Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Selection Sort

Quick Recap of Relevant Topics

- Basic programming constructs
  - Iteration constructs
  - Functions
  - Arrays and matrices, among other things …
- The sorting problem
  - Motivation

Overview of This Lecture

- Selection sort
  - A simple, intuitive sorting technique

Quiz1, Quiz2 and Quiz3 Marks in CS101

Total
24
18
17
25
27
24

Rank all students in decreasing order of "Total" marks

Core problem:

Sort "Total" marks in decreasing order

Simplification:

If two marks are equal, any ordering between them ok

Quiz1, Quiz2 and Quiz3 Marks in CS101

IIT Bombay

Total
3 24
4 18
5 17
2 25
1 27
3 24

Rank all students in decreasing order of "Total" marks

Core problem:

Sort "Total" marks in decreasing order

Simplification:

If two marks are equal, any ordering between them ok

Quiz1, Quiz2 and Quiz3 Marks in CS101

IIT Bombay

	Total
1	27
2	25
3	24
3	24
4	18
5	17

Rank all students in decreasing order of "Total" marks

Core problem:

Sort "Total" marks in decreasing order

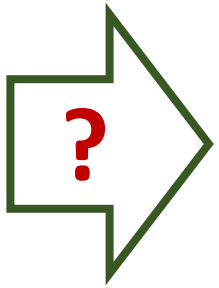
Simplification:

If two marks are equal, any ordering between them ok

How Do We Do It?

IIT Bombay

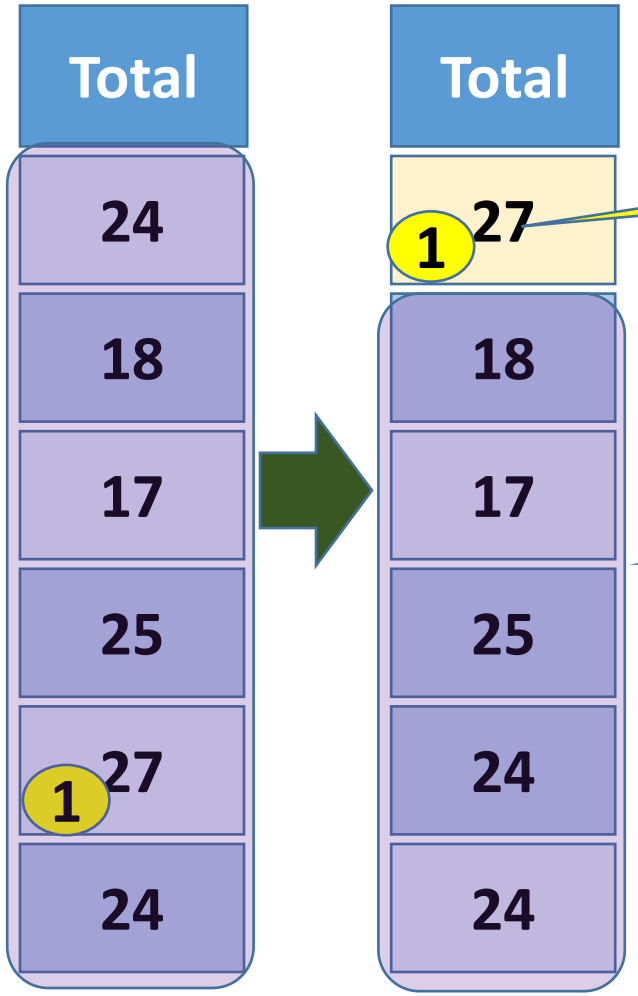
Total
24
18
17
25
27
24



Total
27
25
24
24
18
17

How Do We Do It?

IIT Bombay



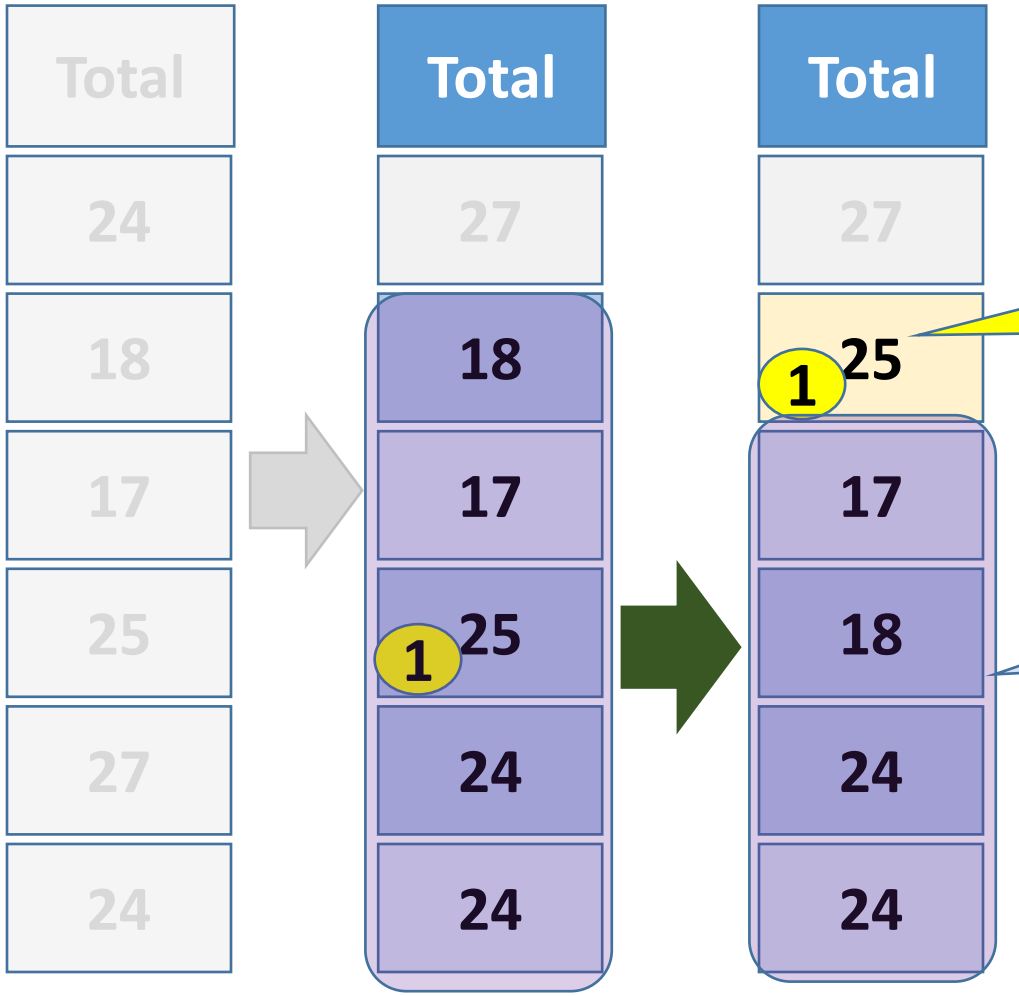
Get only the first element in its right place

Solve part of the problem

Arrive at a similar but "simpler" problem to solve:

Sort remaining marks

How Do We Do It?



