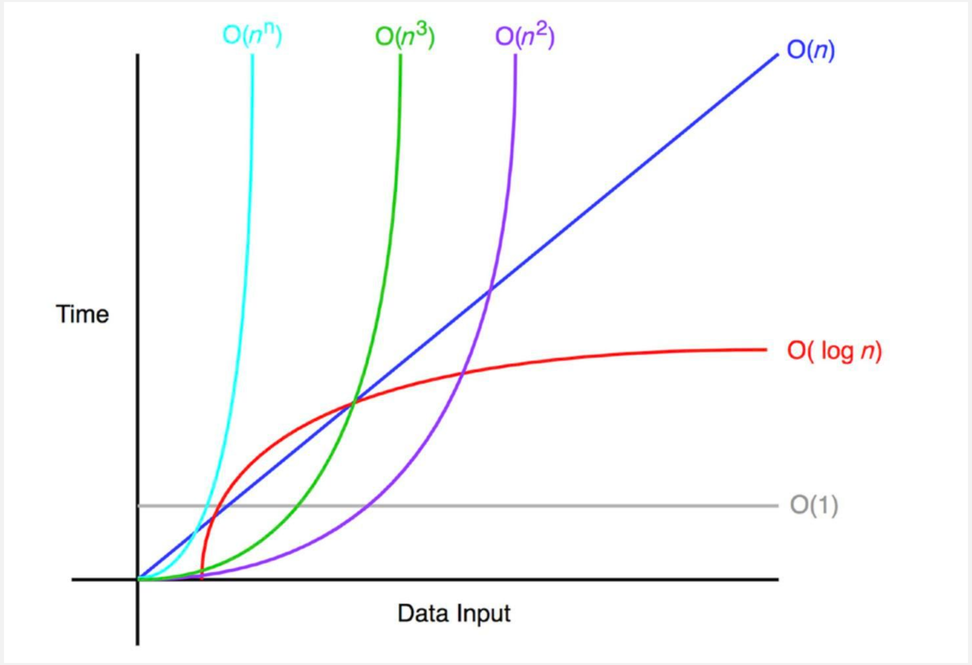
Competitive Programming

PEDRO PAREDES

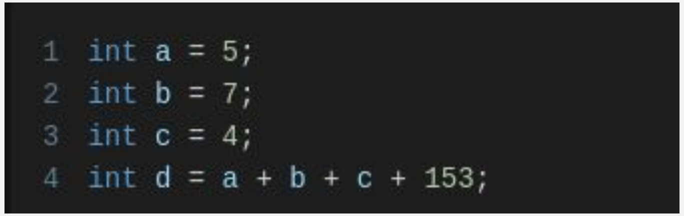
Time Complexity and Sorting

Time Complexity

Crash course in Big O notation

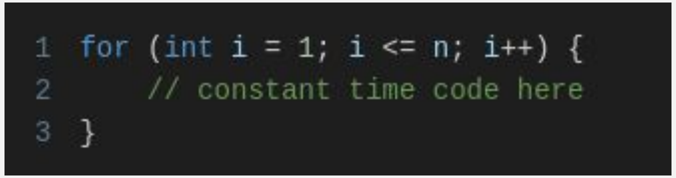


Examples

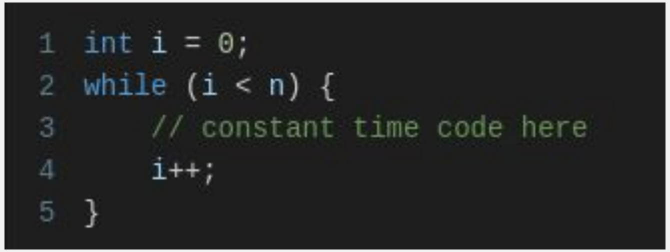


```
int a = 5;
int b = 7;
int c = 4;
int d = a + b + c + 153;
```

This is O(1)

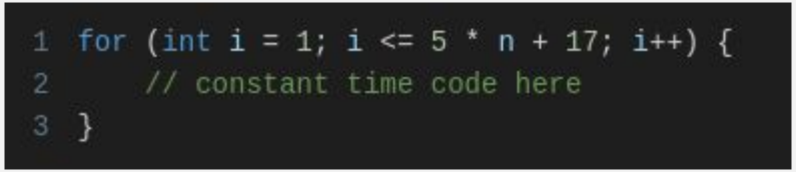


```
for (int i = 1; i <= n; i++) {
    // constant time code here
}
```

