Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
ордена Трудового Красного Знамени
федеральное государственное
бюджетное образовательное учреждение
высшего образования

Московский Технический Университет Связи и Информатики

Кафедра
"Математическая кибернетика и информационные технологии"

Курсовая работа
по дисциплине
"Структуры и алгоритмы обработки данных"

Выполнил студент
группы БВТ2005
Макбари Д. А.

Москва 2022

Содержание

Задача 1

Задан массив $a1,a2,\ldots,an$, состоящий из n положительных целых чисел.
Изначально вы находитесь в позиции 1, и ваше количество очков равно $a1$.
Можно совершать два типа шагов:
шаг вправо — перейти из текущей позиции x в $x+1$ и получить $ax+1$ очков. Этот шаг можно делать только если $x<n$.
шаг влево — перейти из текущей позиции x в $x-1$ и получить $ax-1$ очков. Этот шаг можно делать только если $x>1$. Также нельзя совершать два или более шагов влево подряд.
Вам требуется совершить ровно k шагов. Не более z из них могут быть шагами влево.
Какое наибольшее количество очков можно получить?
Входные данные
В первой строке записано одно целое число t ($1\leq t\leq 104$) — количество наборов входных данных.
В первой строке каждого набора входных данных записаны три целых числа n,k и z ($2\leq n\leq 105$, $1\leq k\leq n-1$, $0\leq z\leq min(5,k)$) — количество элементов в массиве, суммарное количество шагов, которое вы должны сделать, и максимальное количество шагов влево, которое вы можете сделать.
Во второй строке каждого набора входных данных записаны n целых чисел $a1,a2,\ldots,an$ ($1\leq ai\leq 104$) — данный массив.
Сумма n по всем наборам входных данных не превосходит $3\cdot 105$.
Выходные данные
Выведите t целых чисел — для каждого набора входных данных выведите наибольшее количество очков, которое можно получить, если требуется сделать ровно k шагов, не более z из них могут быть шагами влево и не должно быть двух шагов влево подряд.

Решение:

```
def task1(arr, k, z):
    maxsum = 0
    for lmoves in range(z + 1):
        rmoves = k - 2 * lmoves
        for lrpos in range(rmoves + 1):
            if lrpos + 1 < len(arr) - 1:
                tempsum = lmoves * (arr[lrpos] + arr[lrpos + 1]) +
                sum(arr[:rmoves + 1])
                maxsum = tempsum if tempsum > maxsum else maxsum
    return maxsum
```

```
print("Ввод")
t = int(input("t = "))
results = []
for i in range(t):
    [k, z] = input("k, z = ").split()
    arr = input("arr = ").split()
    arr = [int(arr[i]) for i in range(len(arr))]
    results.append(task1(arr, int(k), int(z)))
print("Вывод:")
for i in range(t):
    print(results[i])
```

Задача 2

Вам дан массив a из n целых чисел.
Вы хотите сделать все элементы a равными нулю, применив следующую операцию ровно три раза:
Выберите отрезок, к каждому числу на этом отрезке добавьте число кратное len, где len это длина этого отрезка (добавленные числа могут быть разными).
Можно доказать, что таким образом всегда можно превратить все элементы a в нули.
Входные данные
В первой строке записано одно целое число n ($1 \le n \le 100000$): количество элементов массиве.
Во второй строке записаны n элементов массива a, разделенные пробелами: $a1, a2, \ldots, an$ ($-109 \le ai \le 109$).
Выходные данные
Выведите шесть строк, описывающих три операции.
Для каждой операции, выведите две строки:
В первой строке выведите два числа l, r ($1 \le l \le r \le n$): границы выбранного отрезка. Во второй строке выведите $r-l+1$ целых чисел $bl, bl+1, \ldots, br$ ($-1018 \le bi \le 1018$): числа, которые нужно прибавить к $al, al+1, \ldots, ar$, соответственно; bi должно делиться на $r-l+1$.

Решение:

```
def task2(arr):
    n = len(arr)

    if n == 1:
```

```
            print(1, 1)
            print(-arr[0])
            print(1, 1)
            print(0)
            print(1, 1)
            print(0)
            return

        # step 1
        print(1, 1)
        print(-arr[0])

        # step 2
        print(1, n)
        print(0, end = ' ')
        for i in range(1, n):
            print(-n * arr[i], end = ' ')
        print()

        # step 3
        print(2, n)
        for i in range(1, n):
            print((n - 1) * arr[i], end = ' ')
        print()

print("Ввод:")
arr = input("arr = ").split()
arr = [int(arr[i]) for i in range(len(arr))]
print("Вывод:")
task2(arr)
```

Задача 3

Вам задан массив a, состоящий из n целых чисел. Индексы элементов массива начинаются с нуля (то есть первый элемент — это $a0$, второй — $a1$, и так далее).

