(int, input().split())
    ans += min(a,b)
    tmp = abs(a-b)
    if tmp % 3 == 1:
        if tmp < diff[0]:
            diff[1], diff[0] = diff[0], tmp
        else:
            diff[1] = min(diff[1], tmp)
    if tmp % 3 == 2:
        if tmp < diff[2]:
            diff[3], diff[2] = diff[2], tmp
        else:
            diff[3] = min(diff[3], tmp)
if ans % 3 == 0:
    print(ans)
elif ans % 3 == 1:
    print(ans + min(diff[2], diff[0]+diff[1]))
else:
    print(ans + min(diff[0], diff[2] + diff[3]))
```

А еще вы могли заметить, что решение заметно выросло как в плане длины кода, так и в смысловом — теперь вовсе недостаточно иметь один дифф, а потребовалось аж 4 и это просто для кратности 3. Соответственно, чем больше число, которому сумма должна быть кратна, тем больше диффов и становится ясно, что так решать нецелесообразно.

Рассмотрим эффективный алгоритм для решения всех подобных задач, где фигурирует сумма и кратность чему-либо. Собственно, главная идея: давайте будем считать ответ сразу, то есть решать задачу ДИНАМИЧЕСКИ. Вопрос: а как реализовать? Предлагается следующая возможность — почему бы нам не считать всевозможные суммы на каждом шаге, при этом запоминать макс/мин суммы определенного остатка при делении на число, чтобы в дальнейшем возможно найти подходящую сумму? Давайте на примере:

Задача 10

Дано число n затем n строк, в каждой по паре натуральных различных чисел. Из каждой пары берется одно число так, чтобы итоговая сумма всех таких чисел была кратна 10 и максимальна.

Пример входных данных:

3

10 20

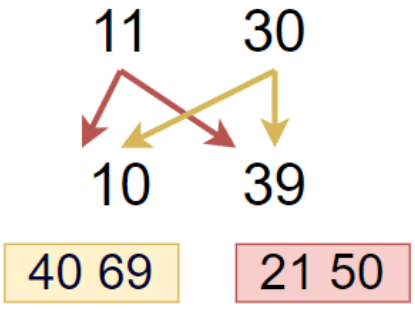
21 11

1 9

Для таких входных данных значением искомое суммы будет число 50

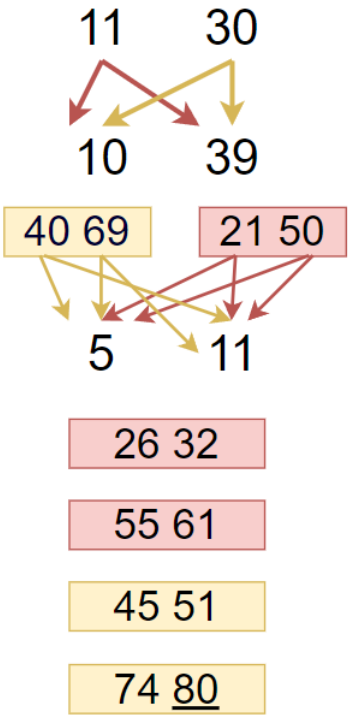
Рассуждения.

Начнем с того, что у нас есть две строки, и соответственно разберем всевозможные суммы, то есть сначала считаем первые два числа. Затем считаем новое число из новой строки и сложим его с каждым предыдущим, затем сделаем точно также со вторым числом и потенциально получим 4 суммы, каждая из которых имеет свой остаток при делении на 10.



Хорошо, давайте теперь добавим третью строку и посчитаем все новые суммы на

новом шаге, опираясь на красные и желтые суммы (следовательно новые красные суммы это старые красные + одно число из третьей пары, желтые аналогично).



После суммирования можно заметить, что максимальной суммой, кратной 10 будет 80. А как мы ее получили? Сначала мы взяли число 30, затем 39, а уже потом 11. Вроде бы понятно что сделали. А как это систематизировать? Предлагается завести массив из 10 элементов — каждый отвечает за макс. сумму с определенным остатком при дел. на 10.

Сначала заполним его мусором. Затем на первой итерации в качестве новых сумм просто поместим первые два считанных числа — каждое в свою макс. ячейку с опр. остатком. Затем происходит следующее: к числам 11 и 30 поочередно прибавим сначала 10, затем 39 из второй строки. Получим новые максимальные суммы, имеющих опр. ост. при дел. на 10. Перезапишем их в массив.

a = [-100000, -100000, -100000, -100000, -100000, -100000, -100000, -100000, -100000, -100000]

i = 0: [30, 11, -100000, -100000, -100000, -100000, -100000, -100000, -100000, -100000]

i = 1: [50, 21, -100000, -100000, -100000, -100000, -100000, -100000, -100000, 69]

i = 2: [80, 61, -100000, 32, -100000, 74, 55, 26, -100000, -100000]

Теперь точно такую же операцию проводим и для новых сумм — поочередно прибавляя к каждой числа из новой строки. Вот и вся идея. Таким образом, мы подсчитываем промежуточный ответ на каждом шаге цикла for, при этом подсчитывая всевозможные суммы, включающие (!) по одному элементу с каждой строки. Перейдем к реализации.

Решение.

