IIT Bombay

Computer Programming

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

Session: Sorting Strings and Other Data Types

Quick Recap of Relevant Topics

- The sorting problem
- Selection sort
- Merge sort
- Counting “basic” steps in sorting an array

Overview of This Lecture

- Sorting strings and other data types
- Other techniques for sorting …

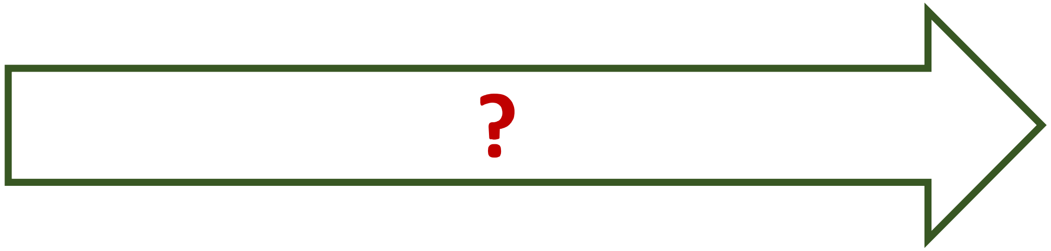
Food For Thought …

- We've looked at techniques for sorting arrays of integers
- What if we wanted to sort arrays of
  - floats, doubles, …
  - strings …
  - more complex data items …
- Do we need to re-invent sorting techniques?

NO!!! Almost the same techniques used for integers also work in these other settings!

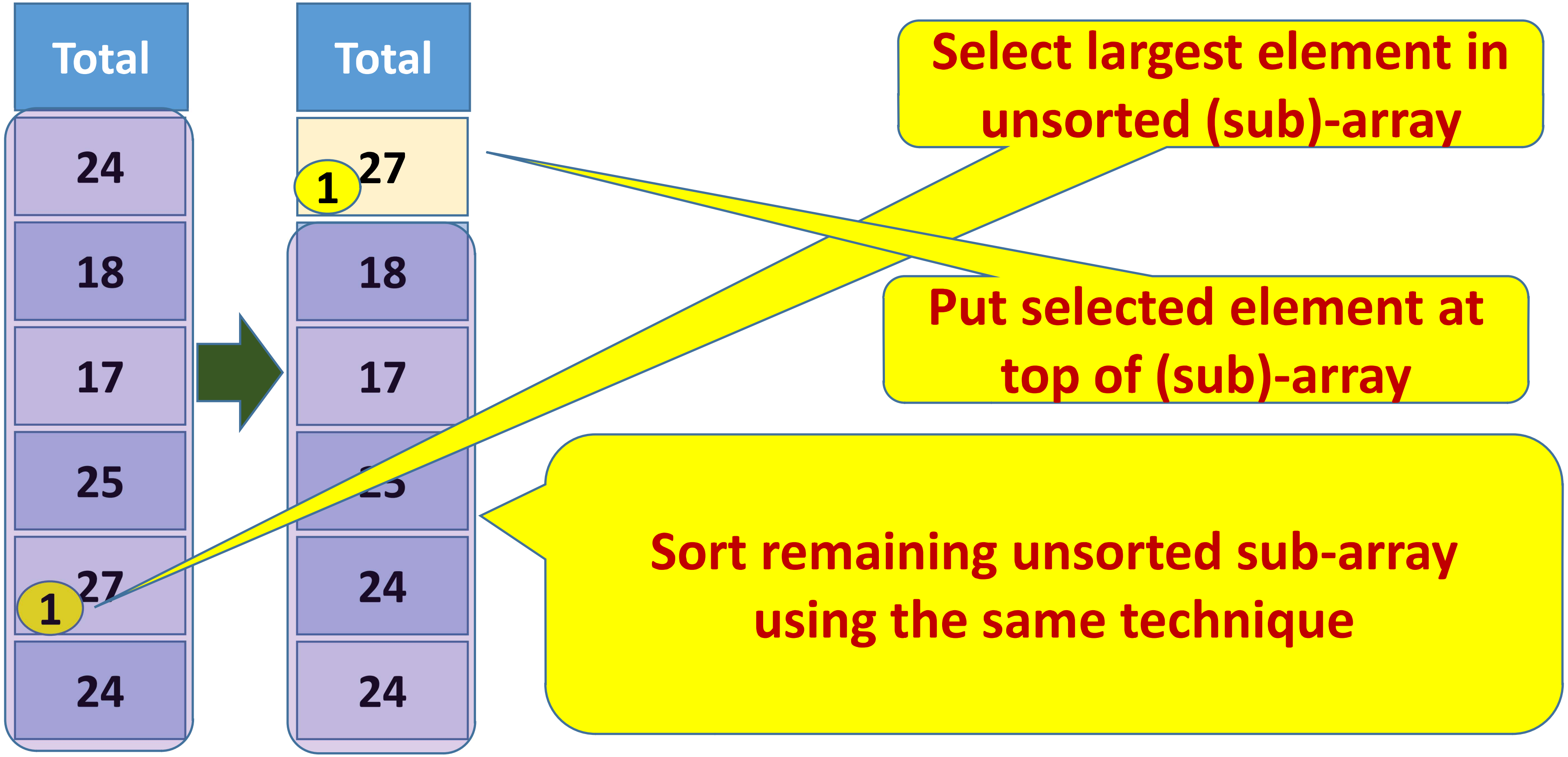
Recap: How Did We Sort Integers?

