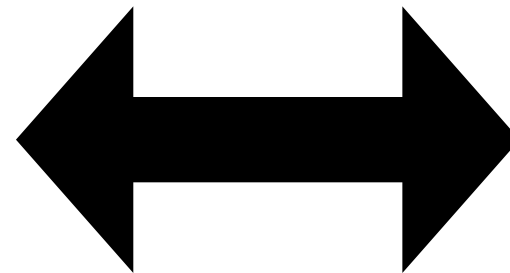


Coding 1

Next Lecture

**Competitive Programming
and Contests**

**Wednesday 4 October
14:00—16:00
Room Lab I**



Algorithm Design

**Thursday 5 October
9:00—11:00
Room L1**

Is ok for everybody?

🔗 Background

If you wish to refresh your mind on basic Algorithms and Data Structures, I suggest you to look at the well-known book [Introduction to Algorithms, 3rd Edition](#) by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

I strongly suggest you to watch the following video lectures as soon as possible.

- [Insertion Sort and Merge Sort](#)
- [Heaps and Heap Sort](#)
- [Counting Sort, Radix Sort, Lower Bounds for Sorting](#)
- [Binary Search Trees](#)
- [AVL trees](#)
- [Hashing with Chaining](#)

Exam

The exam consists of two parts: Homeworks 50% - Written/oral test 50%.

Extra points for

- active participation to our Google group
- serious participation to online contests, e.g., CodeForces, TopCoder, Hacker Rank, Google Code Jam, ...
- successful interview with a big company

For full marks in the "Homeworks" part of the exam you must solve all the problems. I'll check your solutions by reading the code and their descriptions and by submitting them to the problems' sites. I should be able to submit any solution without any change. **A solution that doesn't pass the tests (for any reason) will be considered wrong!**

Please create a [github](#) repository to collect all your solutions and their descriptions. The repository can be either private or public. In both cases I should be able to access it. Please send me a link to your repository and keep the repository updated. I should be able to monitor your progresses.

The quality of your code will be evaluated. As your style will improve a lot with practice, I'll evaluate only the quality of the last submitted solutions.

How to solve a problem

- Read carefully the description of the problem.
- Make sure you understand the problem by checking the examples.
- Design a first trivial solution.
- Think about a more efficient solution. The use of some running examples usually helps a lot in finding a better solution. If you are not able to find such solution, try to find some hints by discussing with other students, by asking questions on the group, by looking at the solution in our Web page, or by searching on internet. This is perfectly fine for the first problems and for most difficult ones. In any case, make sure you really understand the solution and the properties it is exploiting! Always compare your solution with existing ones.
- Write a brief description of your solution in English. Provide an analysis of its time and space complexity.
- Implement your own solution in C++.
- Submit your implementation to the problem's site. Fix it until it passes all the tests.

This is exactly what I'm expecting!

The screenshot shows the GitHub interface for the repository 'CPC-Problems'. The repository has 1 watch, 0 stars, and 0 forks. The description is 'Collection of all the solved and implemented problems of the course Competitive Programming and...'. The repository has 37 commits, 1 branch, and 0 releases. The latest commit is 'Update README.md' by user '9c1ac5f' 21 hours ago. The file list on the left includes 'Kadane', 'Leaders in Array', 'Missing element in Array', 'Next Larger Element', 'Sliding Window', 'Towers', 'Trapping Water', and 'README.md'. A large green sign with the text 'Start Now Don't Wait' is overlaid on the bottom half of the page. A blue callout bubble points to the 'Unwatch' button with the text 'Am I watching yours? If not, send email'.

This repository Search Pull requests Issues Marketplace Explore

CPC-Problems

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Collection of all the solved and implemented problems of the course Competitive Programming and...

37 commits 1 branch 0 releases

Branch: master New pull request Create new file Upload files Find file Clone or download

committed on GitHub Update README.md Latest commit 9c1ac5f 21 hours ago

Kadane

Leaders in Array

Missing element in Array

Next Larger Element

Sliding Window

Towers

Trapping Water

README.md

README.md

CPC-Pro

Start Now Don't Wait

Am I watching yours?
If not, send email

Description

 [README.md](#)

Update README.md

21 hours ago

 [Sliding.cpp](#)

Adding new solutions

3 days ago

 README.md

Sliding window algorithm

Description

To find the maximum element in all the windows of size k , we use a queue in which we insert all the elements that can be useful to determine which is the maximum. So we start from an empty queue and we insert the elements of the array following this rule:

- If there are elements from the head of the queue which are no longer in the window, we remove those elements;
- If there are elements from the back of the queue which are smaller than the current element, we remove those elements. Then we insert the new element from the back of the queue.