```
n = int(input())
ans = [0]*10 #Массив всех макс. сумм, которые имеют опр. ост. при дел. на 10
for i in range(n):
    a, b = map(int, input().split())
    #Создаем копию массива, чтобы не повредить данные изначального
    #а также адекватно посчитать новые подх. суммы.
    ans_new = [-1000000000]*10

    for j in range(10): #Блок подсчета всех сумм, путем прибавления числа a
        ost = (ans[j] + a) % 10
        #Сравниваем с ans_new, потому что изначально он забит мусором
        #И если мы нашли сумму, то она уже будет больше.
        #Если сравнивать с ans, то есть риск получить промежуточную сумму
        #неадекватного значения
        if ans[j] + a > ans_new[ost]:
            ans_new[ost] = ans[j] + a

    for j in range(10): #Блок подсчета всех сумм, путем прибавления числа b
        ost = (ans[j] + b) % 10
```

```
        if ans[j] + b > ans_new[ost]:
            ans_new[ost] = ans[j] + b

    ans = ans_new #Обнов-ние всех макс. сумм, при каком-то ост. при дел. на 10
print(ans[0])
```

Для закрепления разберем еще одну задачу, относительно упростив код:

Задача 11

Дано число n затем n строк, в каждой по тройке натуральных различных чисел. Из каждой тройки берется одно число так,чтобы итоговая сумма всех таких чисел была кратна 8 и была бы минимальна.

Пример входных данных:

3

8 20

21 11

5 13

Для таких входных данных значением искомое суммы будет число 24

Решение.

```
ans = [0]*8
n = int(input())
for i in range(n):
    a = [int(j) for j in input().split()]
    ans_new = [100000000]*8
    for k in range(3):
        for j in range(8):
            ost = (a[k] + ans[j]) % 8
            ans_new[ost] = min(ans_new[ost], a[k] + ans[j])
    ans = ans_new
print(ans[0])
```

Рассмотрим также следующий тип задач:

Задача 12

Дано число n затем n строк, в каждой по тройке натуральных различных чисел. Найдите три суммы таких, чтобы минимальная была нечетна, а максимальная четна. Каждое число обязательно прибавляется в одну из сумм и только одно. Выведите макс. сумму, удовлетворяющую вышеописанным условиям. Подразумевается, что подобную сумму получить можно.

Решение.

```
n = int(input())
s = [0]*3 #0 - мин, 1 - ср., 2 - макс.
#10 20 30
#diff_0 = средний - мин.
#diff_1 = средний - макс.
#diff_2 = макс. - мин.
diff = [10000000]*3

for i in range(n):
    a = sorted(list(map(int, input().split())))
    for j in range(3):
        s[j] += a[j]
    if abs(a[0]-a[1]) % 2 == 1:
        diff[0] = min(diff[0], abs(a[0]-a[1]))
    if abs(a[2]-a[1]) % 2 == 1:
        diff[1] = min(diff[0], abs(a[2]-a[1]))
    if abs(a[0]-a[2]) % 2 == 1:
        diff[2] = min(diff[2], abs(a[0]-a[2]))

if s[0] % 2 != 0 and s[2] % 2 == 0:
    print(s[2])

elif s[0] % 2 == 0 and s[2] % 2 == 0:
```

```
    if diff[0] != 10000000:
        print(s[2])
    else:
        print(s[2]-diff[2]-diff[1])

elif s[0] % 2 != 0 and s[2] % 2 != 0:
    if diff[1] != 10000000 and diff[0] != 10000000 and diff[2] != 10000000:
        print(s[2] - min(diff[1], diff[2]))
    elif diff[1] != 10000000:
        print(s[2] - diff[1])
    elif diff[0] != 10000000 and diff[2] != 10000000:
        print(s[2] - diff[2])

else:
    if diff[1] != 10000000 and diff[0] != 10000000 and diff[2] != 10000000:
        print(s[2] - min(diff[1], diff[2]))
     elif diff[1] != 10000000:
        print(s[2] - diff[1])
    elif diff[0] != 10000000 and diff[2] != 10000000:
        print(s[2] - diff[2])
```

Также разберем задачи на префиксные суммы:

```
База-прототип: Дана последовательность натуральных чисел.
Необходимо найти максимально возможную сумму
её непрерывной подпоследовательности,
кратную чему-либо.
```

На конкретной задаче:

Задача 13

Дана последовательность натуральных чисел. Необходимо найти максимально возможную сумму её непрерывной подпоследовательности, кратную 10. Выведите наибольшую длину последовательности, при которой сумма будет кратна 10 и максимальна.

Первая строка содержит натуральное число N — общее количество чисел в наборе. Каждая из следующих N строк содержит одно число.

Рассуждения.

Непрерывные суммы могут начинаться как с 0-го элемента, так и с 1-го, 2-го, . . . , (n-1)-го. Идея алгоритма будет следующей: на каждом шаге будем прибавлять новый элемент к сумме всех предыдущих. Из данной суммы будем вычитать мин. непрерывную сумму, имеющую такой же остаток при делении на 10, что и рассматриваемая нами сумма. Если такой мин. суммы нет, то нынешняя сумма и является такой минимальной.

Пример:

7
23
13
4
11
6
19
8

Ответом для примера будет: 4

Решение.