Вы можете развернуть не более одного подмассива (последовательного отрезка) этого массива. Напомним, что подмассив a с границами l и r равен $a[l;r]=al,al+1,\ldots,ar$.

Ваша задача — развернуть такой подмассив, чтобы сумма элементов на чет-

ных позициях получившегося массива была максимально возможной (то есть сумма элементов $a0,a2,\ldots,a2k$ для целого числа $k=\lfloor n-12 \rfloor$ должна быть максимально возможной).
Вам необходимо ответить на t независимых наборов тестовых данных.
Входные данные Первая строка входных данных содержит одно целое число t ($1 \leq t \leq 2 \cdot 104$) — количество наборов тестовых данных. Затем следуют t наборов тестовых данных.
Первая строка набора содержит одно целое число n ($1 \leq n \leq 2 \cdot 105$) — длину a.
Вторая строка набора содержит n целых чисел $a0,a1,\ldots,an-1$ ($1 \leq ai \leq 109$), где ai — это i-й элемент a.
Гарантируется, что сумма n не превосходит $2 \cdot 105$ ($\sum n \leq 2 \cdot 105$).
Выходные данные
Для каждого набора тестовых данных выведите ответ на отдельной строке — максимально возможная сумма элементов на четных позициях после разворота не более одного подмассива (последовательного отрезка) a.

Решение:

```
from sys import maxsize

def maxSubArray(a):
    size = len(a)
    max_so_far = -maxsize - 1
    max_ending_here = 0
    start = 0
    end = 0
    s = 0
    for i in range(0,size):
        max_ending_here += a[i]
        if max_so_far < max_ending_here:
            max_so_far = max_ending_here
            start = s
            end = i
        if max_ending_here < 0:
            max_ending_here = 0
            s = i+1
    if max_so_far > 0:
        return start, end
    else:
        return 0, -1
```

```
def task3(arr):
    shift = []
    for i in range(1, len(arr), 2):
        shift.append(arr[i] - arr[i - 1])
    start, end = maxSubArray(shift)
    start *= 2
    end = end * 2 + 1
    l = arr[:start]
    mid = arr[start:end + 1]
    mid.reverse()
    r = arr[end + 1:]
    l.extend(mid)
    l.extend(r)
    sum1 = 0
    for i in range(0, len(l), 2):
        sum1 += l[i]

    shift = []
    for i in range(2, len(arr), 2):
        shift.append(arr[i - 1] - arr[i])
    start, end = maxSubArray(shift)
    start = start * 2 + 1
    end = end * 2 + 1 + 1
    l = arr[:start]
    mid = arr[start:end + 1]
    mid.reverse()
    r = arr[end + 1:]
    l.extend(mid)
    l.extend(r)
    sum2 = 0
    for i in range(0, len(l), 2):
        sum2 += l[i]
    return max(sum1, sum2)

print("Ввод:")
res = []
t = int(input("t = "))
for i in range(t):
    arr = input("arr = ").split()
    arr = [int(arr[i]) for i in range(len(arr))]
```

```
    res.append(task3(arr))
print("Вывод:")
for i in range(t):
    print(res[i])
```

