#include <iostream>

#include <tuple>

#include <vector>

std::pair<std::string, int> linear\_search(int arr[], int size, int key) {

int count = 0;

for (int i = 0; i < size; i++) {

count++;

if (key == arr[i]) {

return std::make\_tuple("Present", count);

}

}

return std::make\_tuple("Not Present", count);

}

int main() {

std::pair<std::string, int> pair;

std::vector<std::pair<std::string, int> > vec;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int arr[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

arr[j] = element;

}

int key;

std::cin >> key;

pair = linear\_search(arr, no\_of\_elements, key);

vec.push\_back(pair);

}

for (auto v : vec) {

std::cout << v.first << " " << v.second << "\n";

}

return 0;

}

OUTPUT :

3

8

34 35 65 31 25 89 64 30

89

5

977 354 244 546 355

244

6

23 64 13 67 43 56

63

Present 6

Present 3

Not Present 6

#include <iostream>

#include <tuple>

#include <vector>

std::pair<std::string, int> binary\_search(int arr[], int lower\_bound, int upper\_bound, int key) {

int count = 0;

while (lower\_bound <= upper\_bound) {

int mid = (lower\_bound + upper\_bound) / 2;

count++;

if (arr[mid] == key) {

return std::make\_tuple("Present", count);

} else if (arr[mid] < key)

lower\_bound = mid + 1;

else if (arr[mid] > key)

upper\_bound = mid - 1;

}

return std::make\_tuple("Not Present", count);

}

int main() {

std::pair<std::string, int> pair;

std::vector<std::pair<std::string, int> > vec;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int arr[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

arr[j] = element;

}

int key;

std::cin >> key;

pair = binary\_search(arr, 0, no\_of\_elements - 1, key);

vec.push\_back(pair);

}

for (auto v : vec) {

std::cout << v.first << " " << v.second << "\n";

}

return 0;

}

OUTPUT:

3

5

12 23 36 39 41

41

8

21 39 40 45 51 54 68 72

69

10

101 246 438 561 796 896 899 4644 7999 8545

7999

Present 3

Not Present 4

Present 3

#include <cmath>

#include <iostream>

int main() {

int n, key, ctr = 1, i = 0, j, flag = 0, t;

std::cin >> t;

while (t--) {

std::cin >> n;

int a[n];

for (i = 0; i < n; i++) std::cin >> a[i];

i = 0;

std::cin >> key;

if (a[0] == key)

std::cout << "Present " << ctr << "\n";

else {

while (1) {

if (a[(int)pow(2, i)] > key) {

for (j = pow(2, i); j < std::min((int)pow(2, i + 1), n); j++) {

ctr++;

if (a[j] == key) flag = 1;

}

if (flag)

std::cout << "Present " << ctr << "\n";

else

std::cout << "Not Present " << ctr << "\n";

break;

}

ctr++;

i++;

}

}

}

return 0;

}

OUTPUT :

3

5

12 23 36 39 41

41

8

21 39 40 45 51 54 68 72

69

10

101 246 438 561 796 896 899 4644 7999 8545

7999

Present 3

Not Present 4

Present 3

#include <iostream>

#include <vector>

int binary\_search(int arr[], int lower\_bound, int upper\_bound, int key) {

while (lower\_bound <= upper\_bound) {

int mid = (lower\_bound + upper\_bound) / 2;

if (key == arr[mid])

return mid;

else if (key > arr[mid])

lower\_bound = mid + 1;

else if (key < arr[mid])

upper\_bound = mid - 1;

}

return -1;

}

int duplicate\_count(int arr[], int index, int size, int key) {

int left\_count = 0, right\_count = 0;

int i = index, j = index;

while (key == arr[i] && i >= 0) {

left\_count++;

i--;

}

while (key == arr[j] && j < size) {

right\_count++;

j++;

}

return left\_count + right\_count - 1;

}

int main() {

int n;

std::cin >> n;

std::pair<int, int> pair;

std::vector<std::pair<int, int> > vec;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int arr[no\_of\_elements];

for (int i = 0; i < no\_of\_elements; i++) {

int element;

std::cin >> element;

arr[i] = element;

}

int duplicates = 0;

int key;

std::cin >> key;

int index = binary\_search(arr, 0, no\_of\_elements - 1, key);

if (index != -1)

duplicates = duplicate\_count(arr, index, no\_of\_elements, key);

pair = std::make\_tuple(key, duplicates);

vec.push\_back(pair);

}

for (auto v : vec) {

if (v.second == 0) {

std::cout << "key not present"

<< " " << v.second << "\n";

} else

std::cout << v.first << " " << v.second << "\n";

}

return 0;

}

OUTPUT :

2

10

235 235 278 278 763 764 790 853 981 981

981

15

1 2 2 3 3 5 5 5 25 75 75 75 97 97 97

75

981 2

75 3

#include <iostream>

#include <vector>

int difference(int arr[], int size, int key) {

int count = 0;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if (arr[i] - arr[j] == key)

count++;

}

}

return count;

}

int main() {

std::vector<int> vec;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

array[j] = element;

}

int key;

std::cin >> key;

int count = difference(array, no\_of\_elements, key);

vec.push\_back(count);

}

for (auto v : vec) {

if (v == 0) {

std::cout << "No pair found."