Total
24
18
17
25
27
24



Total
27
25
24
24
18
17

Recap: Selection Sort (Decreasing Order)



Select largest element in unsorted (sub)-array

Put selected element at top of (sub)-array

Sort remaining unsorted sub-array using the same technique

Recap: Selection Sort in C++ (For Integers)

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;
}
```

Recap: Selection Sort in C++ (For Integers)

IIT Bombay

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

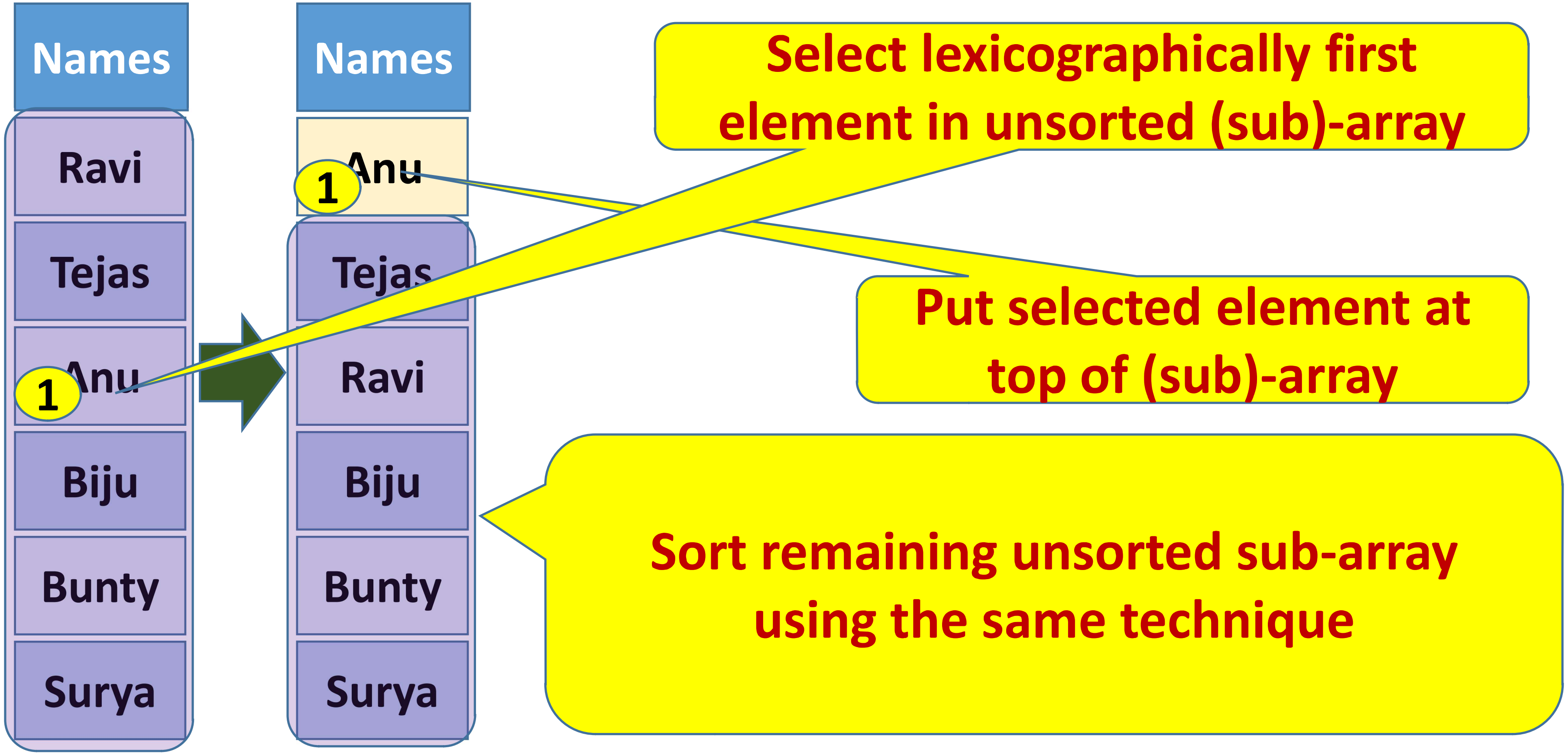
Can We Use The Same Idea To Sort Strings?

Names
Ravi
Tejas
Anu
Biju
Bunty
Surya

Lexicographic Sort (dictionary order)

Names
Anu
Biju
Bunty
Ravi
Surya
Tejas

Selection Sort (Lexicographic Order)



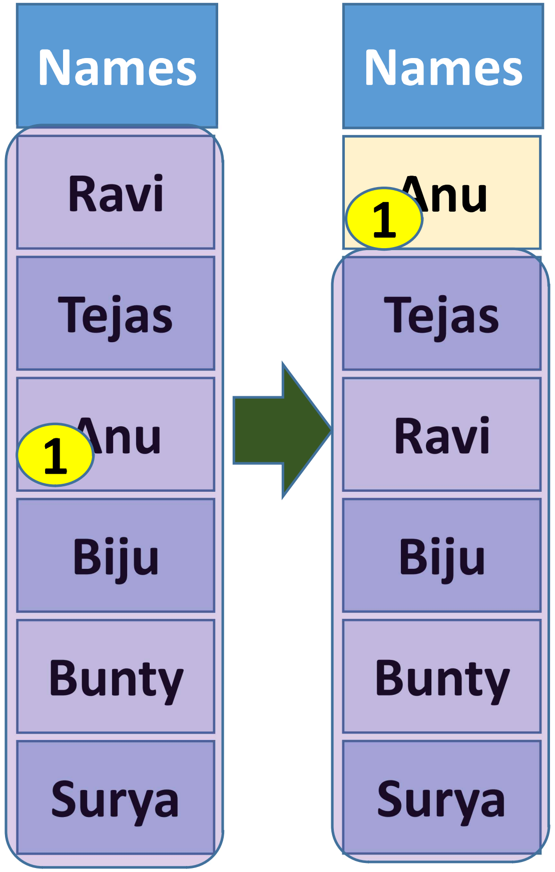