Задача 4

У Пети есть прямоугольная доска размера $n \times m$. Изначально на доске размещено k фишек, i-я из них находится в клетке, на пересечении sxi-й строки и syi-го столбца.
За одно действие Петя может сдвинуть все фишки влево, вправо, вниз или вверх на 1 ячейку.
Если фишка была в клетке (x,y), то после операции:
влево, ее координаты будут $(x,y-1)$; вправо, ее координаты будут $(x,y+1)$; вниз, ее координаты будут $(x+1,y)$; вверх, ее координаты будут $(x-1,y)$. Если фишка находится у стенки доски, и действие, выбранное Петей, двигает ее в направлении стены, то фишка остается на своей текущей позиции.
Обратите внимание, что несколько фишек могут располагаться в одной и той же клетке.
Для каждой фишки Петя выбрал позицию, в которой она должна побывать. Обратите внимание, что фишка не обязательно заканчивает в этой позиции.
Так как у Пети не очень много свободного времени, он готов сделать не более $2nm$ действий, описанных выше.
Вам предстоит выяснить, какие действия должен делать Петя, чтобы все фишки побывали хотя бы раз в выбранных для них клетках. Или определить, что за $2nm$ действий выполнить это невозможно.
Входные данные
Первая строка содержит три целых числа n,m,k $(1 \le n,m,k \le 200)$ — количество строк и столбцов доски и количество фишек соответственно.
Следующие k содержат по два целых чисел sxi,syi $(1 \le sxi \le n, 1 \le syi \le m)$ — начальная позиция i-й фишки.
Следующие k содержат по два целых чисел fxi,fyi $(1 \le fxi \le n, 1 \le fyi \le m)$ — позиция, которую должна посетить i-я фишка хотя бы раз.
Выходные данные
В первой строке выведите количество операций, чтобы каждая фишка посетила хотя бы раз позицию, которую выбрал для нее Петя.
Во второй строке выведите последовательность операций. Для обозначения операций влево, вправо, вниз, вверх используйте символы L,R,D,U соответственно.
Если искомой последовательности не существует, в единственной строке вы-

ведите -1.

Решение:

```
def task4(n, m, k, start, end):
    print(n + m + n * m - 3)
    for i in range(n - 1):
        print("U", end = '')
    for i in range(m - 1):
        print("L", end = '')
    for i in range(n):
        if i:
            print("D", end = '')
        for j in range(m - 1):
            if i % 2:
                print("L", end = '')
            else:
                print("R", end = '')

print("Ввод:")
[n, m, k] = input("n, m, k = ").split()
start = []
end = []
print("Начальные позиции:")
for i in range(int(k)):
    [x, y] = input().split()
    start.append([int(x), int(y)])
print("Конечные позиции:")
for i in range(int(k)):
    [x, y] = input().split()
    end.append([int(x), int(y)])
print("Вывод:")
task4(int(n), int(m), int(k), start, end)
```

Задача 5

Летние каникулы начались, поэтому Алиса и Боб хотят играть и веселиться, но... Их мама не согласна с этим. Она говорит, что они должны прочитать какое-то количество книг перед всеми развлечениями. Алиса и Боб прочитают каждую книгу вместе, чтобы быстрее закончить это задание.
В семейной библиотеке есть n книг. i-я книга характеризуется тремя целыми

числами: ti — количество времени, которое Алиса и Боб должны потратить, чтобы прочитать ее, ai (равное 1, если Алисе нравится i-я книга, и 0, если не нравится), и bi (равное 1, если Бобу нравится i-я книга, и 0, если не нравится).
Поэтому им нужно выбрать какие-то книги из имеющихся n книг таким образом, что:
Алисе нравятся не менее k книг из выбранного множества и Бобу нравятся не менее k книг из выбранного множества; общее время, затраченное на прочтение этих книг минимизировано (ведь они дети и хотят начать играть и веселиться как можно скорее). Множество, которое они выбирают, одинаковое и для Алисы и для Боба (они читают одни и те же книги), и они читают все книги вместе, таким образом, суммарное время чтения равно сумме ti по всем книгам, которые находятся в выбранном множестве.
Ваша задача — помочь им и найти любое подходящее множество книг или определить, что такое множество найти невозможно.
Входные данные
Первая строка теста содержит два целых числа n и k ($1 \leq k \leq n \leq 2 \cdot 105$).
Следующие n строк содержат описания книг, по одному описанию в строке: i-я строка содержит три целых числа ti, ai и bi ($1 \leq ti \leq 104$, $0 \leq ai, bi \leq 1$), где:
ti — количество времени, необходимое для прочтения i-й книги;
ai, равное 1, если Алисе нравится i-я книга, и 0 в обратном случае;
bi, равное 1, если Бобу нравится i-я книга, и 0 в обратном случае.
Выходные данные
Если подходящего решения не существует, выведите число -1. Иначе выведите целое число T — минимальное суммарное время, необходимое для прочтения подходящего множества книг.