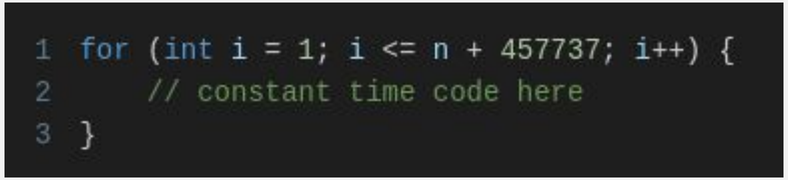


```
int i = 0;
while (i < n) {
    // constant time code here
    i++;
}
```

This is O(n)



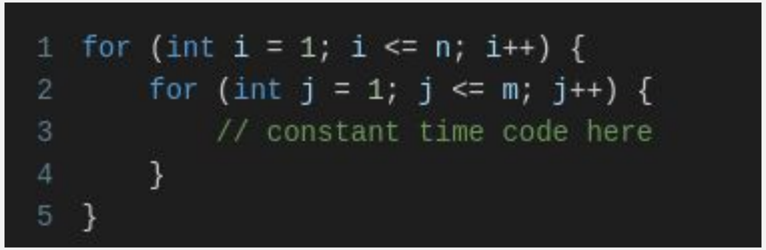
```
for (int i = 1; i <= 5 * n + 17; i++) {
    // constant time code here
}
```



```
for (int i = 1; i <= n + 457737; i++) {
    // constant time code here
}
```

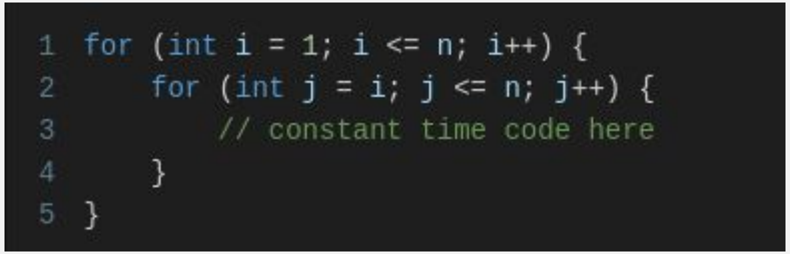
Still O(n)

More examples



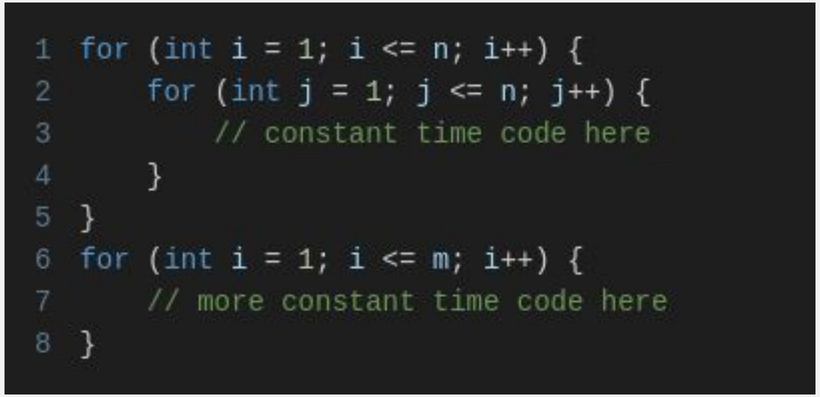
```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
        // constant time code here
    }
}
```

O(nm)



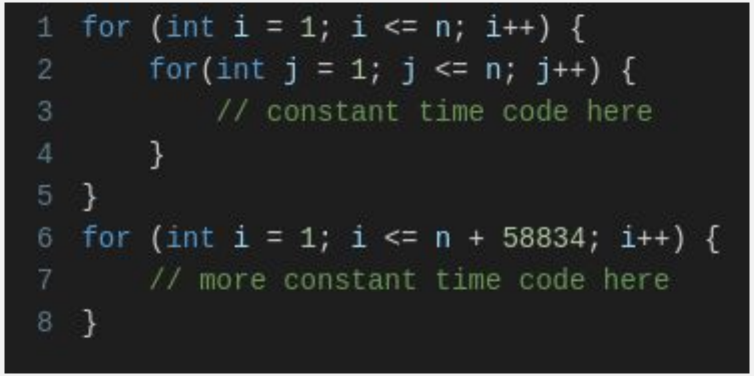
```
for (int i = 1; i <= n; i++) {
    for (int j = i; j <= n; j++) {
        // constant time code here
    }
}
```

