Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе Жадные алгоритмы (Stepic)

Студент Жумиков Егор группы P3218

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Задача 1: покрыть отрезки точками

По данным n отрезкам необходимо найти множество точек минимального размера, для которого каждый из отрезков содержит хотя бы одну из точек.

В первой строке дано число $1 \le n \le 100$

отрезков. Каждая из последующих n строк содержит по два числа $0 \le l \le r \le 10^9$, задающих начало и конец отрезка. Выведите оптимальное число m точек и сами m

точек. Если таких множеств точек несколько, выведите любое из них.

Sample Input 1:

```
3
1 3
2 5
3 6
```

Sample Output 1:

```
1
3
```

Sample Input 2:

```
4
4 7
1 3
2 5
5 6
```

Sample Output 2:

```
2
3 6
```

Исходный код к задаче 1

```
from bisect import bisect

n = int(input())
dots = []
tt = ['b', 'e']
ss = [False] * n
result = []

class Dot:
    def __init__(self, v, t, s, segments=[]):
        self.v = v
        self.t = t
        self.s = s
        self.segments = segments

    def __str__(self):
        return f'D({self.v}, {self.t}, {self.s}, {len(self.segments)})'

for i in range(n):
    for j, v in enumerate(map(int, input().split())):
        # TODO: optimize insertion
        dots.append(Dot(v, tt[j], i))

dots = sorted(dots, key=lambda d: d.v * 1000 + (0 if d.t == 'b' else 1))
curr_segments = []
```

```
for i in range(n * 2):
    if dots[i].t == 'b':
        curr_segments.append(dots[i].s)
    else:
        if ss[dots[i].s]:
            continue

        result.append(dots[i].v)
        for s in curr_segments:
            ss[s] = True
        curr_segments = [ ]

print(len(result))
print(' '.join(map(str, result)))
```

Задача 2: непрерывный рюкзак

Первая строка содержит количество предметов $1 \le n \le 10^3$ и вместимость рюкзака $0 \le W \le 2 \cdot 10^6$. Каждая из следующих n строк задаёт стоимость $0 \le c_i \le 2 \cdot 10^6$ и объём $0 < w_i \le 2 \cdot 10^6$ предмета (n, W, c_i, w_i — целые числа). Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

Sample Input:

```
3 50
60 20
100 50
120 30
```

Sample Output:

```
180.000
```

Исходный код к задаче 2

```
class Item:
    def __init__(self, c, w):
        self.c = c
        self.w = w
        self.p = self.c / self.w

    def __str__(self):
        return f'({self.p, self.c, self.w})'

n, w = map(int, input().split())
items = sorted([Item(*[int(x) for x in input().split()]) for i in range(n) ], key=lambda i: i.p)

s = 0
while w > 0 and len(items) > 0:
    i = items.pop()
    cw = min(w, i.w)
    s += cw * i.p
    w -= cw

print(s)
```

Задача 3: различные слагаемые

По данному числу $1 \le n \le 10^9$ найдите максимальное число k, для которого n можно представить как сумму k различных натуральных слагаемых. Выведите в первой строке число k, во второй — k слагаемых.

Sample Input 1:

```
4
```

Sample Output 1:

```
2
1 3
```

Sample Input 2:

```
6
```

Sample Output 2:

```
3
1 2 3
```

Исходный код к задаче 3

```
from bisect import bisect

n = int(input())

class S:
    def __getitem__(self, i):
        return (i * (i + 1)) // 2

    def __len__(self):
        return n + 1

    def __repr__(self):
        return '[ ' + ', '.join(str(self[i]) for i in range(len(self))) + ' ]'

s = S()
c = bisect(s, n) - 1
print(c)
print(' '.join(str(i if i < c else n - s[i - 1]) for i in range(1, c + 1)))
```

Задача 4: кодирование Хаффмана

По данной непустой строке s длины не более 10^4, состоящей из строчных букв латинского алфавита, постройте оптимальный беспрефиксный код. В первой строке выведите количество различных букв k, встречающихся в строке, и размер получившейся закодированной строки. В следующих k строках запишите коды букв в формате "letter: code". В последней строке выведите закодированную строку.