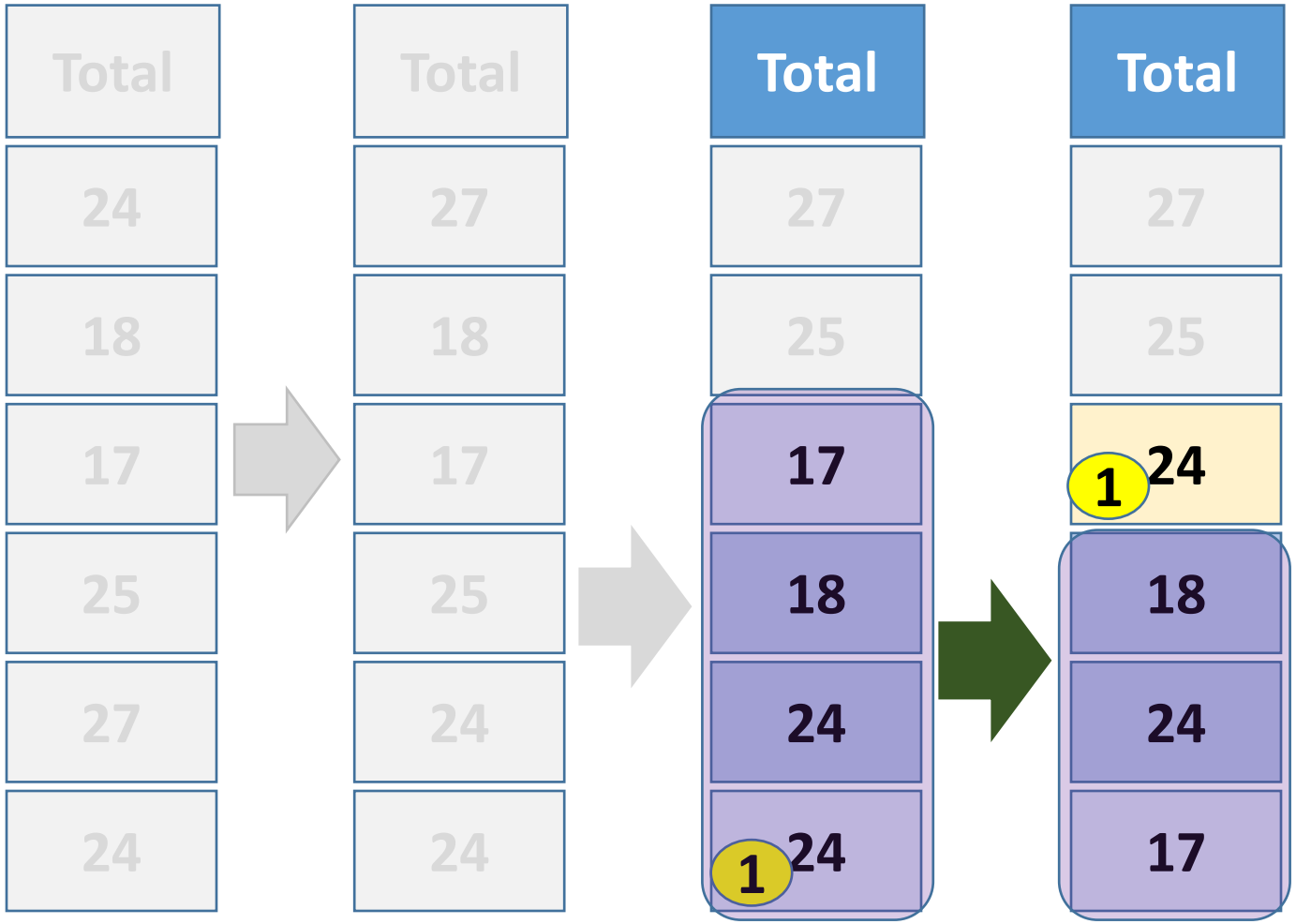
Solve part of the problem

Get only the first (in current subproblem) element in its right place

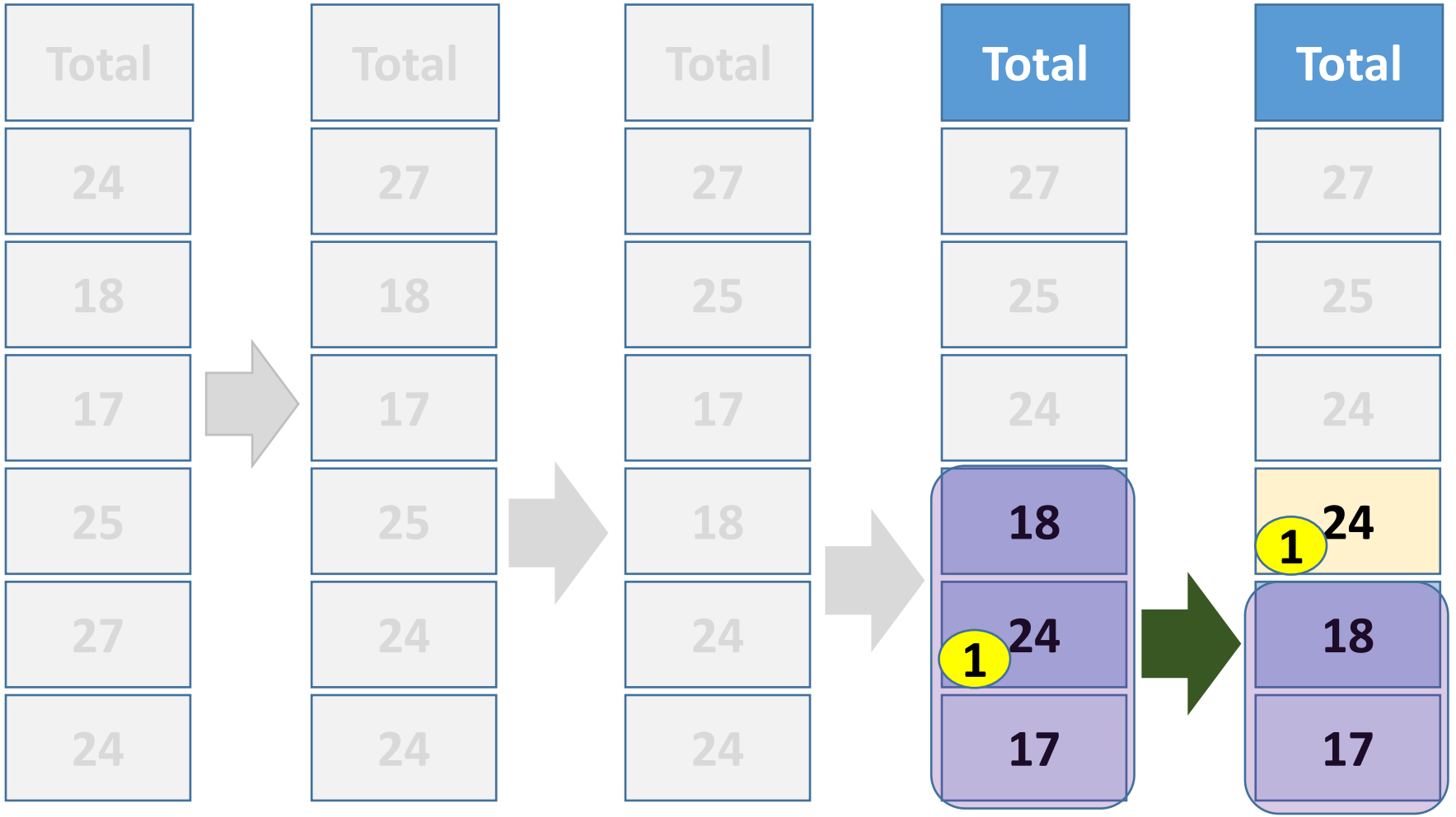
Arrive at a similar but "simpler" problem to solve:

Sort remaining marks

How Do We Do It?