Notice that in doing these, to determine the maximum element we just have to watch at the first element in the queue, because it is certainly in the window (otherwise it would have been removed since we start the check from the head), and it's the maximum because when we insert a new element we remove all the smaller elements and we put it at the end of the queue, so the elements are sorted from the largest (the first) to the smallest (the last).

Complexity

The time complexity of the algorithm is $O(n)$. To see why, we have to notice that every element is inserted in the queue just once, so all the insertions cost $O(n)$, and every time we insert an element (after k iterations) we remove at least one, so we can't insert more elements than we remove. So the total time complexity is $O(n)$. To see the space complexity, we have to notice that during the execution of the algorithm, it can not happen that there are elements not in the window in the queue, because imagine we insert an element x which is not the maximum (so it

Code

```
1  #include <iostream>
2  #include <vector>
3  #include <stack>
4  #include <tuple>
5
6
7  std::pair<std::vector<int>, int> getArrayInput(){
8      int size;
9      int k;
10     std::cin >> size >> k;
11     std::vector<int> v(size);
12
13     for(int i=0; i<size; i++){
14         std::cin >> v[i];
15     }
16     return std::make_pair (v, k);
17 }
18
19 void maxWindow(std::vector<int> v, int k){
20     std::deque<std::pair<int, int>> q;
21
22     for (int i=0; i<v.size(); i++){
23
24         while (!q.empty() && q.front().second <= i - k)
25             q.pop_front();
26
27         while (!q.empty() && q.back().first <= v[i])
28             q.pop_back();
29
30         q.push_back(std::make_pair(v[i], i));
31
32         if (i >= k-1) std::cout << q.front().first << ' ';
33
34     }
35
36     std::cout << std::endl;
37 }
38
```

```
39 int main() {
40
41     int numberTest;
42     std::cin >> numberTest;
43     std::vector<std::vector<int>> test(numberTest);
44     std::vector<int> k(numberTest);
45
46     for (int i=0; i<numberTest; i++){
47         std::tie (test[i], k[i]) = getArrayInput();
48     }
49
50     for (int i=0; i<numberTest; i++)
51         maxWindow(test[i], k[i]);
52
53     return 0;
54 }
```

Exercises

1. Towers

<http://codeforces.com/problemset/problem/37/A?locale=en>

Little Vasya has received a young builder's kit. The kit consists of several wooden bars, the lengths of all of them are known. The bars can be put one on the top of the other if their lengths are the same.

Vasya wants to construct the minimal number of towers from the bars. Help Vasya to use the bars in the best way possible.

Input

The first line contains an integer N ($1 \leq N \leq 1000$) — the number of bars at Vasya's disposal. The second line contains N space-separated integers l_i — the lengths of the bars. All the lengths are natural numbers not exceeding 1000.

Output

In one line output two numbers — the height of the largest tower and their total number. Remember that Vasya should use all the bars.