<< "\n";

} else

std::cout << v << "\n";

}

return 0;

}

OUTPUT :

3

5

1 5 84 209 341

10

24 28 48 71 86 89 92 120 194 201

15

64 69 82 95 99 107 113 141 171 350 369 400 511 590 666

No sequence found.

2, 7, 8

1, 6, 9

#include <iostream>

#include <vector>

std::tuple<int, int, int> sum(int arr[], int size) {

for (int i = 0; i < size - 2; i++) {

for (int j = i + 1; j < size - 1; j++) {

for (int k = i + 2; k < size; k++) {

if (arr[i] + arr[j] == arr[k])

return std::make\_tuple(i + 1, j + 1, k + 1);

}

}

}

return std::make\_tuple(-1, -1, -1);

}

int main() {

std::tuple<int, int, int> tuple;

std::vector<std::tuple<int, int, int> > vec;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

array[j] = element;

}

tuple = sum(array, no\_of\_elements);

vec.push\_back(tuple);

}

for (auto v : vec) {

if (std::get<0>(v) == -1) {

std::cout << "No sequence found."

<< "\n";

} else

std::cout << std::get<0>(v) << ", " << std::get<1>(v) << ", " << std::get<2>(v) << "\n";

}

return 0;

}

OUTPUT:

2

5

1 51 84 21 31

20

10

24 71 16 92 12 28 48 14 20 22

4

2

4

#include <iostream>

#include <tuple>

std::tuple<int, int> insertion\_sort(int array[], int size) {

int shifts = 0, comparisons = 0, temp = 0;

for (int i = 1; i < size; i++) {

temp = array[i];

int j = i - 1;

while (j >= 0 && array[j] > temp) {

array[j + 1] = array[j];

j--;

shifts++;

comparisons++;

}

comparisons++;

array[j + 1] = temp;

}

return std::make\_tuple(comparisons, shifts);

}

int main() {

std::tuple<int, int> tuple;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

array[j] = element;

}

tuple = insertion\_sort(array, no\_of\_elements);

for (int z = 0; z < no\_of\_elements; z++)

std::cout << array[z] << " ";

std::cout << "\ncomparisons = " << std::get<0>(tuple) << "\n";

std::cout << "shifts = " << std::get<1>(tuple) << "\n";

}

return 0;

}

OUTPUT :

3

8

-23 65 -31 76 46 89 45 32

-31 -23 32 45 46 65 76 89

comparisons = 20

shifts = 13

10

54 65 34 76 78 97 46 32 51 21

21 32 34 46 51 54 65 76 78 97

comparisons = 37

shifts = 28

15

63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325

-12 22 31 42 46 54 63 65 86 223 324 325 553 645 652

comparisons = 68

shifts = 54

#include <iostream>

std::tuple<int, int> selection\_sort(int array[], int size) {

int min = 0, min\_element\_position = 0;

int swaps = 0, comparisons = 0;

for (int i = 0; i < size - 1; i++) {

min = array[i];

min\_element\_position = i;

for (int j = i + 1; j < size; j++) {

if (array[j] < min) {

min = array[j];

min\_element\_position = j;

}

comparisons++; // comparisons = n \* (n - 1) / 2

}

array[min\_element\_position] = array[i];

array[i] = min;

swaps++; // swaps = n - 1

}

return std::make\_tuple(comparisons, swaps);

}

int main() {

std::tuple<int, int> tuple;

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

array[j] = element;

}

tuple = selection\_sort(array, no\_of\_elements);

for (int k = 0; k < no\_of\_elements; k++) {

std::cout << array[k] << " ";

}

std::cout << "\ncomparisons = " << std::get<0>(tuple) << "\nswaps = " << std::get<1>(tuple) << "\n";

}

return 0;

}

OUTPUT:

