# Competitive Programming

Sorting

# Sorting

- Solving a problem might require multiple steps
  - Often, these will depend on familiar data structures an algorithms
  - E.g., sort
- Need to be able to quickly apply existing, efficient implementations
- An O( n log n ) sort should be available in just about any language you expect to use.

# Sort in Python

```
#!/usr/bin/python3
import sys

n = int( input() )

seq = [ int( input() ) for x in range( n ) ]

seq.sort()
```

# Sort in Java

```
import java.util.*;

…

  public static void main( String[] args ) throws Exception {
    Scanner input = new Scanner( System.in );

    int n = input.nextInt();
    int[] seq = new int [ n ];
    for ( int i = 0; i < n; i++ )
      seq[ i ] = input.nextInt();

    Arrays.sort( seq );
```

# Sort in C++

```
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  int n;
  cin >> n;

  vector< int > seq( n );
  for ( int &x : seq )
    cin >> x;

  sort( seq.begin(), seq.end() );
```

# Choosing an Order

- You will want to be able to control the sort order.
- … while avoiding writing a sort routine yourself.
- You can just write your own comparison function … usually.

# Choosing an order in Python

```
#!/usr/bin/python3
import sys

def keyFunc( a ):
    return -a

n = int( input() )

seq = [ int( input() ) for x in range( n ) ]

seq.sort( key=keyFunc )
```

# Choosing an order in Python

```
#!/usr/bin/python3
import sys

n = int( input() )

seq = [ int( input() ) for x in range( n ) ]

seq.sort( key=lambda x: -x )
```

# One more in Python

```
#!/usr/bin/python3
import sys
import functools

def compFunc( a, b ):
    if a > b:
        return -1
    if b > a:
        return 1
    return 0

n = int( input() )

seq = [ int( input() ) for x in range( n ) ]

seq.sort( key=functools.cmp_to_key( compFunc ) )
```

# Choosing an order in Java

```
import java.util.*;

…

  public static void main( String[] args ) throws Exception {
    Scanner input = new Scanner( System.in );

    int n = input.nextInt();
    ArrayList< Integer > seq = new ArrayList<>();
    for ( int i = 0; i < n; i++ )
      seq.add( input.nextInt() );

    seq.sort( new MyComp() );
```

# Choosing an order in Java

```
static class MyComp implements Comparator< Integer > {
  public int compare( Integer a, Integer b ) {
    if ( a > b )
      return -1;
    if ( a < b )
      return 1;
    return 0;
  }

  public boolean equals( Integer a, Integer b ) {
    return a == b;
  }
}
```

# Choosing an order in C++

```
#include <vector>
#include <algorithm>

using namespace std;

bool comp( int a, int b ) {
  return b < a;
}

int main() {
  int n;
  cin >> n;

  vector< int > seq( n );
  for ( int &x : seq )
    cin >> x;

  sort( seq.begin(), seq.end(), comp );
```