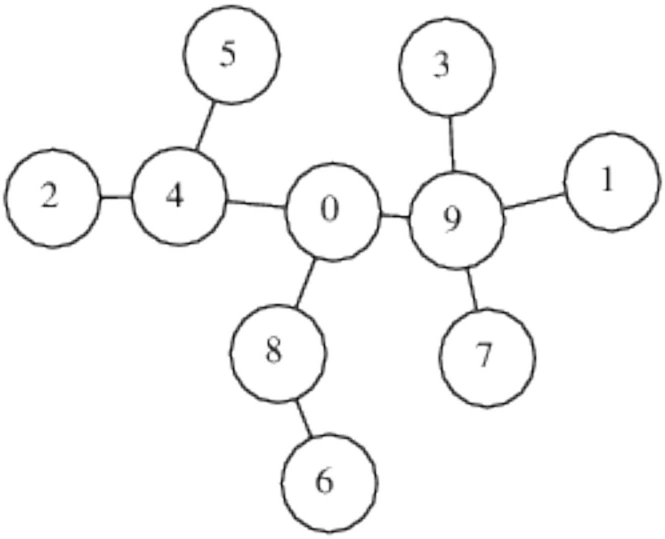
Skill test for python developer

Instructions

There are 6 programming problems described below. Pick 2 out of 6 problems that you'd like to solve and write a solution in python. Please pay attention to `Complexity` section added to every task.

Task #1

A computer network consisting of N routers and N−1 links connecting them is given. Routers are labeled with distinct integers within the range [0..(N−1)]. Links connect routers in such a way that each distinct pair of routers is connected either by a direct link or through a path consisting of direct links. There is exactly one way to reach any router from another and the number of direct links that must be traversed is called the *distance* between these two routers. For example, consider the following network consisting of ten routers and nine links:



Routers 2 and 4 are connected directly, so the distance between them is 1. Routers 4 and 7 are connected through a path consisting of direct links 4−0, 0−9 and 9−7; hence the distance between them is 3.

The location of a router in the network determines how quickly a packet dispatched by that router can reach other routers. The *peripherality* of a router is the average distance to all other routers on the network. For example, the peripherality of router 4 in the network shown above is 2.11, because:

distance to 0:	1
distance to 1:	3
distance to 2:	1
distance to 3:	3
distance to 5:	1
distance to 6:	3
distance to 7:	3
distance to 8:	2
distance to 9:	2
average:	19/9 = 2.11

The peripherality of router 0 is 1.66 and no other router has lower peripherality.

Write a function

```
def min_router_peripherality(T)
```

that, given a non-empty zero-indexed array T consisting of N integers describing a network of N routers and N−1 links, returns the label of the router that has minimum peripherality. If there is more than one router that has minimum peripherality, the function should return the lowest label.

Array T describes a network of routers as follows:

- if T[P] = Q and P ≠ Q, then there is a direct link between routers P and Q.

For example, given the following array T consisting of ten elements:

```
T[0] = 9    T[1] = 1    T[2] = 4
T[3] = 9    T[4] = 0    T[5] = 4
T[6] = 8    T[7] = 9    T[8] = 0
T[9] = 1
```

the function should return 0, because this array describes the network shown above and router 0 has minimum peripherality.

Assume that:

- N is an integer within the range [1..100,000];
- each element of array T is an integer within the range [0..(N−1)];
- there is exactly one (possibly indirect) connection between any two distinct routers.

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Task #2

A positive integer N is given. Consider the sequence of numbers [0, 1, ..., N]. What is the total number of zeros in the decimal representations of these numbers?

N can be very large. Hence, it is given in the form of a non-empty string S of length L, containing a decimal representation of N. S contains no leading zeros.

Write a function:

```
def number_of_zeros(S)
```

that, given a string S, which is a decimal representation of some positive integer N, returns the total number of zeros in the decimal representations of numbers [0, 1, ..., N]. If the result exceeds 1,410,000,016, the function should return the remainder from the division of the result by 1,410,000,017.

For example, for S="100" the function should return 12 and for S="219" it should return 42.

Assume that:

- L is an integer within the range [1..10,000];
- string S consists only of digits (0-9);
- string S contains no leading zeros.

Complexity:

- expected worst-case time complexity is O(L);
- expected worst-case space complexity is O(L) (not counting the storage required for input arguments).

Task #3

N countries (numbered from 0 to N−1) participate in a space mission. Each country has trained a certain number of astronauts and each country has to delegate a certain number of astronauts to the mission's crew. How many different ways are there to select the crew?

For example, suppose there are three countries A-land, B-land and C-land and

- A-land has 6 astronauts;
- B-land has 4 astronauts;
- C-land has 7 astronauts.

and

- A-land has to delegate 1 astronaut;
- B-land has to delegate 3 astronauts;
- C-land has to delegate 4 astronauts.