3

8

-13 65 -21 76 46 89 45 12

-21 -13 12 45 46 65 76 89

comparisons = 28

swaps = 7

10

54 65 34 76 78 97 46 32 51 21

21 32 34 46 51 54 65 76 78 97

comparisons = 45

swaps = 9

15

63 42 223 645 652 31 324 22 553 12 54 65 86 46 325

12 22 31 42 46 54 63 65 86 223 324 325 553 645 652

comparisons = 105

swaps = 14

#include <iostream>

void merge(int array[], const int low, const int mid, const int high) {

int i, j, k;

int n1 = mid - low + 1;

int n2 = high - mid;

int leftArray[n1];

int rightArray[n2];

for (i = 0; i < n1; i++)

leftArray[i] = array[low + i];

for (j = 0; j < n2; j++)

rightArray[j] = array[mid + 1 + j];

i = 0;

j = 0;

k = low;

while (i < n1 && j < n2) {

if (leftArray[i] <= rightArray[j]) {

array[k++] = leftArray[i++];

} else

array[k++] = rightArray[j++];

}

while (i < n1) {

array[k++] = leftArray[i++];

}

while (j < n2) {

array[k++] = rightArray[j++];

}

}

void merge\_sort(int array[], const int low, const int high) {

if (low >= high) return;

int mid = low + (high - low) / 2;

merge\_sort(array, low, mid);

merge\_sort(array, mid + 1, high);

merge(array, low, mid, high);

}

bool has\_duplicate(int array[], int size) {

for (int i = 0; i < size - 1; i++) {

if (array[i] == array[i + 1]) return true;

}

return false;

}

int main() {

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements, element;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

std::cin >> element;

array[j] = element;

}

merge\_sort(array, 0, no\_of\_elements - 1);

has\_duplicate(array, no\_of\_elements) ? std::cout << "YES\n" : std::cout << "NO\n";

}

return 0;

}

OUTPUT :

3

5

28 52 83 14 75

NO

10

75 65 1 65 2 6 86 2 75 8

YES

15

75 35 86 57 98 23 73 1 64 8 11 90 61 19 20

NO

#include <iostream>

int main() {

int n;

std::cin >> n;

for (int z = 0; z < n; z++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) std::cin >> array[j];

int k;

std::cin >> k;

std::sort(array, array + no\_of\_elements);

int i = 0, count = 0;

while (count != k - 1) {

if (array[i] != array[i + 1]) count++;

i++;

}

std::cout << array[i] << "\n\n";

}

return 0;

}

OUTPUT :

3

5

23 65 21 76 46 89 45 32

COMPARISONS = 16

10

54 65 34 76 78 97 46 32 51 21

COMPARISONS = 22

15

63 42 223 645 21 324 22 553 12 54 65 86 46 325

COMPARISONS = 43

#include <iostream>

int comparisons = 0;

void merge(int array[], const int low, const int mid, const int high) {

int i, j, k;

int n1 = mid - low + 1;

int n2 = high - mid;

int leftArray[n1];

int rightArray[n2];

for (i = 0; i < n1; i++)

leftArray[i] = array[low + i];

for (j = 0; j < n2; j++)

rightArray[j] = array[mid + 1 + j];

i = 0;

j = 0;

k = low;

while (i < n1 && j < n2) {

if (leftArray[i] <= rightArray[j]) {

array[k++] = leftArray[i++];

} else

array[k++] = rightArray[j++];

comparisons++;

}

while (i < n1) {

array[k++] = leftArray[i++];

}

while (j < n2) {

array[k++] = rightArray[j++];

}

}

void merge\_sort(int array[], const int low, const int high) {

if (low >= high) return;

int mid = low + (high - low) / 2;

merge\_sort(array, low, mid);

merge\_sort(array, mid + 1, high);

merge(array, low, mid, high);

}

int main() {

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) std::cin >> array[j];

merge\_sort(array, 0, no\_of\_elements - 1);

std::cout << "Comparisons = " << comparisons << "\n";

comparisons = 0;

}

return 0;

}

OUTPUT :

3

8

23 65 21 76 46 89 45 32

COMPARISONS = 14

SWAPS = 10

10

54 65 34 76 78 97 46 32 51 21

COMPARISONS = 29

SWAPS = 21

15

63 42 223 645 652 31 324 22 553 12 54 65 86 46 325

COMPARISONS = 45

SWAPS = 39

// #include <time.h>

#include <iostream>

int total\_swaps = 0;

int comparisons = 0;

void swap(int& a, int& b) {

int t = a;

a = b;

b = t;

}

int partition(int array[], const int startIndex, const int endIndex) {

// srand(time(NULL));

// int random\_index = rand() % 15;

// array[endIndex] = array[random\_index];

int pivot = array[endIndex];

int i = startIndex - 1;

for (int j = startIndex; j < endIndex; j++) {

if (array[j] < pivot) {

i++;

swap(array[i], array[j]);

total\_swaps++;