Sample Input 1:

```
a
```

Sample Output 1:

```
1 1
a: 0
0
```

Sample Input 2:

```
abacabad
```

Sample Output 2:

```
4 14
a: 0
b: 10
c: 110
d: 111
01001100100111
```

Исходный код к задаче 4

```
from bisect import bisect
from collections import Counter

class N:
    def __init__(self, p, v=None, l=None, r=None):
        self.v = v
        self.p = p
        self.l = l
        self.r = r

    def __str__(self):
        return f'N({self.p}, {self.v or ("(" + str(self.l) + ", " + str(self.r) + ")")})'


s = input()
p = Counter(s)
nodes = []
letters = []
a = {}

class NL:
    def __getitem__(self, i):
        return nodes[i].p

    def __len__(self):
        return len(nodes)

    def __repr__(self):
        return '[ ' + ', '.join(str(nodes[i]) for i in range(len(self))) + ' ]'

for l, c in p.most_common()[::-1]:
    nodes.append(N(c, l))
    a[l] = ''
    letters.append(l)

k = len(nodes)

while len(nodes) > 1:
    l = nodes.pop(0)
    r = nodes.pop(0)
    n = N(l.p + r.p, l=l, r=r)
    nodes.insert(bisect(NL(), n.p), n)

nodes_stack = [('', nodes[0])]

while len(nodes_stack):
    cs, n = nodes_stack.pop()

    if n.v:
        if cs == '':
            cs = '0'

        a[n.v] = cs
    else:
        nodes_stack.append((cs + '0', n.l))
        nodes_stack.append((cs + '1', n.r))

s = ''.join(a[c] for c in s)
print(f'{k} {len(s)}')
for l in letters:
    print(f'{l}: {a[l]}')
print(s)
```

Задача 5: декодирование Хаффмана

Восстановите строку по её коду и беспрефиксному коду символов.

В первой строке входного файла заданы два целых числа k и l через пробел — количество различных букв, встречающихся в строке, и размер получившейся закодированной строки, соответственно. В следующих k

строках записаны коды букв в формате "letter: code". Ни один код не является префиксом другого. Буквы могут быть перечислены в любом порядке. В качестве букв могут встречаться лишь строчные буквы латинского алфавита; каждая из этих букв встречается в строке хотя бы один раз. Наконец, в последней строке записана закодированная строка. Исходная строка и коды всех букв непусты. Заданный код таков, что закодированная строка имеет минимальный возможный размер.

В первой строке выходного файла выведите строку s. Она должна состоять из строчных букв латинского алфавита. Гарантируется, что длина правильного ответа не превосходит 10^4 символов.

Sample Input 1:

```
1 1
a: 0
0
```

Sample Output 1:

```
a
```

Sample Input 2:

```
4 14
a: 0
b: 10
c: 110
d: 111
01001100100111
```

Sample Output 2:

```
abacabad
```

Исходный код к задаче 5

```
from bisect import bisect
from collections import Counter

class N:
    def __init__(self, v=None, l=None, r=None):
        self.v = v
        self.l = l
        self.r = r

    def __getitem__(self, i):
        r = None
        if i == '0':
            r = self.l
        elif i == '1':
            r = self.r

        r = r or N()
```

```
        self[i] = r
        return r

    def __setitem__(self, i, item):
        if i == '0':
            self.l = item
        elif i == '1':
            self.r = item

    def __str__(self):
        return f'N("{self.v}", 0: {self.l}, 1: {self.r})'

root = N()
k, l = map(int, input().split())

for i in range(k):
    l, c = input().split()
    l = l[0]
    c = list(c)
    n = root

    while len(c):
        n = n[c.pop(0)]

    n.v = l

n = root
s = input()
r = ''

for c in s:
    n = n[c]
    if n.v:
        r += n.v
        n = root


print(r)
```

Задача 6: очередь с приоритетами

Первая строка входа содержит число операций $1 \le n \le 10^5$. Каждая из последующих n строк задают операцию одного из следующих двух типов:

- Insert x, где $0 \le x \le 10^9$ — целое число;
- ExtractMax.

Первая операция добавляет число x в очередь с приоритетами, вторая — извлекает максимальное число и выводит его.

Sample Input:

```
6
Insert 200
Insert 10
ExtractMax
Insert 5
Insert 500
ExtractMax
```

Sample Output:

```
200
500
```

Исходный код к задаче 6

```
# solution goes here:
a = [(lambda: [[(lambda command, v, *_: {'Insert': lambda:
bisect.insort_left(l, int(v)), 'ExtractMax': lambda:
print(l.pop())}[command]())(*(input().split() + ['0'])) for i in
range(int(input()))]for l in [list()]])() for l in [[]] for bisect
in [__import__('bisect')]]
```