```
n = int(input())
s = [1000000]*10
s[0] = 0
summ = 0
ans = 0
ind = [-1]*10
dlina = 0

for i in range(n):
    summ += int(input())

    if summ - s[summ % 10] > ans:
        ans = summ - s[summ % 10]
        dlina = i - ind[summ % 10]

    elif summ - s[summ % 10] == ans:
        dlina = max(dlina, i - ind[summ % 10])

    if summ < s[summ % 10]:
        s[summ % 10] = summ
        ind[summ % 10] = i

print(dlina)
```

Рассмотрим задачу, в которой нужно придумать, как именно использовать префиксные суммы:

Задача 14

Дана последовательность натуральных чисел. Рассматриваются все её непрерывные подпоследовательности, в которых количество простых чисел кратно K = 3. Найдите наибольшую сумму такой подпоследовательности.

Пример:

7
23
13
4
11
6
19
8

Ответом для примера будет: 61

Рассмотрим пример из условия.

В этом наборе можно выбрать две непрерывные последовательности, содержащие по 3 простых числа (23+13+4+11+6=57) и (13+4+11+6+19+8=61).

Решение.

```
def prime(x):
    for i in range(2, int(x**0.5)+1):
        if x % i == 0:
            return False
    return True

n = int(input())
data = [] #Все числа из файла
num = [0] * n #Индекс этого массива равен индексу data и показывает, явл. ли эл-ты из data
              #Простыми числами. Если да - в массиве будет стоять 1.
for i in range(n):
    data += [int(input())]
    if prime(data[i]):
        num[i] += 1

summa = 0 #Сумма всех единиц массива num
s = [1000000]*3 #Мин. частичные суммы, имеющие опр. остаток при делении на 3.
                #Почему на 3? Потому кол-во простых чисел должно быть кр. 3.
s[0] = 0
ind = [-1]*3 #На каком индексе была получена мин. частичная сумма
ind[0] = 0
dlina = 0 #Длина последовательности, в которой сумма наибольшая
kr = 0 #Число, отражающее макс. кол-во простых чисел в рассматриваемой подпослед.,
       #которое будет кр. 3
ans = 0

for i in range(n):
    summa += num[i] #Подсчет i-ой частичной суммы из массива 1 и 0
    if summa - s[summa % 3] > 0:
        ans = max(ans, sum( data[ind[summa % 3]:i+1] ))
    if summa < s[summa % 3]: #Обновление мин. частичных сумм
        s[summa % 3] = summa #обновляем частичную сумму
        ind[summa % 3] = i   #Индекс последнего элемента этой суммы

print(ans)
```

Также рассмотрим задачу с досрока ЕГЭ 2022 года и ее решение:

Задача 15

В городе M расположена кольцевая автодорога длиной в N километров с движением в обе стороны. На каждом километре автодороги расположены пункты приема мусора определенной вместимости. В пределах кольцевой дороги в одном из пунктов сборки мусора собираются поставить мусороперерабатывающий завод таким образом, чтобы стоимость доставки мусора была минимальной. Стоимость доставки мусора вычисляется, как вместимость пункта сбора умноженная на расстояние от пункта сбора мусора до мусороперерабатывающего завода. Если мусороперерабатывающий завод находится рядом с пунктом сбора расстояние считается нулевым. Контейнеры нумеруются с 1 до N. Рядом с каким пунктом сбора мусора нужно поставить мусороперерабатывающий завод?

Описание входных данных:

Первое число N — количество контейнеров для мусора. Последующие N чисел — количество килограмм мусора, которое производится на точке.

Описание выходных данных:

Одно число — номер контейнера для мусора рядом с которым стоит расположить перерабатывающий завод.

Решение.

```
file = open('test.txt', 'rt')
n = int(file.readline())
a = [int(file.readline()) for i in range(n)]
s = [sum(a[0:n // 2])]
for i in range(1, n):
    s.append(s[i - 1] - a[i - 1] + a[(i - 1 + n // 2) % n])

summa = sum(a)
p = []
for i in range(n):
    p.append(summa - s[i])

price = 0
```

```
for i in range(n // 2):
    price += i * a[i]
for i in range(n // 2, n):
    price += (n - i) * a[i]

minim = price
min_k = -1
for i in range(1, n):
    price -= s[i]
    price += p[i]
    if price < minim:
        minim = price
        min_k = i + 1
print(min_k)
```