Решение:

```
import math

def task5(arr, k):
    alice_and_bob = []
    alice = []
    bob = []
    for item in arr:
        if item[1] and item[2]:
            alice_and_bob.append(item[0])
        elif item[1] and not item[2]:
            alice.append(item[0])
        elif not item[1] and item[2]:
```

```
            bob.append(item[0])
    alice_and_bob.sort()
    alice.sort()
    bob.sort()
    minsum = math.inf
    i = 0
    while i <= k:
        books_left = k - i
        ab_arr = alice_and_bob[:i] # alice and bob
        a_arr = alice[:books_left] # only alice
        b_arr = bob[:books_left] # only bob
        k1 = len(ab_arr) + len(a_arr)
        k2 = len(ab_arr) + len(b_arr)
        current_sum = sum(ab_arr) + sum(b_arr) + sum(a_arr)
        if k1 == k2 == k and current_sum < minsum:
            minsum = current_sum
        i += 1
    if minsum == math.inf:
        return -1
    return minsum

    for i in range(len(alice_and_bob)):
        books_left = k - i
        if i > k:
            break
        if books_left > len(alice) or books_left > len(bob):
            continue
        current_sum = sum(alice_and_bob[:i + 1])
        + sum(alice[:books_left + 1]) + sum(bob[:books_left + 1])
        minsum = current_sum if current_sum < minsum else minsum
    return minsum

print("Ввод:")
[n, k] = input("n, k = ").split()
arr = []
for i in range(int(n)):
    [t, a, b] = input("t, a, b = ").split()
    arr.append([int(t), int(a), int(b)])
print("Вывод:")
print(task5(arr, int(k)))
```

Задача 6

Вы — амбициозный король, который хочет быть Императором Действительных чисел. Но перед этим вам нужно стать Императором Целых чисел.
Рассмотрим числовую ось. Столица вашей империи изначально находится в точке 0. На прямой есть n незахваченных королевств в позициях $0<x1<x2<\ldots<xn$.
Вы хотите захватить все эти королевства.
Вы можете делать два вида действий:
Вы можете переместить свою столицу (пусть ее текущая координата $c1$) в любое захваченное королевство (пусть его позиция $c2$), стоимость такого действия $a\cdot|c1-c2|$. Вы можете из текущей столицы (пусть ее текущая координата $c1$) захватить королевство (пусть его позиция $c2$), стоимость такого действия $b\cdot|c1-c2|$. Вы не можете захватывать королевство, если между вашей столицей и целью есть другие незахваченные королевства. Обратите внимание, что вы не можете расположить столицу в точке, где нет королевства. Другими словами, в любой момент времени ваша столица может быть только в точке 0 или в $x1,x2,\ldots,xn$. Также обратите внимание, что захват королевства не изменяет положение вашей столицы.
Выведите минимальную суммарную стоимость захвата всех королевств. Ваша столица может оказаться в итоге в любой точке.
Входные данные Первая строка содержит одно целое число t ($1\leq t\leq 1000$) — количество наборов входных данных. Далее следуют описания наборов.
Первая строка каждого набора содержит 3 целых числа n, a и b ($1\leq n\leq 2\cdot 105$; $1\leq a,b\leq 105$).
Вторая строка каждого набора содержит n целых чисел $x1,x2,\ldots,xn$ ($1\leq x1<x2<\ldots<xn\leq 108$).
Гарантируется, что сумма значений n по всем наборам входных данных не превосходит $2\cdot 105$.
Выходные данные Для каждого набора входных данных выведите одно число: минимальную стоимость захвата всех королевств.

Решение:

```
import math

# 1 решение
def task6(arr, a, b):
    arr.insert(0, 0)
    capital = 0
    ret = 0
    for target in range(1, len(arr)):
```

```
            mincost = math.inf
            saved_capital = capital
            for new_capital in range(target):
                curr_cost = a * abs(arr[new_capital] - arr[capital])
                    + b * abs(arr[new_capital] - arr[target])
                if mincost > curr_cost:
                    mincost = curr_cost
                    saved_capital = new_capital
            capital = saved_capital
            ret += mincost
        return ret

    # 2 решение
    def __t6(arr, a, b, capital, target, accum, result):
        if target == len(arr):
            result.append(accum)
            return
        for new_capital in range(target):
            inc_cost = a * abs(arr[new_capital] - arr[capital])
            + b * abs(arr[new_capital] - arr[target])
            __t6(arr, a, b, new_capital, target + 1, accum + inc_cost, result)

    def t6(arr, a, b):
        arr.insert(0, 0)
        result = []
        __t6(arr, a, b, 0, 1, 0, result)
        return min(result)

    print("Ввод:")
    result = []
    t = int(input("t = "))
    for i in range(t):
        [a, b] = input("a, b = ").split()
        arr = input("arr = ").split()
        arr = [int(arr[i]) for i in range(len(arr))]
        result.append(t6(arr, int(a), int(b)))
    print("Вывод:")
    for item in result:
        print(item)
```