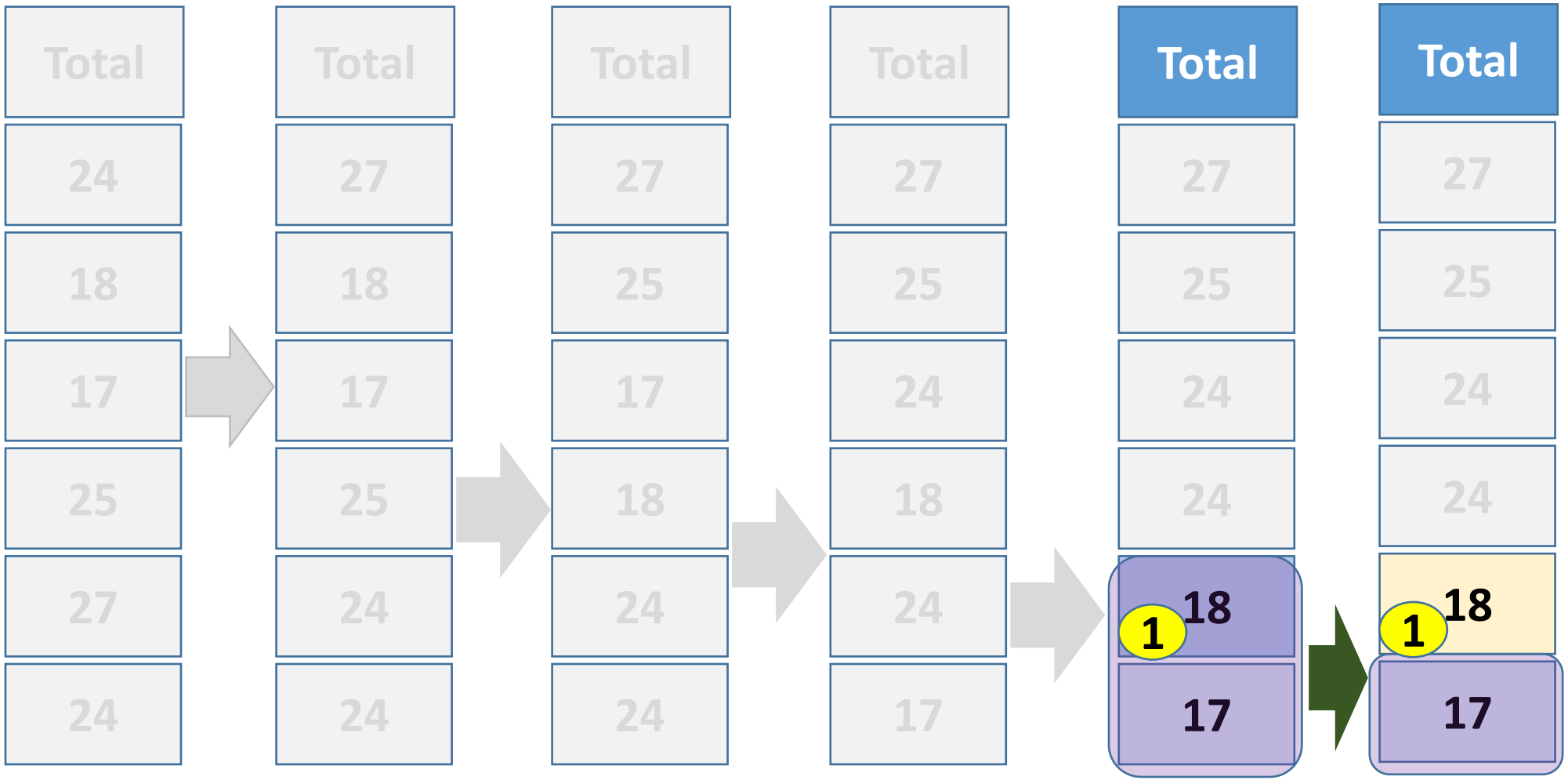


How Do We Do It?

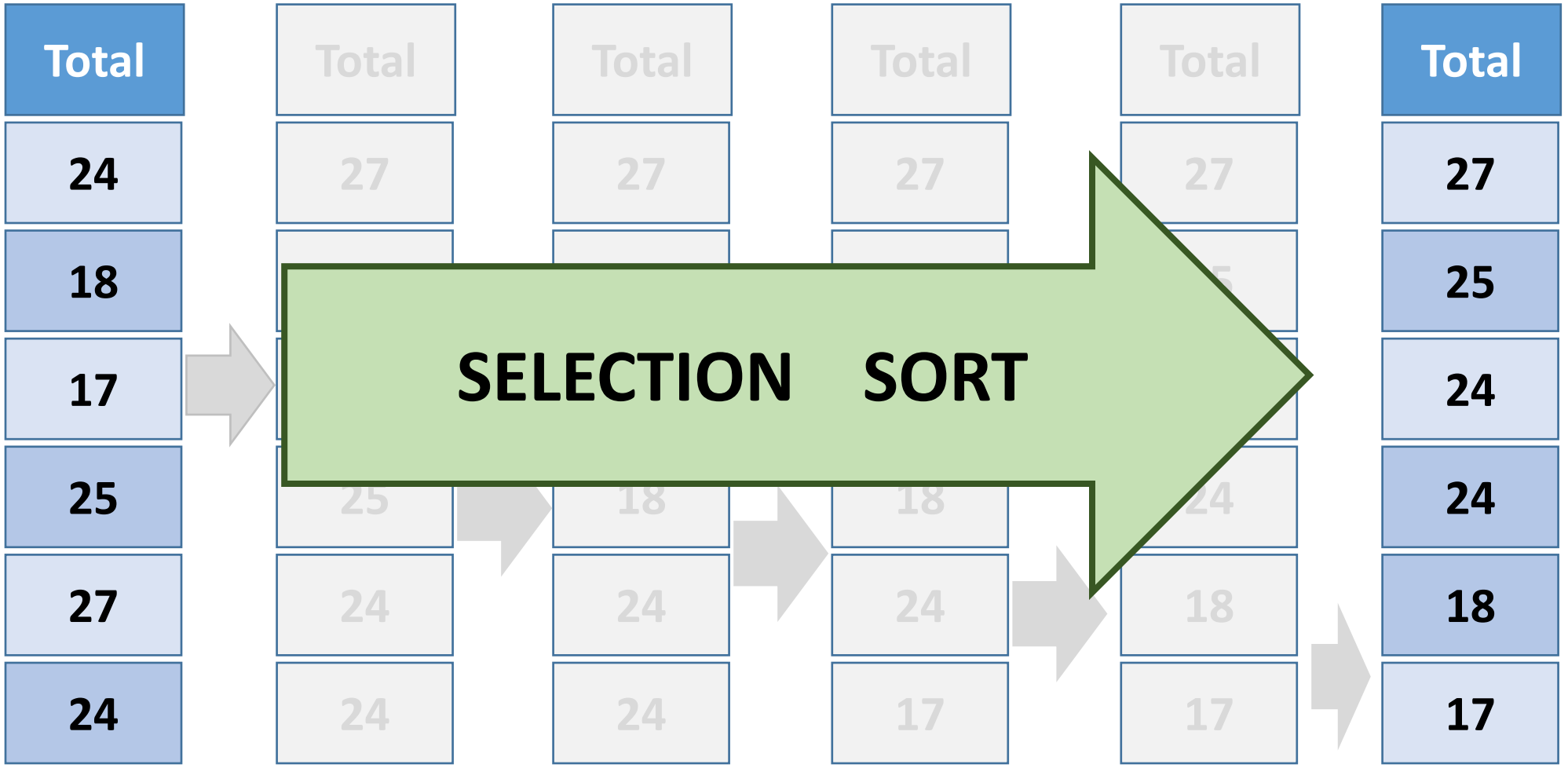


How Do We Do It?

IIT Bombay



How Do We Do It?



A C++ Program For Selection Sort

- Given an array A of n integers
  - Sort them in decreasing order

    $A[0] \geq A[1] \geq A[2] \geq \ldots A[n-1]$
  - If two elements are equal, either of them may be ordered before the other

[Once our program is written, final ordering among equal elements is of course completely determined]

Selection Sort in C++

IIT Bombay

```
int main() {
   int  n;
   cout << “Give number of integers to sort: “;  cin >> n;
  // Input validation
  if (n > 100) { cout << “Too many elements!” << endl; return -1;}
   if (n <= 0) {cout << “Invalid input!” << endl; return -1;}
   …. Rest of code …
   return 0;
}
```

Selection Sort in C++

IIT Bombay

```
int main() {
    … Declarations and input validation …
   int count, A[100];  // Array of integers to sort
   // Read integers to sort
   cout << “Give “ << n << “integers to sort.” << endl;
   for (count = 0; count < n; count++) { cin >> A[count]; }
    … Rest of code …
   return 0;
}
```

Selection Sort in C++

IIT Bombay

```
int main() {
    … Declarations, input validation and reading elements of array A …
   // Selection sort
   int currTop, currMaxIndex;  // A[currTop] … A[n-1] is unsorted array
   for (currTop = 0; currTop < n; currTop ++) {
      // Select maximum element in unsorted part of  array A
      // Let currMaxIndex be its index in array A
      // Swap A[currTop] and A[currMaxIndex]
   }
    … Rest of code …
   return 0;
}
```

Selection Sort in C++

IIT Bombay

```
int main() {
   … Declarations, input validation and reading elements of array A …
   // Selection sort
   int currTop, currMaxIndex; // A[currTop] … A[n-1] is unsorted array
   for (currTop = 0; currTop < n; currTop ++) {
      currMaxIndex = findIndexOfMax(A, currTop, n);
      swap(A, currTop, currMaxIndex);
   }
   … Rest of code …
   return 0;
}
```