$O(n^2)$



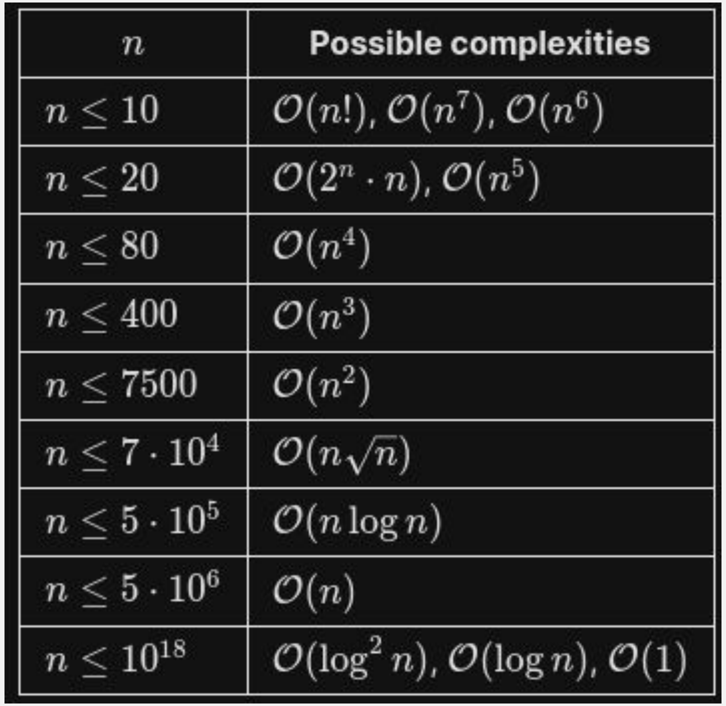
```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        // constant time code here
    }
}
for (int i = 1; i <= m; i++) {
    // more constant time code here
}
```

$O(n^2 + m)$



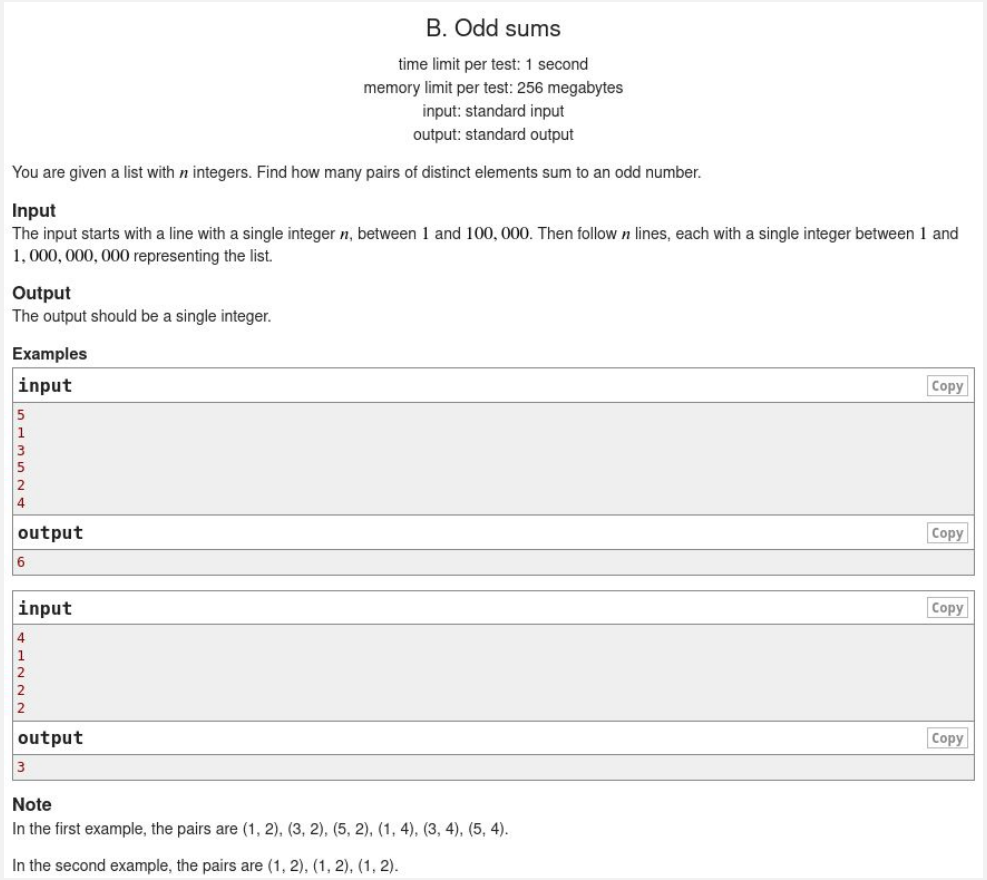
```
for (int i = 1; i <= n; i++) {
    for(int j = 1; j <= n; j++) {
        // constant time code here
    }
}
for (int i = 1; i <= n + 58834; i++) {
    // more constant time code here
}
```