Задача 7

Вам даны две последовательности целых чисел $a1,\ldots,an$ и $b1,\ldots,bm$. Для каждого $j{=}1,\ldots,m$ найдите наибольший общий делитель чисел $a1{+}bj,\ldots,an{+}bj$.
Входные данные
В первой строке записано два целых числа n и m ($1{\le}n,m{\le}2{\cdot}105$).
Во второй строке записано n целых чисел $a1,\ldots,an$ ($1{\le}ai{\le}1018$).
В третьей строке записано m целых чисел $b1,\ldots,bm$ ($1{\le}bj{\le}1018$).
Выходные данные
Выведите m целых чисел. j-е из этих чисел должно быть равно НОД($a1{+}bj,\ldots,an{+}bj$).

Решение:

```
# Алгоритм Евклида
def gcd(a, b):
    while a != 0 and b != 0:
        if a > b:
            a = a % b
        else:
            b = b % a
    return a + b

# НОД всех элементов массива
def gcd_arr(arr):
    result = arr[0]
    for i in range(1, len(arr)):
        result = gcd(result, arr[i])
    return result

def task7(a, b):
    result = []
    for j in range(len(b)):
        temp = []
        for i in range(len(a)):
            temp.append(b[j] + a[i])
        result.append(gcd_arr(temp))
    return result

print("Ввод:")
a = input("a = ").split()
a = [int(a[i]) for i in range(len(a))]
```

```
b = input("b = ").split()
b = [int(b[i]) for i in range(len(b))]
print("Вывод:")
print(task7(a, b))
```

Задача 8

Вам заданы два массива целых чисел a и b длины n.
Вы можете развернуть не более одного подмассива (последовательного отрезка) массива a.
Ваша задача состоит в том, чтобы перевернуть такой подмассив, чтобы сумма $\sum i=1nai\cdot bi$ была максимально возможной.
Входные данные
Первая строка содержит одно целое число n ($1\leq n\leq 5000$).
Вторая строка содержит n целых чисел $a1,a2,\ldots,an$ ($1\leq ai\leq 107$).
Третья строка содержит n целых чисел $b1,b2,\ldots,bn$ ($1\leq bi\leq 107$).
Выходные данные
Выведите одно целое число — максимально возможную сумма после разворота не более одного подмассива (последовательного отрезка) a.

Решение:

```
def task8(a, b):
    maxdiff = 0
    result = [a[0] * b[0], 0, 0]
    for mid in range(len(a)):
        # Для нечетного размера подмассива
        l = mid - 1
        r = mid + 1
        curr_max = a[mid] * b[mid]
        curr_rmax = curr_max
        while l >= 0 and r <= len(a) - 1:
            curr_max += a[l] * b[l] + a[r] * b[r]
            curr_rmax += a[r] * b[l] + a[l] * b[r]
            if curr_rmax - curr_max > maxdiff:
                maxdiff = curr_rmax - curr_max
                result = [curr_rmax, l, r]
            l -= 1
            r += 1

        # Для четного размера подмассива
```

```
            l = mid
            r = mid + 1
            curr_max = 0
            curr_rmax = 0
            while l >= 0 and r <= len(a) - 1:
                curr_max += a[l] * b[l] + a[r] * b[r]
                curr_rmax += a[r] * b[l] + a[l] * b[r]
                if curr_rmax - curr_max > maxdiff:
                    maxdiff = curr_rmax - curr_max
                    result = [curr_rmax, l, r]
                l -= 1
                r += 1
        # Ищем сумму левого подмассива
        lmax = 0
        for i in range(result[1]):
            lmax += a[i] * b[i]
        # Ищем сумму правого подмассива
        rmax = 0
        for i in range(result[2] + 1, len(a)):
            rmax += a[i] * b[i]
        return lmax + result[0] + rmax

print("Ввод:")
a = input("a = ").split()
a = [int(a[i]) for i in range(len(a))]
b = input("b = ").split()
b = [int(b[i]) for i in range(len(b))]
print("Вывод:")
print(task8(a, b))
```