Selection Sort in C++

IIT Bombay

```
// PRECONDITION: start < end
// start, end within array bounds of A
int  findIndexOfMax(int A[],  int start,  int end) {




}
// POSTCONDITION: A[currMaxIndex] at least as large as
// all elements in A[start] through A[end-1], no change in A
```

Array as parameter (not call-by-value)

Selection Sort in C++

IIT Bombay

```
int main() {
   … Declarations, input validation and reading elements of array A …
   // Selection sort
   int currTop, currMaxIndex; // A[currTop] … A[n-1] is unsorted array
   for (currTop = 0; currTop < n; currTop ++) {
       currMaxIndex = findIndexOfMax(A, currTop, n);
       swap(A, currTop, currMaxIndex);
   }
    … Rest of code …
   return 0;
}
```

Selection Sort in C++

IIT Bombay

```
// PRECONDITION: index1, index2 within
//                        bounds of A
void  swap(int A[],  int index1,  int index2) {




}
// POSTCONDITION: A[index1], A[index2] swapped
//                        Array A changed
```

Array as parameter (not call-by-value)

Role of Comparison Operator

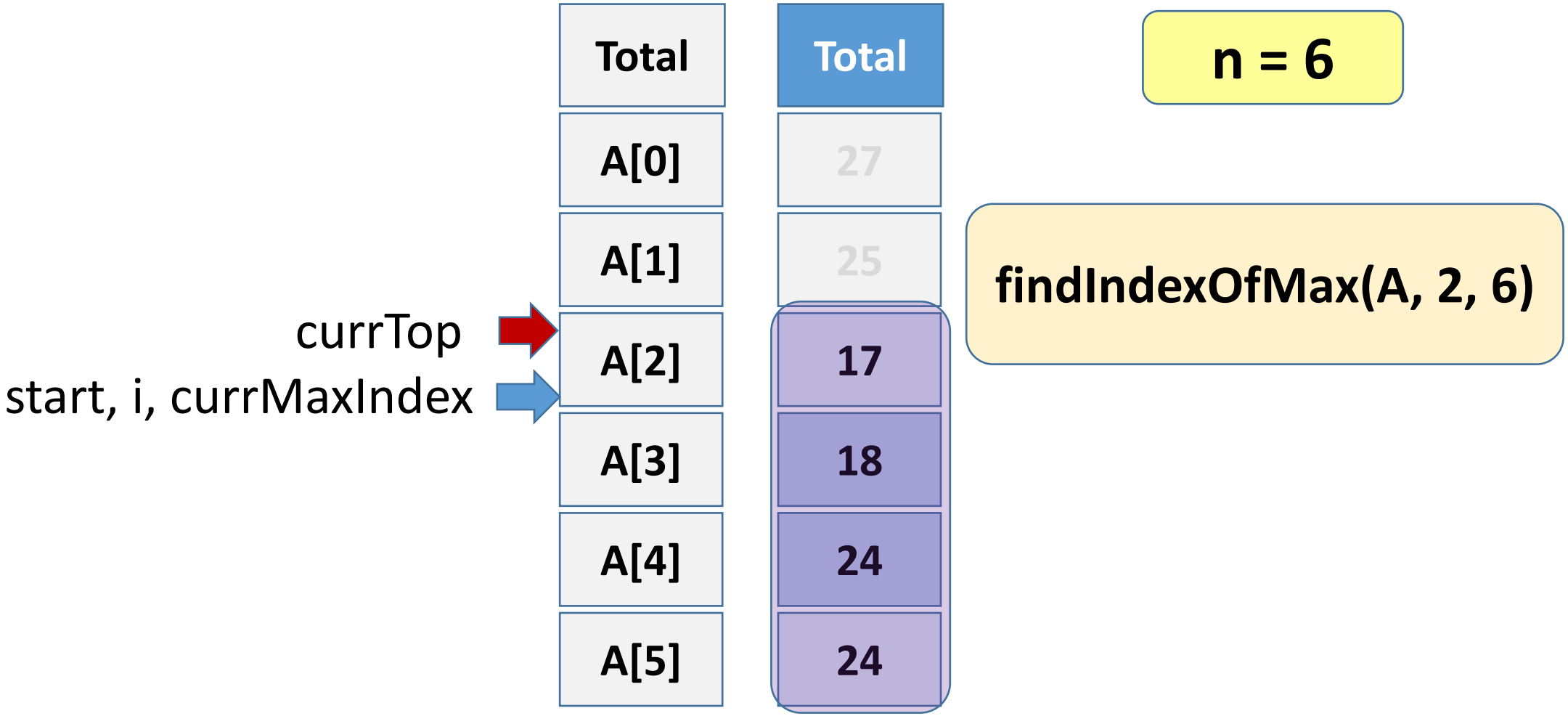
IIT Bombay

```
// PRECONDITION: start < end
// start, end within array bounds of A
int  findIndexOfMax(int A[],  int start,  int e
    int i, currMaxIndex = start;
    for ( i = start ; i < end; i++ ) {
      if (A[i] >= A[currMaxIndex]) { currMaxIndex = i; }
    }
   return currMaxIndex;
 }
// POSTCONDITION: A[currMaxIndex] at least as large as
// all elements in A[start] through A[end-1], no change in A
```

Note the use of ">="