Examples

input

3

1 2 3

output

1 3

input

4

6 5 6 7

output

2 3

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  template<typename T>
8  vector<T> get_input_sequence(size_t n) {
9      vector<T> sequence(n);
10
11      for(size_t i = 0; i < n; ++i)
12          cin >> sequence[i];
13      return sequence;
14  }
15
16  int main() {
17      std::ios_base::sync_with_stdio(false);
18
19      size_t n;
20      cin >> n;
21      auto length = get_input_sequence<int>(n);
22      sort(length.begin(), length.end());
23
24      size_t numTowers = 0, currentLength = 0, currentHeight = 0, maxHeight = 1;
25      for(size_t k = 0; k < n; k++) {
26          if(length[k] == currentLength) {
27              currentHeight += 1;
28              if(currentHeight > maxHeight)
29                  maxHeight = currentHeight;
30          } else {
31              currentLength = length[k];
32              currentHeight = 1;
33              numTowers++;
34          }
35      }
36      cout << maxHeight << " " << numTowers << endl;
37      return 0;
38  }
```

Exercises

2. Finding Team Member

<http://codeforces.com/problemset/problem/579/B?locale=en>

There is a programming contest named SnakeUp, $2n$ people want to compete for it. In order to attend this contest, people need to form teams of exactly two people. You are given the strength of each possible combination of two people. All the values of the strengths are **distinct**.

Every contestant hopes that he can find a teammate so that their team's strength is as high as possible. That is, a contestant will form a team with highest strength possible by choosing a teammate from ones who are willing to be a teammate with him/her. More formally, two people A and B may form a team if each of them is the best possible teammate (among the contestants that remain unpaired) for the other one.

Can you determine who will be each person's teammate?

Input

There are $2n$ lines in the input.

The first line contains an integer n ($1 \leq n \leq 400$) — the number of teams to be formed.

The i -th line ($i > 1$) contains $i - 1$ numbers $a_{i1}, a_{i2}, \dots, a_{i(i-1)}$. Here a_{ij} ($1 \leq a_{ij} \leq 10^6$, all a_{ij} are distinct) consisting of person i and person j (people are numbered starting from 1.)

Output

Output a line containing $2n$ numbers. The i -th number should represent the number of teammate of i -th

Examples

input

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

output

```
2 1 4 3
```

input

```
3
487060
3831 161856
845957 794650 976977
83847 50566 691206 498447
698377 156232 59015 382455 626960
```

output

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

```

7  struct team {
8      size_t m1;
9      size_t m2;
10     int32_t strength;
11
12     team(size_t a, size_t b, int64_t s)
13         : m1(a)
14         , m2(b)
15         , strength(s) {}
16 };
17
18 int main() {
19     std::ios_base::sync_with_stdio(false);
20
21     size_t n;
22     cin >> n;
23
24     vector<team> teams;
25     teams.reserve(n*(n-1)/2);
26
27     for(size_t i = 1; i < 2*n; ++i) {
28         for(size_t j = 0; j < i; ++j) {
29             int32_t strength;
30             cin >> strength;
31             teams.emplace_back(i, j, strength);
32         }
33     }
34
35     sort(teams.begin(),
36         teams.end(),
37         [](const team &a, const team &b) {
38             return a.strength > b.strength;
39         }
40     );
41

```

```

42     vector<size_t> assign(2*n);
43     for(auto &team: teams) {
44         if(assign[team.m1] == 0 and assign[team.m2] == 0) {
45             assign[team.m1] = team.m2 + 1; // +1, 0 is reserved to "no assignment yet"
46             assign[team.m2] = team.m1 + 1;
47         }
48     }
49
50     for(auto assignment: assign)
51         cout << assignment << " ";
52     cout << endl;
53
54     return 0;
55 }

```

Exercises

3. Megacity

<http://codeforces.com/problemset/problem/424/B?locale=en>

The administration of the Tomsk Region firmly believes that it's time to become a megacity (that is, get population of one million). Of improving the demographic situation, they decided to achieve its goal by expanding the boundaries of the city.

The city of Tomsk can be represented as point on the plane with coordinates $(0; 0)$. The city is surrounded with n other locations. Each location has coordinates (x_i, y_i) with the population of k_i people. You can widen the city boundaries to a circle of radius r . In such a case, all locations inside the circle and on its border are included into the city.

Your goal is to write a program that will determine the minimum radius r , to which is necessary to expand the boundaries of the city so that it becomes a megacity.

Input

The first line of the input contains two integers n and s ($1 \leq n \leq 10^3$; $1 \leq s < 10^6$) — the number of locations around Tomsk and the population of the city. Then n lines follow. The i -th line contains three integers — the x_i and y_i coordinate values of the i -th location and the number k_i of people in it ($1 \leq k_i < 10^6$). Each coordinate is an integer and doesn't exceed 10^4 in its absolute value.

It is guaranteed that no two locations are at the same point and no location is at point $(0; 0)$.

Output

In the output, print `-1` (without the quotes), if Tomsk won't be able to become a megacity. Otherwise, in the first line print the minimum radius of the circle that the city needs to expand to in order to become a megacity.

The answer is considered correct if the absolute or relative error don't exceed 10^{-6} .

Examples

input