Задача 9

Жил-был в одной стране Царь по имени Цопа. После очередной царской реформы полномочия Царя стали настолько широкими, что, в частности, он стал собственноручно заниматься финансовой отчётностью.
Известен суммарный доход A его Царства по итогам 0-го года, и суммарный доход B по итогам n-го года (оба числа могут быть отрицательными, что означает — в этот год экономика Царства была убыточной).
Царь хочет продемонстрировать финансовую стабильность, для этого ему надо подобрать единый коэффициент X — коэффициент роста дохода Царства

за один год. Этот коэффициент должен удовлетворять уравнению:
A·Xn = B. Разумеется, Царь не собирается делать такую работу вручную, и требует от вас написать программу, ищущую этот коэффициент X.
Следует отметить, что дробные числа крайне не любят в экономических структурах Царства, поэтому как входные данные, так и искомый коэффициент должны быть целыми. Искомый коэффициент X может оказаться равным нулю или даже быть отрицательным.
Входные данные
В единственной строке записаны три целых числа A, B, n, удовлетворяющих условиям: $|A|, |B| \leq 1000$, $1 \leq n \leq 10$.
Выходные данные
Выведите единственное целое число — искомый целый коэффициент X, или фразу «No solution», если такого коэффициента не существует, или он не целый. Если ответов несколько, выведите любой.

Решение:

```
def task9(A, B, n):
    result = 0
    if A == 0 and B or (A < 0 and B > 0 or B < 0 and A > 0) and n % 2 == 0:
        return "No solution"
    elif A == 0 and B == 0:
        return 5
    elif (A < 0 and B > 0 or B < 0 and A > 0) and n % 2:
        result = -((-B / A) ** (1 / n))
    else:
        result = (B / A) ** (1 / n)
    if result != int(result):
            return "No solution"
    return int(result)

print("Ввод:")
[A, B, n] = input("A, B, n = ").split()
print("Вывод:")
print(task9(int(A), int(B), int(n)))
```

Задача 10

У Алисы есть торт, и она собирается его разрезать. Она выполнит следующую операцию $n-1$ раз: выберет кусочек пирога (изначально весь торт является одним куском) с весом $w \geq 2$ и разрежем его на два более маленьких кусочка с

весами $\lfloor w2 \rfloor$ и $\lceil w2 \rceil$ ($\lfloor x \rfloor$ и $\lceil x \rceil$ обозначают округление вниз и вверх, соответственно).
После того как Алиса разрежет торт на n кусочков, она расположит эти n кусочков на столе в произвольном порядке. Обозначим за ai вес i-го кусочка. Вам дан массив a. Определите, существует ли некоторый начальный вес торта и последовательность операций разрезания такие, чтобы веса кусочков получались равными a.
Входные данные
Первая строка содержит одно целое число t ($1 \leq t \leq 104$) — количество наборов входных данных.
Первая строка каждого набора содержит одно целое число n ($1 \leq n \leq 2 \cdot 105$).
Вторая строка содержит n целых чисел $a1, a2, \ldots, an$ ($1 \leq ai \leq 109$).
Гарантируется, что сумма n для всех наборов входных данных не превышает $2 \cdot 105$.
Выходные данные
Для каждого набора входных данных выведите одну строку: выведите YES, если массив a может быть получен с помощью действий Алисы, иначе выведите NO.
Вы можете выводить буквы в любом регистре (например, YES, Yes, yes, yEs будут приняты как положительный ответ).

Решение:

```
import bisect
import math

def task10(w1):
    w1.sort()
    w2 = [sum(w1)]

    while len(w1) != 0 and len(w2) != 0:
        if w1[-1] > w2[-1]:
            return "NO"
        elif w1[-1] == w2[-1]:
            w1.pop(-1)
            w2.pop(-1)
        else:
            item = w2.pop(-1)
            bisect.insort(w2, math.ceil(item / 2))
            bisect.insort(w2, math.floor(item / 2))
    return "YES"
```

```
print("Ввод:")
result = []
t = int(input("t = "))
for i in range(t):
    arr = input("arr = ").split()
    arr = [int(arr[i]) for i in range(len(arr))]
    result.append(task10(arr))
print("Вывод:")
for item in result:
    print(item)
```

Вывод

Я разработал алгоритмы решения и их реализацию для десяти задач, используя язык Python.