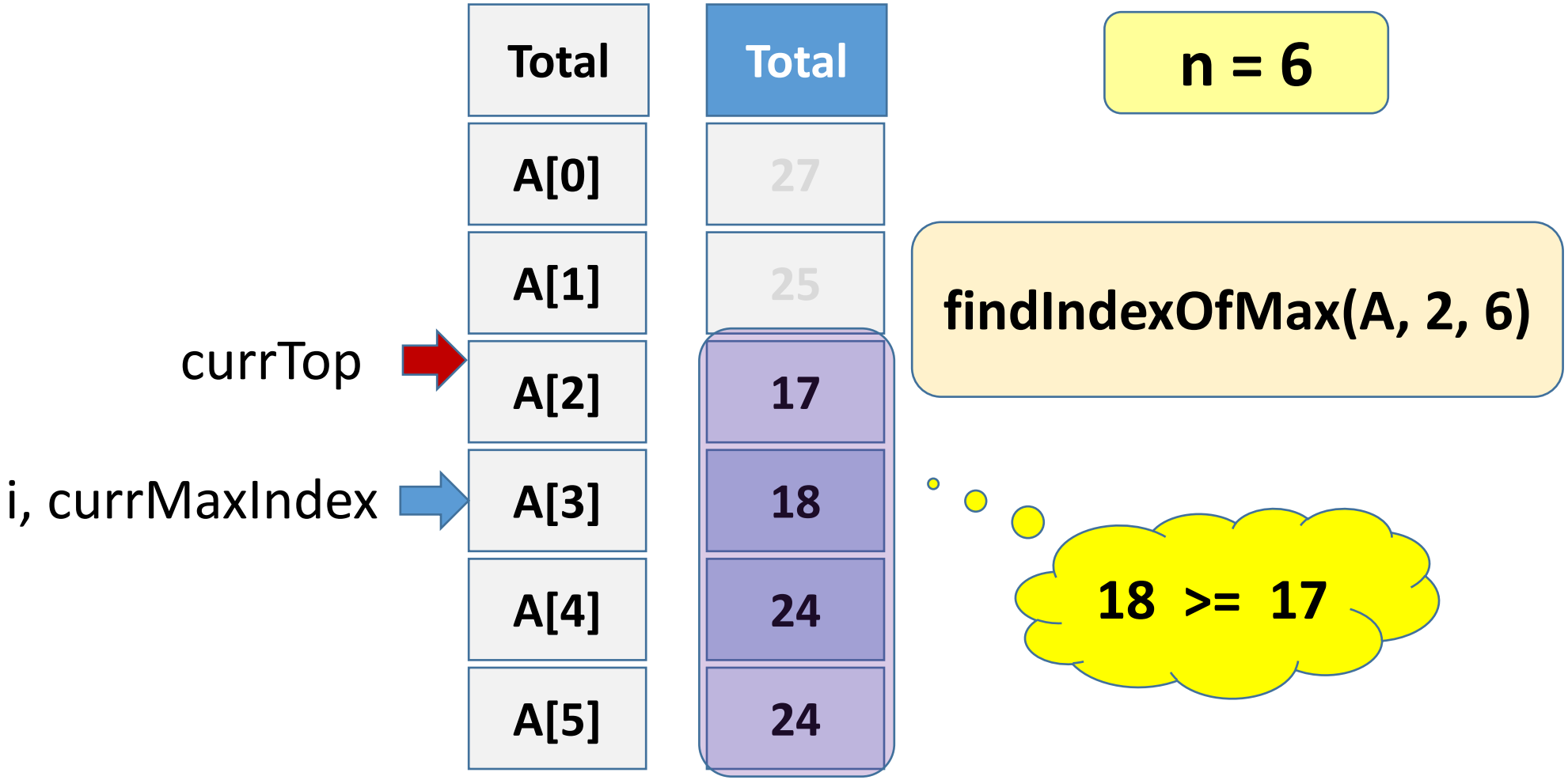
Selection Sort Using >=

IIT Bombay



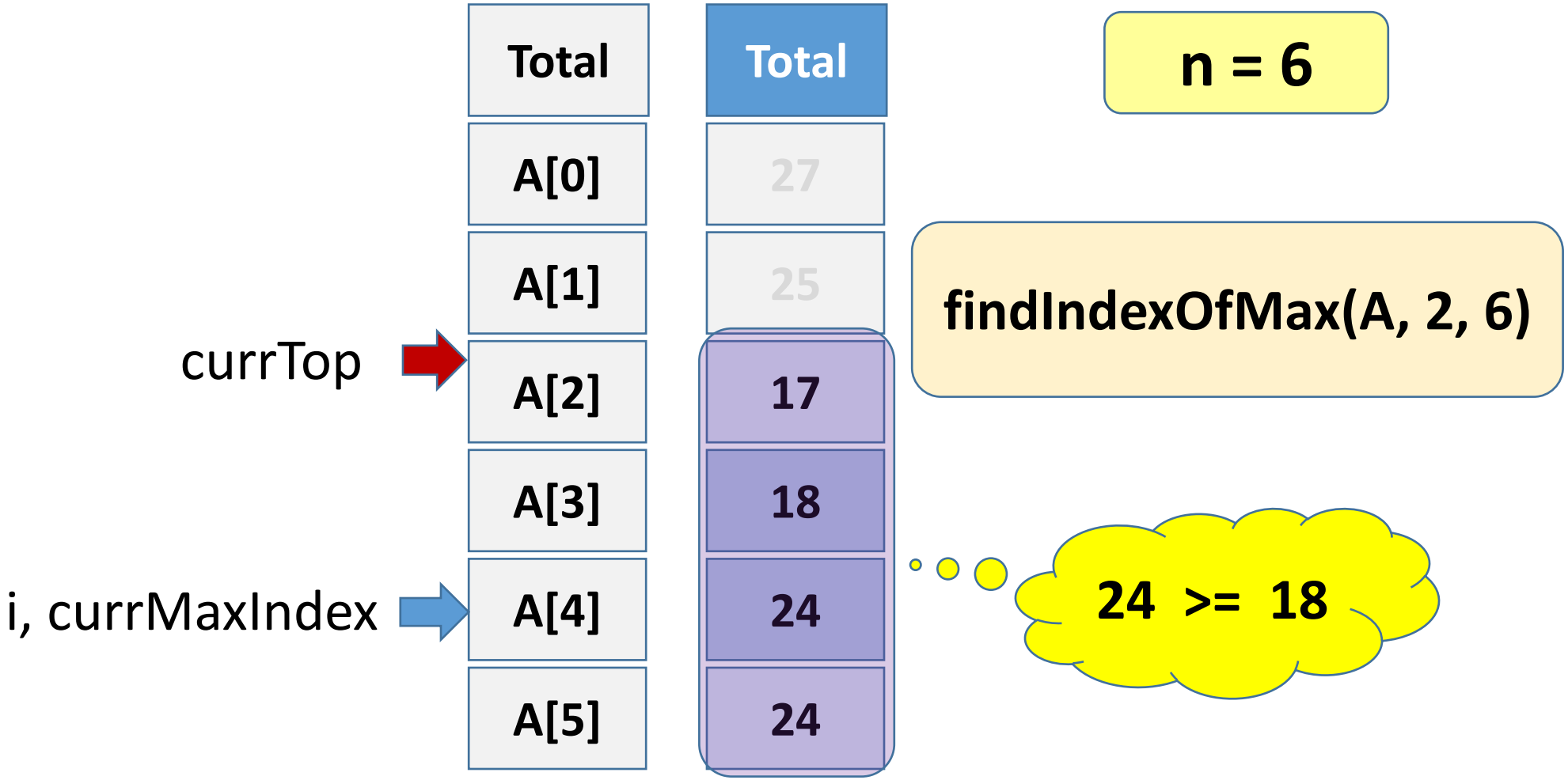
Selection Sort Using >=

IIT Bombay



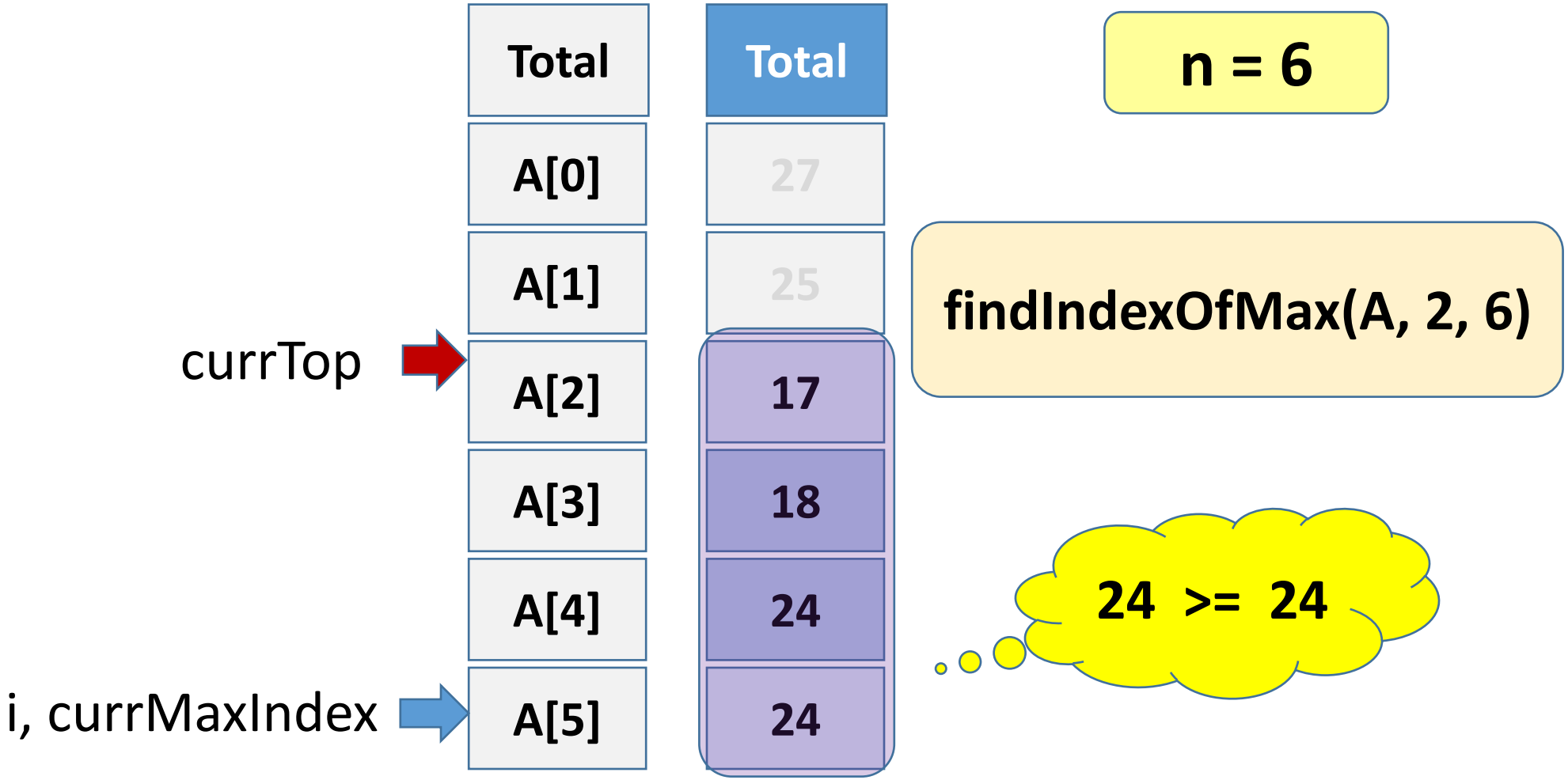
Selection Sort Using >=

IIT Bombay



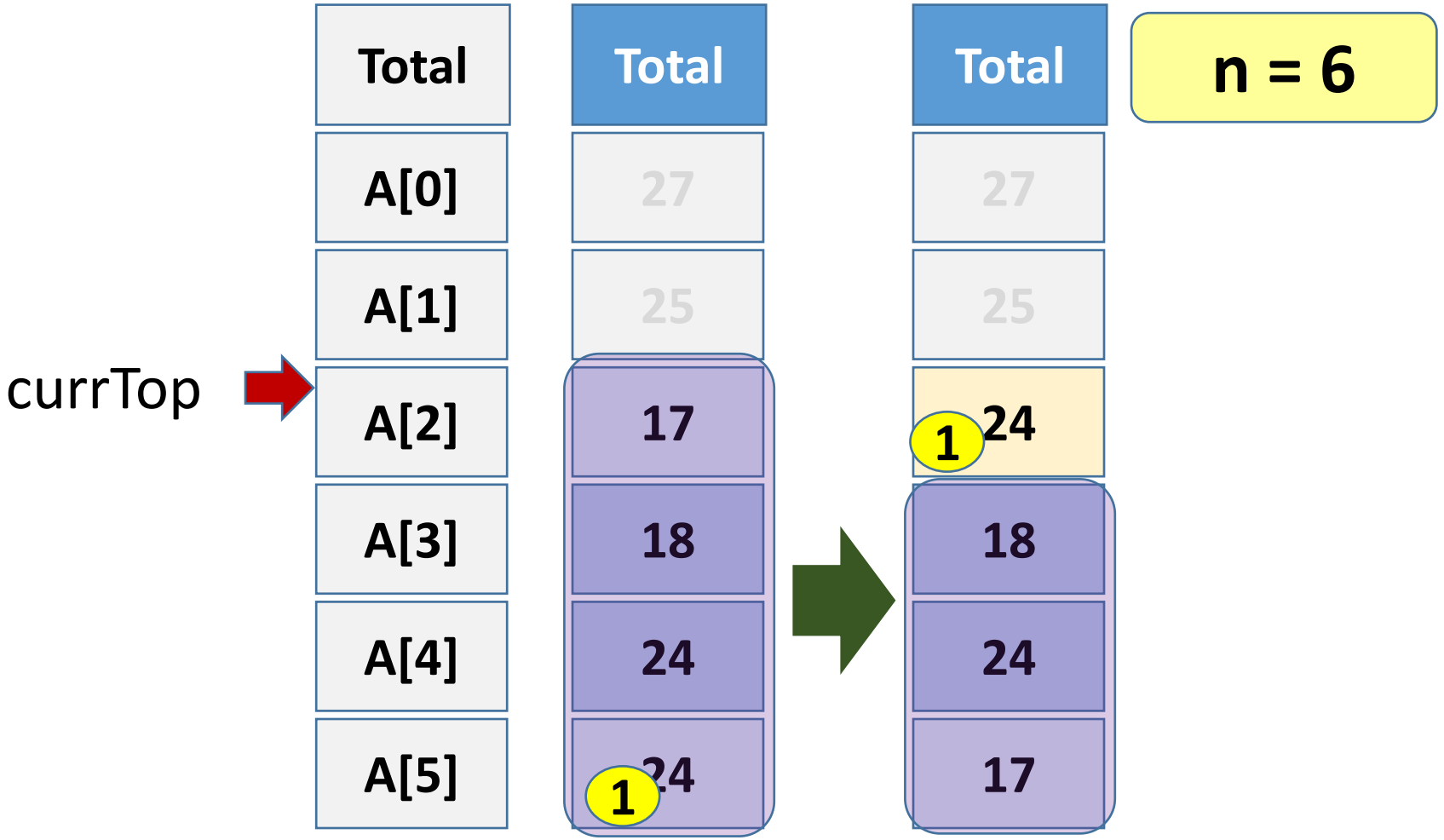
Selection Sort Using >=

IIT Bombay



Selection Sort Using >=

IIT Bombay



Role of Comparison Operator

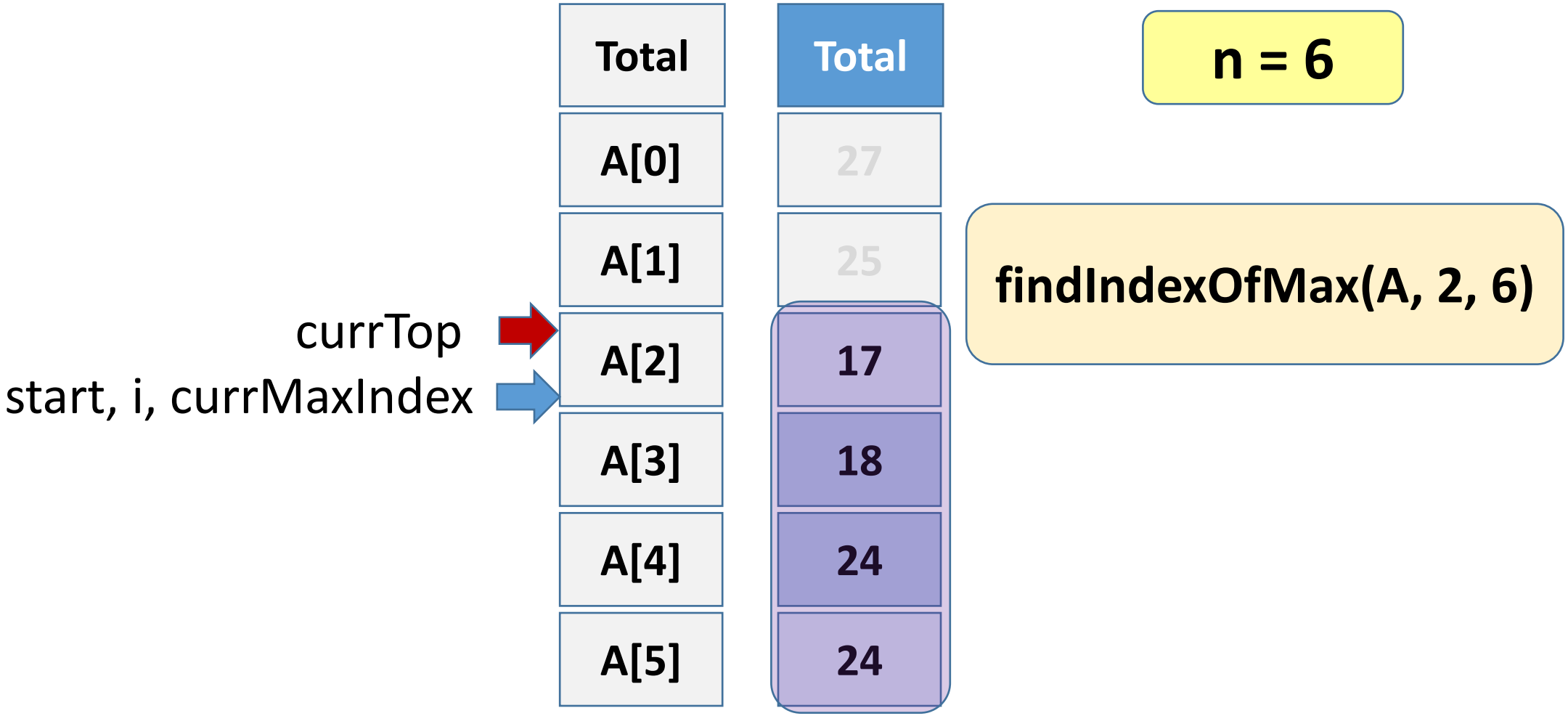
IIT Bombay

```
// PRECONDITION: start < end
// start, end within array bounds of A
int  findIndexOfMax(int A[],  int start
    int i, currMaxIndex = start;
    for ( i = start ; i < end; i++ ) {
      if (A[i]  > A[currMaxIndex]) { currMaxIndex = i; }
    }
   return currMaxIndex;
 }
// POSTCONDITION: A[currMaxIndex] at least as large as
// all elements in A[start] through A[end-1], no change in A
```

What if we used “>” ?