```
4 999998
1 1 1
2 2 1
3 3 1
2 -2 1
```

output

```
2.8284271
```

input

```
4 999998
1 1 2
2 2 1
3 3 1
2 -2 1
```

output

```
1.4142136
```

input

```
2 1
1 1 999997
2 2 1
```

output

```
-1
```



```

8  int main(){
9      std::ios_base::sync_with_stdio(false);
10
11     size_t n;
12     cin >> n;
13     int pop;
14     cin >> pop;
15
16     vector<pair<double, uint64_t>> cities;
17     cities.reserve(n);
18
19     for(size_t i = 0; i < n; i++) {
20         int x, y, population;
21         cin >> x >> y >> population;
22         double distance = sqrt(x*x + y*y);
23         cities.emplace_back(distance, population);
24     }
25
26     sort(cities.begin(),
27          cities.end(),
28          [](const pair<double, uint64_t> & a, const pair<double, uint64_t> & b) {
29              return a.first < b.first;
30          }
31     );
32
33     for(auto &a : cities) {
34         pop += a.second;
35         if(pop >= 1000000) {
36             cout.precision(8);
37             cout << a.first << endl;
38             return 0;
39         }
40     }
41
42     cout << -1 << endl;
43     return 0;
44 }

```

Exercises

4. Two Heaps

<http://codeforces.com/problemset/problem/353/B?locale=en>

Valera has $2 \cdot n$ cubes, each cube contains an integer from 10 to 99. He arbitrarily chooses n cubes and puts them in the first heap. The remaining cubes form the second heap.

Valera decided to play with cubes. During the game he takes a cube from the first heap and writes down the number it has. Then he takes a cube from the second heap and write out its two digits near two digits he had written (to the right of them). In the end he obtained a single fourdigit integer — the first two digits of it is written on the cube from the first heap, and the second two digits of it is written on the second cube from the second heap.

Valera knows arithmetic very well. So, he can easily count the number of distinct fourdigit numbers he can get in the game. The other question is: how to split cubes into two heaps so that this number (the number of distinct fourdigit integers Valera can get) will be as large as possible?

Input

The first line contains integer n ($1 \leq n \leq 100$). The second line contains $2 \cdot n$ space-separated integers a_i ($10 \leq a_i \leq 99$), denoting the numbers on the cubes.

Output

In the first line print a single number — the maximum possible number of distinct four-digit numbers Valera can obtain. In the second line print $2 \cdot n$ numbers b_i ($1 \leq b_i \leq 2$). The numbers mean: the i -th cube should be put to the b_i -th heap in your division.

Examples

input

1
10 99

output

1
2 1

input

2
13 24 13 45

output

4
1 2 2 1

Exercises

5. Find Pair

<http://codeforces.com/problemset/problem/160/C?locale=en>

You've got another problem dealing with arrays. Let's consider an arbitrary sequence containing n (not necessarily different) integers a_1, a_2, \dots, a_n . We are interested in all possible pairs of numbers (a_i, a_j) , $(1 \leq i, j \leq n)$. In other words, let's consider all n^2 pairs of numbers, picked from the given array.

For example, in sequence $a = \{3, 1, 5\}$ are 9 pairs of numbers:
(3, 3), (3, 1), (3, 5), (1, 3), (1, 1), (1, 5), (5, 3), (5, 1), (5, 5).

Let's sort all resulting pairs lexicographically by non-decreasing. Let us remind you that pair (p_1, q_1) is *lexicographically less* than pair (p_2, q_2) only if either $p_1 < p_2$, or $p_1 = p_2$ and $q_1 < q_2$.

Then the sequence, mentioned above, will be sorted like that:
(1, 1), (1, 3), (1, 5), (3, 1), (3, 3), (3, 5), (5, 1), (5, 3), (5, 5)

Let's number all the pair in the sorted list from 1 to n^2 . Your task is formulated like this: you should find the k -th pair in the ordered list of all possible pairs of the array you've been given.

Input

The first line contains two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq n^2$). The second line contains the array containing n integers a_1, a_2, \dots, a_n ($-10^9 \leq a_i \leq 10^9$). The numbers in the array can coincide. All numbers are separated with spaces.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout`, streams or the `%I64d` specifier instead.

Output

In the single line print two numbers — the sought k -th pair.

Examples

input

2 4
2 1

output

2 2

input

3 2
3 1 5

output

1 3