Time/Big O table for competitive programming



n	Possible complexities
$n \leq 10$	$\mathcal{O}(n!)$, $\mathcal{O}(n^7)$, $\mathcal{O}(n^6)$
$n \leq 20$	$\mathcal{O}(2^n \cdot n)$, $\mathcal{O}(n^5)$
$n \leq 80$	$\mathcal{O}(n^4)$
$n \leq 400$	$\mathcal{O}(n^3)$
$n \leq 7500$	$\mathcal{O}(n^2)$
$n \leq 7 \cdot 10^4$	$\mathcal{O}(n\sqrt{n})$
$n \leq 5 \cdot 10^5$	$\mathcal{O}(n \log n)$
$n \leq 5 \cdot 10^6$	$\mathcal{O}(n)$
$n \leq 10^{18}$	$\mathcal{O}(\log^2 n)$, $\mathcal{O}(\log n)$, $\mathcal{O}(1)$

Example problem: odd sum



B. Odd sums

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a list with n integers. Find how many pairs of distinct elements sum to an odd number.

Input

The input starts with a line with a single integer n, between 1 and $100,000$. Then follow n lines, each with a single integer between 1 and $1,000,000,000$ representing the list.

Output

The output should be a single integer.

Examples

input

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

output

```
6
```

input

```
4
1
2
2
2
```

output

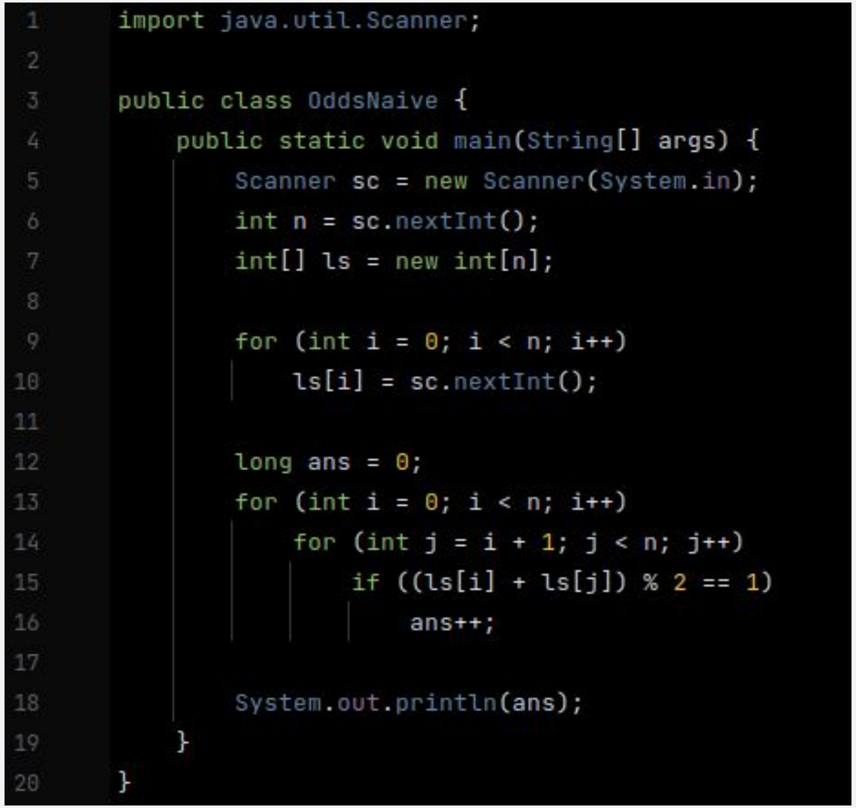
```
3
```

Note

In the first example, the pairs are (1, 2), (3, 2), (5, 2), (1, 4), (3, 4), (5, 4).