Select lexicographically first element in unsorted (sub)-array

Put selected element at top of (sub)-array

Sort remaining unsorted sub-array using the same technique

Selection Sort (Lexicographic Order)

IIT Bombay



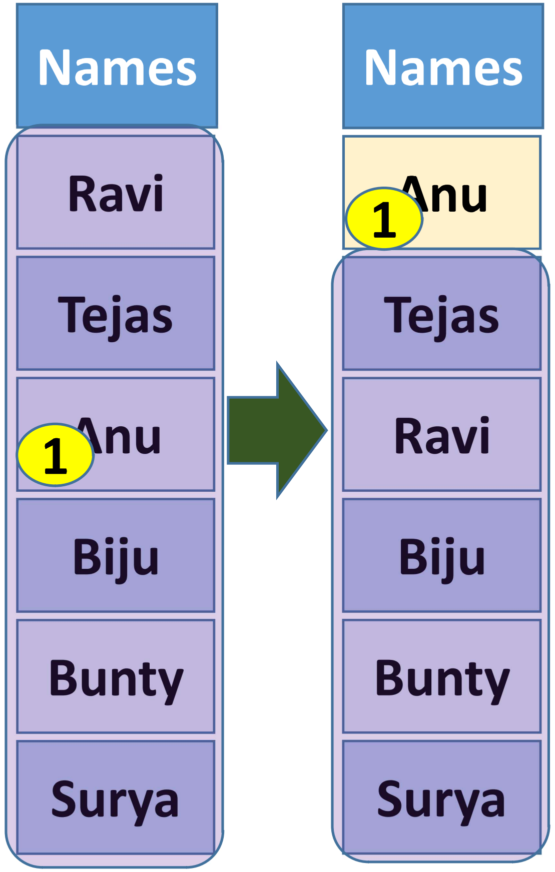
Names	Names
Ravi	Anu 1
Tejas	Tejas
Anu 1	Ravi
Biju	Biju
Bunty	Bunty
Surya	Surya

If we can select lexicographically first string in unsorted (sub)-array, we can sort strings lexicographically

How do we select lexicographically first element?

Iterate over array and compare (order) pairs of elements

Selection Sort (Lexicographic Order)



If we have a function to lexicographically compare a pair of strings, we can sort strings !!!

Comparing Strings

- Assume availability of function