}

comparisons++;

}

swap(array[i + 1], array[endIndex]);

total\_swaps++;

return i + 1;

}

void quick\_sort(int array[], const int startIndex, const int endIndex) {

if (startIndex >= endIndex) return;

int pi = partition(array, startIndex, endIndex);

quick\_sort(array, startIndex, pi - 1);

quick\_sort(array, pi + 1, endIndex);

}

int main() {

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) std::cin >> array[j];

quick\_sort(array, 0, no\_of\_elements - 1);

std::cout << "Comparisons = " << comparisons << "\nSwaps = " << total\_swaps;

std::cout << "\n";

comparisons = 0;

total\_swaps = 0;

}

return 0;

}

OUTPUTS :

2

10

123 656 54 765 344 514 765 34 765 234

3

123

15

43 64 13 78 864 346 786 456 21 19 8 434 76 270 601

8

78

#include <iostream>

int highest\_frequency(int array[], const int& arr\_size) {

int max = array[0], index = 0;

for (int i = 0; i < arr\_size; i++) {

if (max < array[i]) {

max = array[i];

index = i;

}

}

return index;

}

int main() {

const int total\_alphabets = 26;

char alphabet\_array[total\_alphabets];

std::memset(alphabet\_array, ' ', sizeof(alphabet\_array));

for (int i = 0; i < total\_alphabets; i++) alphabet\_array[i] = char(i + 97);

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

char array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

char element;

std::cin >> element;

array[j] = element;

}

int count[total\_alphabets];

std::memset(count, 0, sizeof(count));

for (int z = 0; z < no\_of\_elements; z++) {

count[alphabet\_array[array[z] - 97] - 97]++;

} /\* this loop is the solution part of the question \*/

int highest\_frequency\_index = highest\_frequency(count, total\_alphabets);

if (count[highest\_frequency\_index] == 1)

std::cout << "No Duplicates Present.\n";

else

std::cout << alphabet\_array[highest\_frequency\_index] << " - " << count[highest\_frequency\_index] << "\n";

}

return 0;

}

OUTPUT:

3

10

a e d w a d q a f p

a – 3

15

r k p g v y u m q a d j c z e

No Duplicates Present .

20

g t l l t c w a w g l c w d s a a v c l

1 - 4

#include <iostream>

bool two\_sum(int array[], int left, int right, const int key) {

bool flag = false;

while (left <= right) {

int temp\_sum = array[left] + array[right];

if (temp\_sum == key) {

flag = true;

std::cout << "(" << array[left] << ", " << array[right] << ") ";

if (array[left] == array[left + 1] && array[right] != array[right - 1])

left++;

else if (array[left] != array[left + 1] && array[right] == array[right - 1])

right--;

else {

left++;

right--;

}

} else if (temp\_sum > key)

right--;

else

left++;

}

return flag;

}

int main() {

int n;

std::cin >> n;

for (int i = 0; i < n; i++) {

int no\_of\_elements;

std::cin >> no\_of\_elements;

int array[no\_of\_elements];

for (int j = 0; j < no\_of\_elements; j++) {

int element;

std::cin >> element;

array[j] = element;

}

std::sort(array, array + no\_of\_elements);

int key;

std::cin >> key;

bool pair\_exist = two\_sum(array, 0, no\_of\_elements - 1, key);

if (!pair\_exist) std::cout << "No such pair exist.";

std::cout << "\n";

}

return 0;

}

OUTPUT :

2

10

64 28 97 40 12 72 84 24 38 10

50

(10, 40) ( 12, 38)

15

56 10 72 91 29 3 41 45 61 20 11 39 9 12 94

302

No such pair exist.

#include <iostream>

int main() {

int m, n, i, j, total, k = 0, x;

std::cin >> m;

int a[m];

for (i = 0; i < m; i++) std::cin >> a[i];

std::cin >> n;

int b[n], c[n];

for (i = 0; i < n; i++) std::cin >> b[i];

std::sort(a, a + m);

std::sort(b, b + n);

for (i = 0; i < m; i++) {

for (j = 0; j < n; j++) {

if (a[i] == b[j]) {

total = 0;

for (x = 0; x < k; x++)

if (a[i] == c[x]) total++;

if (total == 0) {

c[k] = a[i];

k++;

}

}

}

}

for (i = 0; i < k; i++) std::cout << c[i] << " ";

return 0;

}

OUTPUT:

7

10 10 34 39 55 76 85

12

10 10 11 30 30 34 51 55 69 72 89

10 34 55