In the second example, the pairs are (1, 2), (1, 2), (1, 2).

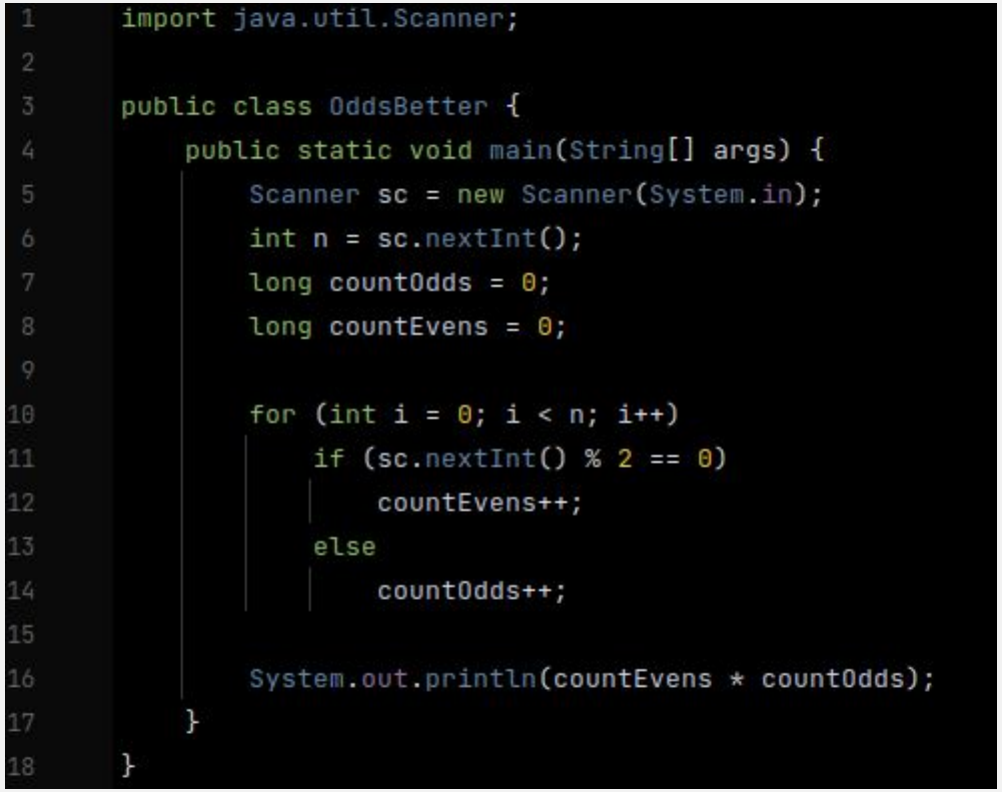
Naive solution



```java
import java.util.Scanner;

public class OddsNaive {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] ls = new int[n];

        for (int i = 0; i < n; i++)
            ls[i] = sc.nextInt();

        long ans = 0;
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)
                if ((ls[i] + ls[j]) % 2 == 1)
                    ans++;

        System.out.println(ans);
    }
}
```

This is $O(n^2)$

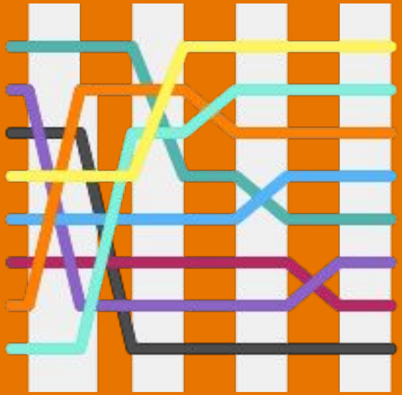
Better solution



```java
import java.util.Scanner;

public class OddsBetter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long countOdds = 0;
        long countEvens = 0;

        for (int i = 0; i < n; i++)
            if (sc.nextInt() % 2 == 0)
                countEvens++;
            else
                countOdds++;

        System.out.println(countEvens * countOdds);
    }
}
```

This is O(n)

Sorting



Know how to fast sort in your favorite language!!