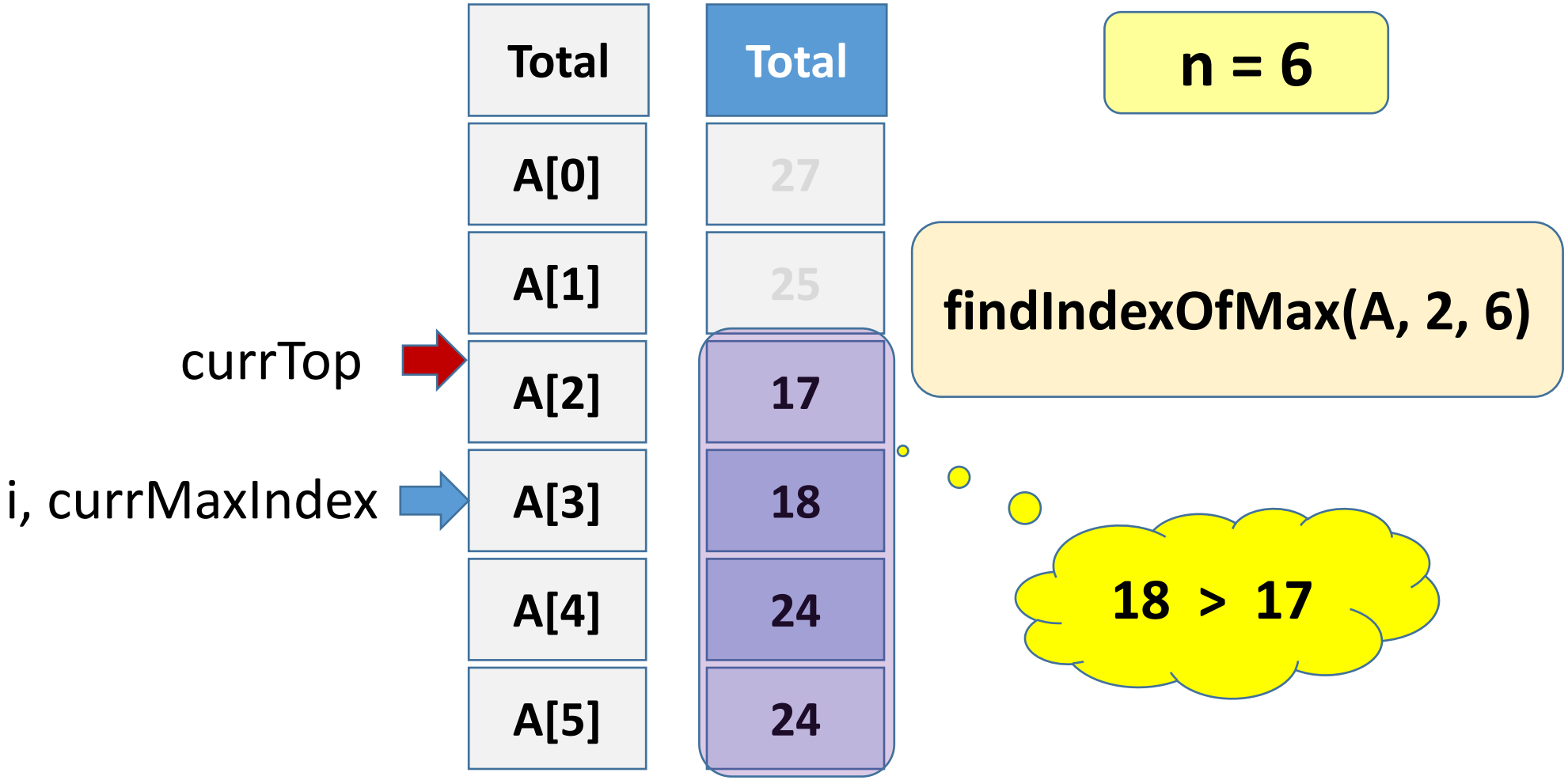
Selection Sort Using >

IIT Bombay



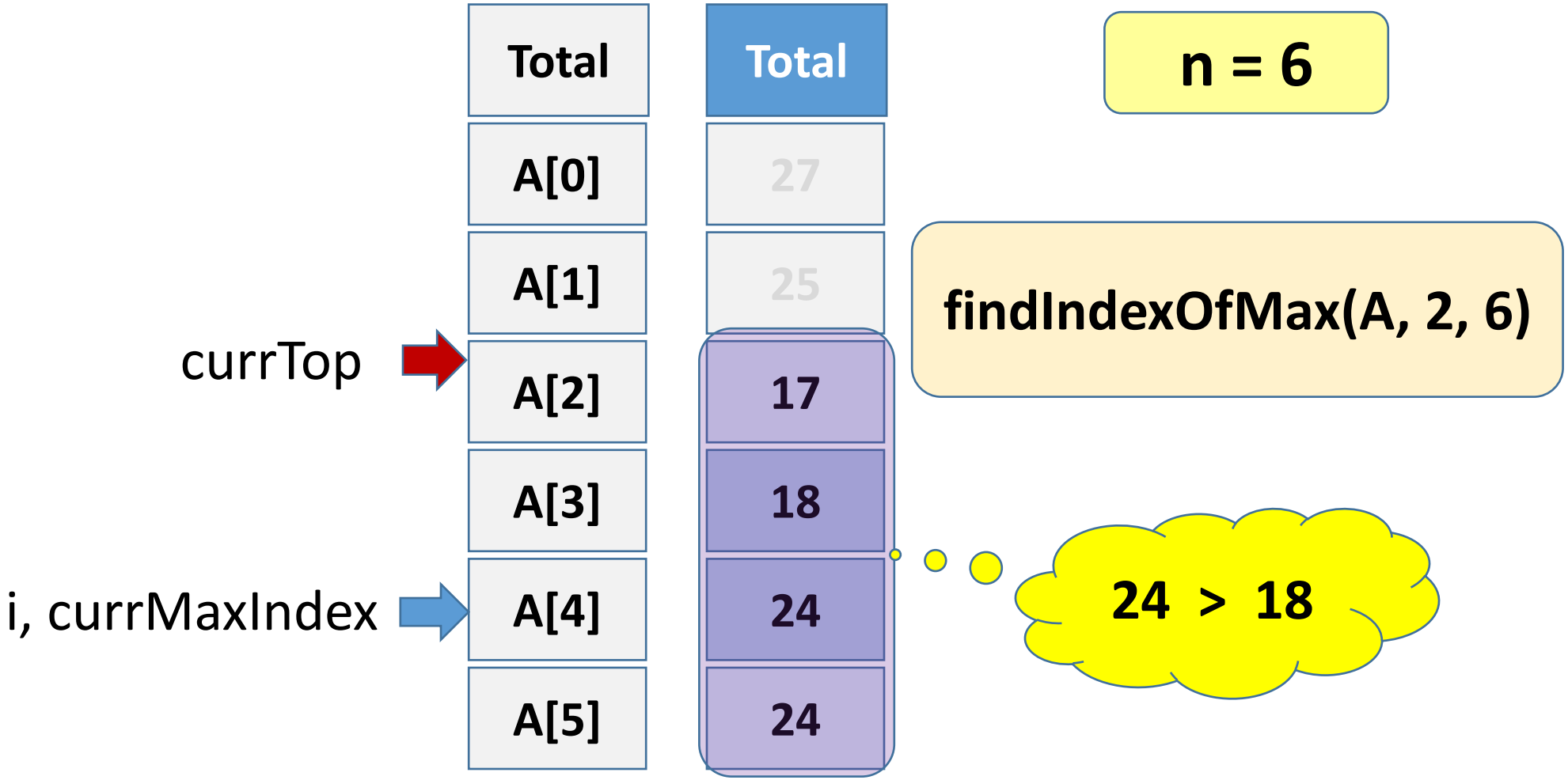
Selection Sort Using >

IIT Bombay



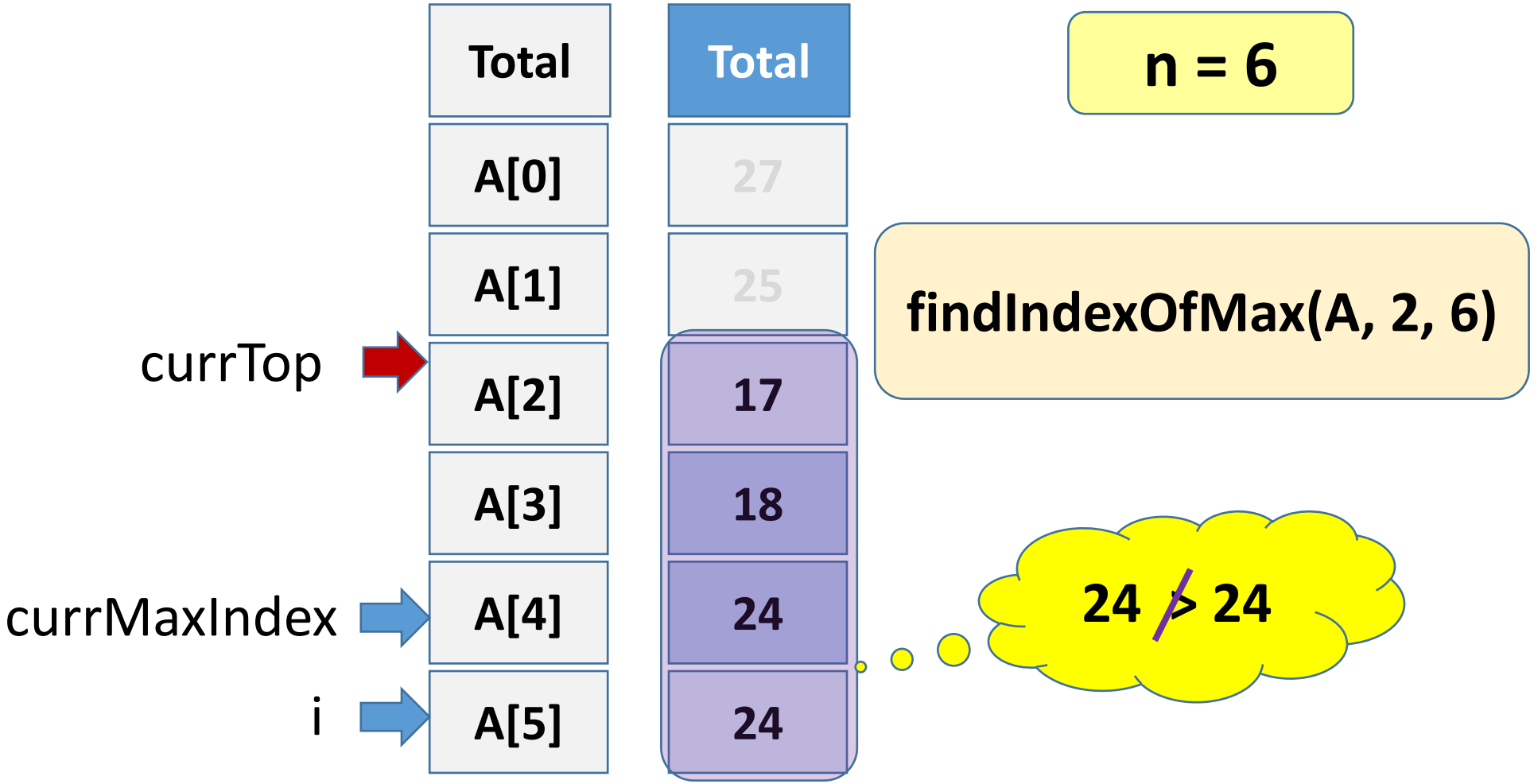
Selection Sort Using >

IIT Bombay



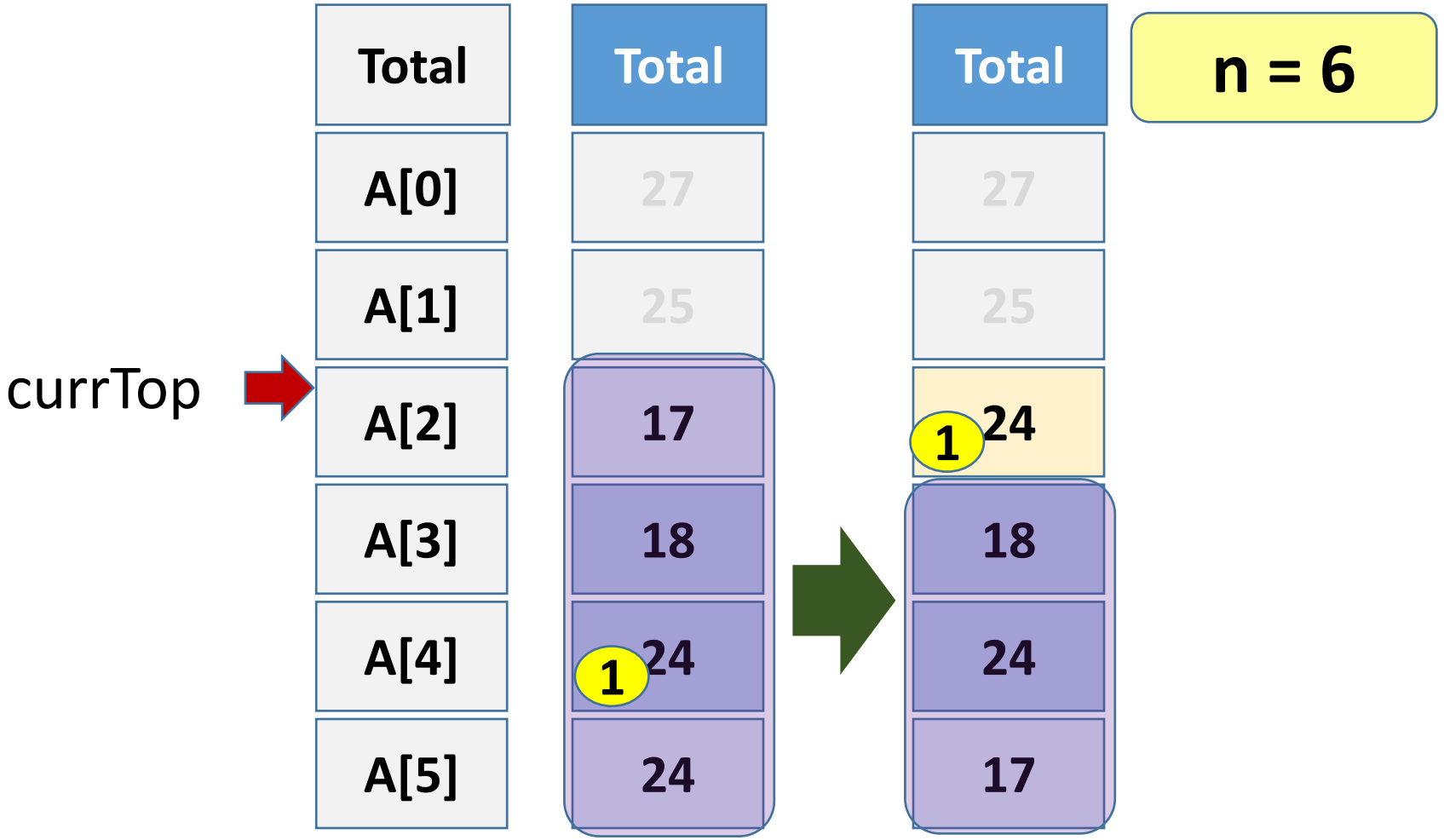
Selection Sort Using >

IIT Bombay



Selection Sort Using >

IIT Bombay



Role of Comparison Operator

IIT Bombay

```
// PRECONDITION: start < end
// start, end within array bounds o
int  findIndexOfMax(int A[],  int sta
    int i, currMinIndex  = sta
    for ( i = start ; i <  nd; i++ ) {
      if (A[i] <= A[ currMinIndex ]) { currMinIndex  = i; }
```

What if we used "<=" ?

Choice of comparison operator crucially determines sorting order (increasing/decreasing), and also how equal elements are ordered!

```
// all elements in A[start] through A[end-1], no change in A
```

Summary

- Selection sort
  - Intuition
  - C++ implementation
  - Choice of comparison operator and its effects