Then

- there are 6 different ways in which A-land can delegate 1 out of 6 astronauts;
- there are 4 different ways in which B-land can delegate 3 out of 4 astronauts;
- there are 35 different ways in which C-land can delegate 4 out of 7 astronauts.

Each country's choice is independent, so the total number of different ways to build the mission crew is 6*4*35=840.

Write a function

```
def space_crews(T,D)
```

that, given two non-empty zero-indexed arrays T and D consisting of N integers each, returns the number of different ways in which the space crew can be selected, where:

- T[K] = number of astronauts in country K;
- D[K] = number of astronauts to be delegated from country K.

Assume that:

- N is an integer within the range [1..1,000];
- each element of array T is an integer within the range [0..1,000,000];
- each element of array D is an integer within the range [0..1,000,000];
- T[i] ≥ D[i] for i=0..(N−1).

For example, given N=3 and

```
T[0] = 6  T[1] = 4  T[2] = 7
D[0] = 1  D[1] = 3  D[2] = 4
```

the function should return 840, as explained above. If the result exceeds 1,410,000,016, the function should return the remainder of the result modulo 1,410,000,017.

Complexity:

- expected worst-case time complexity is O(max(T)*log(max(T))+N);
- expected worst-case space complexity is O(N+max(T)), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Task #4

A non-empty zero-indexed array A consisting of N integers is given. The *first covering prefix* of array A is the smallest integer P such that 0≤P<N and such that every value that occurs in array A also occurs in sequence A[0], A[1], ..., A[P].

For example, the first covering prefix of the following 5−element array A:

```
A[0] = 2  A[1] = 2  A[2] = 1
A[3] = 0  A[4] = 1
```

is 3, because sequence [A[0], A[1], A[2], A[3]] equal to [2, 2, 1, 0], contains all values that occur in array A.

Write a function

```
def ps(A)
```

that, given a zero-indexed non-empty array A consisting of N integers, returns the first covering prefix of A.

Assume that:

- N is an integer within the range [1..1,000,000];
- each element of array A is an integer within the range [0..N−1].

For example, given array A such that

```
A[0] = 2  A[1] = 2  A[2] = 1
A[3] = 0  A[4] = 1
```

the function should return 3, as explained above.

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Task #5

Given an array A of N integers we draw N discs in a 2D plane, such that i-th disc has center in (0,i) and a radius A[i]. We say that k-th disc and j-th disc intersect, if k ≠ j and k-th and j-th discs have at least one common point.

Write a function

```
def number_of_disc_intersections(A)
```

which given an array A describing N discs as explained above, returns the number of pairs of intersecting discs. For example, given N=6 and

```
A[0] = 1  A[1] = 5  A[2] = 2
A[3] = 1  A[4] = 4  A[5] = 0
```

there are 11 pairs of intersecting discs:

- 0th and 1st;
- 0th and 2nd;
- 0th and 4th;
- 1st and 2nd;
- 1st and 3rd;
- 1st and 4th;
- 1st and 5th;
- 2nd and 3rd;
- 2nd and 4th;
- 3rd and 4th;
- 4th and 5th.

so the function should return 11.

Assume that:

- N is an integer within the range [0..10,000,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

The function should return −1 if the number of intersecting pairs exceeds 10,000,000.

Complexity:

- expected worst-case time complexity is O(N*log(N));
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Task #6

A zero-indexed array A consisting of N integers is given. An*equilibrium index* of this array is any integer P such that 0 ≤ P < N and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

A[0] + A[1] + ... + A[P−1] = A[P+1] + ... + A[N−2] + A[N−1].

Sum of zero elements is assumed to be equal to 0. This can happen if P = 0 or if P = N−1.

For example, consider the following array A consisting of N = 7 elements:

```
A[0] = -7   A[1] =  1   A[2] = 5
A[3] =  2   A[4] = -4   A[5] = 3
A[6] =  0
```

P = 3 is an equilibrium index of this array, because A[0] + A[1] + A[2] = A[4] + A[5] + A[6].

P = 6 is also an equilibrium index, because: A[0] + A[1] + A[2] + A[3] + A[4] + A[5] = 0 and there are no elements with indices greater than 6.

P = 7 is not an equilibrium index, because it does not fulfill the condition 0 ≤ P < N.

Write a function

```
def equi(A)
```

that, given a zero-indexed array A consisting of N integers, returns any of its equilibrium indices. The function should return −1 if no equilibrium index exists.

Assume that:

- N is an integer within the range [0..10,000,000];

- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

For example, given array A such that

```
A[0] = -7   A[1] =  1   A[2] = 5
A[3] =  2   A[4] = -4   A[5] = 3
A[6] =  0
```

the function may return 3 or 6, as explained above.

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.