C++

Use:

sort(arr, arr + N)

where ***N*** is the number of elements to be sorted

You will need to import **<algorithm>** and this is part of the stl namespace, so you need to:

using namespace std;

Java

Use:

Arrays.sort(arr)

You need to import ***java.util.Arrays***

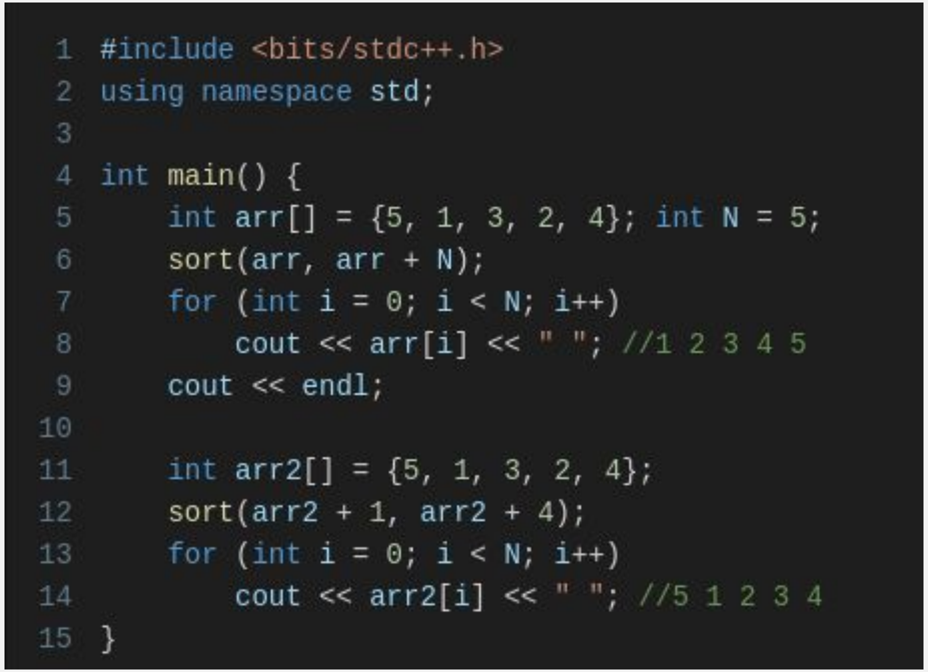
Arrays.sort() function uses quicksort on primitive data types such as longs, which can be slow! Use Integer instead of int to avoid this

Python

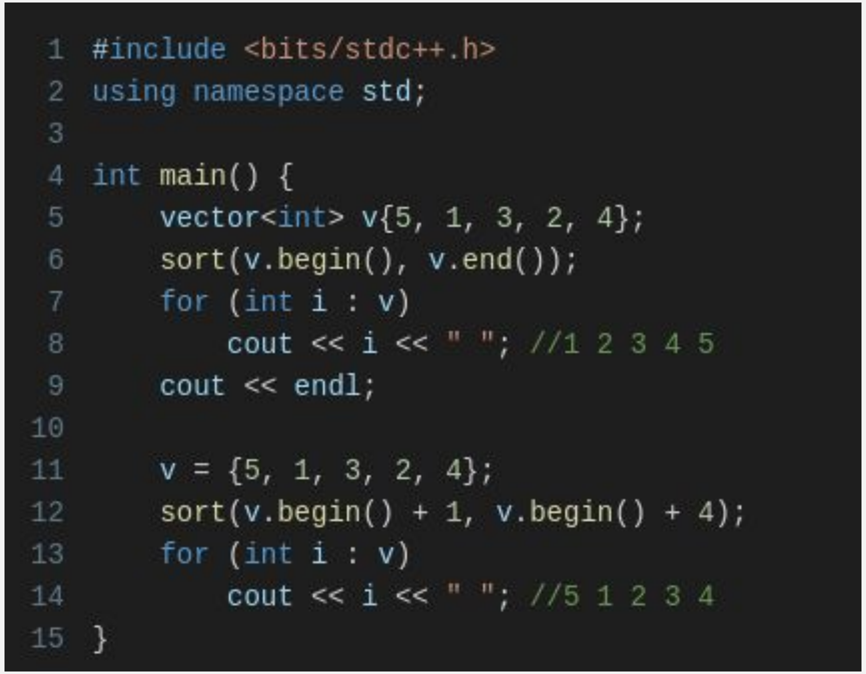
Use:

arr.sort()