```
// PRECONDITION: s1, s2 strings
bool lexEarlier(string s1, string s2) {
                                …
}
// POSTCONDITION: return true if and only if s1
// lexicographically before s2
```

- We’ll see later (when we study strings) how to implement **lexEarlier**

Selection Sort in C++ (For Strings)

IIT Bombay

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

Selection Sort In C++ (For Strings)

```
int  findIndexOfLexFirst(string A[],  int start,  int end) {
     int i, currLexFirstIndex = start;
     for ( i = start ; i < end; i++ ) {
       if ( lexEarlier (A[i], A[currLexFirstIndex]) ) {
           currLexFirstIndex = i;
       }
     }
    return currLexFirstIndex;
  }
```

Array of strings

How About Merge Sort On Strings?

- Did we just get lucky with selection sort?
- Can we sort strings using merge sort, given the function **lexEarlier** ?

What Were The Steps In Merge Sort?

- Divide an array of size n into two sub-arrays of size ≈ n/2
  - Sub-array sizes may differ by 1 if n is odd
  - Easy!
- Sort each sub-array of size n/2
  - Use same technique as for sorting original array (recurse !!!)
  - Termination case of recursion: arrays of size 1
- Merge sorted sub-arrays, each of size n/2
  - One pass over each sorted sub-array

Crucial Step

Recap: Merging Sorted Sub-arrays (of int) In C++

```
// PRECONDITION: A[start] … A[mid-1]  and A[mid] … A[end-1] sorted in
//                              decreasing order
void mergeSortedSubarrays(int A[], int start, int mid, int end) {
   int i,   j; int tempA[100],  index = start;
   for (i = start, j = mid; ((i < mid) || (j < end)); ) {  // Merging loop
      // Determine whether A[i] or A[j] should appear next in sorted order
      // Update tempA[index] accordingly
     index ++;
  } // end of merging loop
  // Copy tempA[start] … tempA[end-1]  to  A[start] … A[end-1]
  return;
}
// POSTCONDITION:  A[start] … A[end-1] sorted in decreasing order
```

Recap: Merging Loop (for int) In C++

```
for (i = start, j = mid; ((i < mid) || (j < end)); ) {  // Merging loop
     if ((i < mid) && (j < end)) {  // None of the two subarrays fully seen yet
            if (A[j] > A[i])      {tempA[index] = A[j];   j++;}
            else                  {tempA[index] = A[i];   i++;}
    }
    else { if (i < mid) {tempA[index] = A[i]; i++;}  // A[mid] … A[end-1] seen
            else          {tempA[index] = A[j]; j++;} // A[start] … A[mid-1] seen
    }
    index ++;
  } // end of merging loop
```

Merging Sorted Sub-arrays Of Strings In C++

```
// PRECONDITION: A[start] … A[mid-1]  and A[mid] … A[end-1] sorted in
//                              lexicographic order
void mergeSortedSubarrays(string A[], int start, int mid, int end) {
   int i,   j; string tempA[100] ;  int index = start;
   for (i = start, j = mid; ((i < mid) || (j < end)); ) {  // Merging loop
      // Determine whether A[i] or A[j] should appear next in sorted order
      // Update tempA[index] accordingly
     index ++;
  } // end of merging loop
  // Copy tempA[start] … tempA[end-1]  to  A[start] … A[end-1]
  return;
}
// POSTCONDITION:  A[start] … A[end-1] sorted in lexicographic order
```

Merging Loop (For Strings) In C++

```
for (i = start, j = mid; ((i < mid) || (j < end)); ) {  // Merging loop
    if ((i < mid) && (j < end)) {  // None of the two subarrays fully seen yet
          if ( lexEarlier(A[j], A[i]) )  {tempA[index] = A[j];   j++;}
          else                          {tempA[index] = A[i];   i++;}
   }
   else { if (i < mid) {tempA[index] = A[i]; i++;}  // A[mid] … A[end-1] seen
          else         {tempA[index] = A[j]; j++;} // A[start] … A[mid-1] seen
   }
   index ++;
} // end of merging loop
```

What About Arrays Of Other Data Types?

- Pretty much the same technique as for int/string arrays
- Define an ordering function/operator
  - >= for integers
  - lexEarlier(s1, s2) for strings
  - …
- Rest of it is the same …

Other Sorting Techniques

- Selection sort and merge sort not the only sorting techniques
- Other important sorting techniques
  - Quick sort
  - Heap sort
  - Insertion sort …

  All of these can be implemented in C++

- An extremely interesting area of study

Summary

- Sorting strings
  - Key use of comparison operator
- Sorting other data types
- Other sorting techniques