Two examples in C++

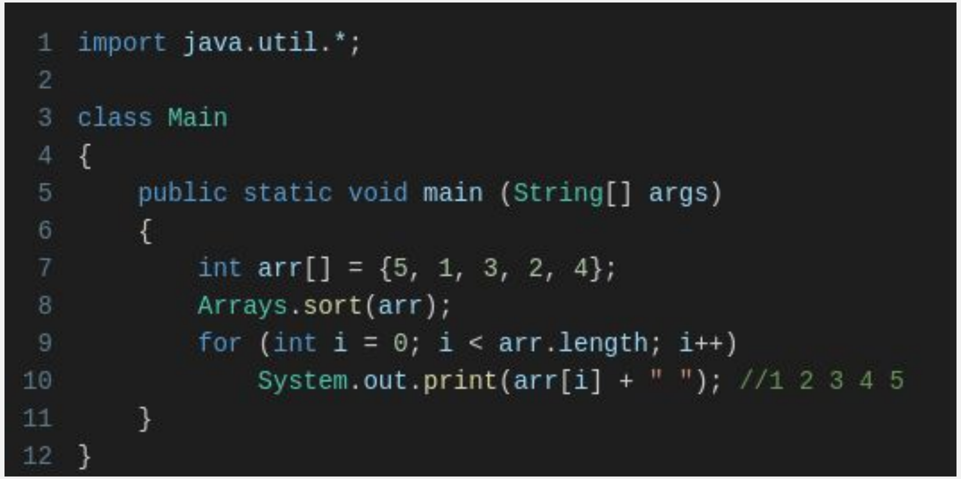


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int arr[] = {5, 1, 3, 2, 4}; int N = 5;
    sort(arr, arr + N);
    for (int i = 0; i < N; i++)
        cout << arr[i] << " "; //1 2 3 4 5
    cout << endl;

    int arr2[] = {5, 1, 3, 2, 4};
    sort(arr2 + 1, arr2 + 4);
    for (int i = 0; i < N; i++)
        cout << arr2[i] << " "; //5 1 2 3 4
}
```

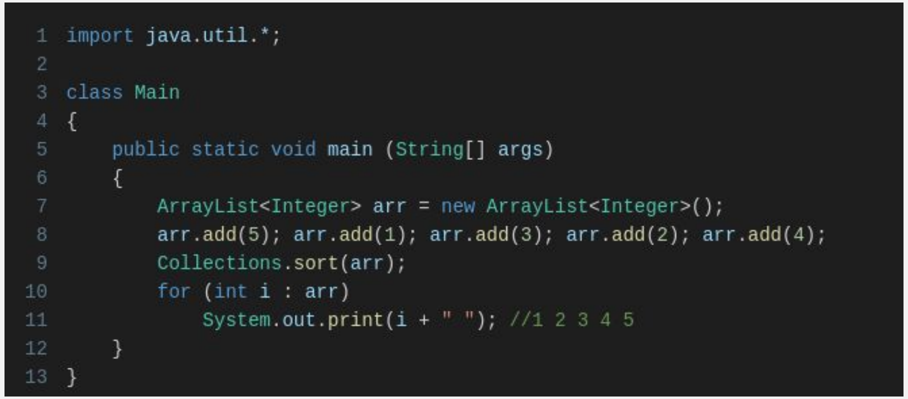


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<int> v{5, 1, 3, 2, 4};
    sort(v.begin(), v.end());
    for (int i : v)
        cout << i << " "; //1 2 3 4 5
    cout << endl;

    v = {5, 1, 3, 2, 4};
    sort(v.begin() + 1, v.begin() + 4);
    for (int i : v)
        cout << i << " "; //5 1 2 3 4
}
```

Two examples in Java



```java
import java.util.*;

class Main
{
    public static void main (String[] args)
    {
        int arr[] = {5, 1, 3, 2, 4};
        Arrays.sort(arr);
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " "); //1 2 3 4 5
    }
}
```



```java
import java.util.*;

class Main
{
    public static void main (String[] args)
    {
        ArrayList<Integer> arr = new ArrayList<Integer>();
        arr.add(5); arr.add(1); arr.add(3); arr.add(2); arr.add(4);
        Collections.sort(arr);
        for (int i : arr)
            System.out.print(i + " "); //1 2 3 4 5
    }
}
```

Sources:

Most of the code shown and the time table were taken from: *https://usaco.guide/*