

CSE1301 Computer Programming Lecture 32: List Sorting

1

Topics

- Sorting lists
- Selection sort
- Insertion sort
- Bubble sort

2

Sorting

- Aim:
 - start with an unsorted array
 - end with a sorted array
- How to sort student records?
 - depends on purpose
 - by name, ID number, marks
- Exercise: how to sort words?

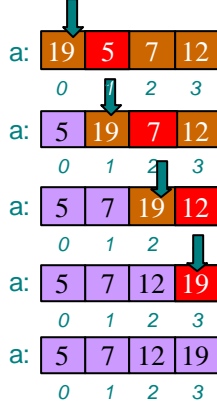
3

Selection Sort

- Basic idea:
 - find the minimum element
 - exchange it with the first unsorted element of the array
 - repeat for the rest of the array

4

Selection Sort -- Example



5

Selection Sort: Algorithm and Code

<pre> selectionSort(array, N) { set count to 0 while (count < N) { set posmin to index of smallest element in rest of array swap item at posmin with item at count add 1 to count } } </pre>	<pre> void selectionSort(int *arr, int N) { int posmin; int count, tmp; for(count=0;count<N;count++) { posmin= findIndexMin(arr, count, N); tmp=arr[posmin]; arr[posmin]=arr[count]; arr[count]=tmp; } } </pre>
---	---

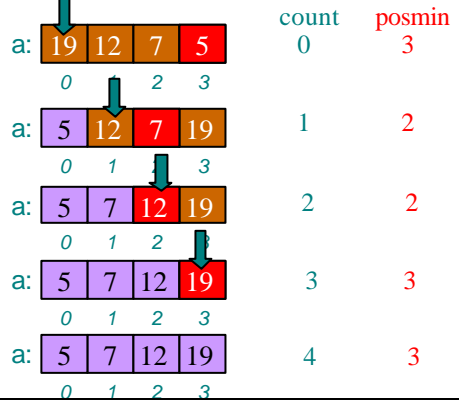
6

Selection Sort: Algorithm and Code (cont)

<pre> findIndexMin(array, start, N) { set posmin to start set count to start while (count < N) { if (current element < element at posmin) { set posmin to count } increment count by 1 } return posmin } </pre>	<pre> int findIndexMin(int *arr, int start, int N) { int posmin=start; int index; for(index=start; index<N; index++) { if (arr[index]<arr[posmin]) { posmin=index; } } return posmin; } </pre>
---	--

7

Selection Sort -- Example



8

Selection Sort Analysis

- What is the time complexity of this algorithm?
- Worst case == Best case == Average case
- Each iteration performs a linear search on the rest of the array

• first element	N +
• second element	N-1 +
• ...	
• penultimate element	2 +
• last element	1
<hr/>	
• Total	$N(N+1)/2 = O(N^2)$

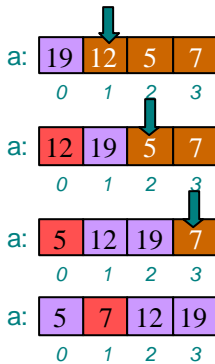
9

Insertion Sort

- Basic idea (*sorting cards*):
 - Take the first unsorted item (assume that the portion of the array in front of this item is sorted)
 - Insert the item in the correct position in the sorted part of the array

10

Insertion Sort -- Example



11

Insertion Sort: Algorithm and Code

<pre> insertionSort(array) { set count to 1 while (count < N) { set val to array[count] set pos to count-1 while (pos is in the array and val < item in pos) { shuffle item in pos one place to right decrement pos by 1 } put val in pos+1 add 1 to count } } </pre>	<pre> void insertionSort(int *arr, int N) { int pos; int count, val; for(count=1;count<N;count++) { val = arr[count]; for(pos=count-1;pos>=0;pos--) { if (arr[pos]>val) { arr[pos+1]=arr[pos]; } else { break; } } arr[pos+1] = val; } } </pre>
---	--

12

Insertion Sort -- Example

	count	val	pos
a: 19 12 5 7	1	12	0
0 1 2 3			
a: 19 19 5 7	1	12	-1
0 1 2 3			
a: 12 19 5 7			
0 1 2 3			
a: 12 19 5 7			
0 1 2 3			

13

Insertion Sort -- Example (cont)

	count	val	pos
a: 12 19 5 7	2	5	1
0 1 2 3			
a: 12 19 5 7	2	5	0
0 1 2 3			
a: 12 12 5 7	2	5	-1
0 1 2 3			
a: 5 12 19 7			
0 1 2 3			
a: 5 12 19 7			
0 1 2 3			

14

Insertion Sort -- Example (cont)

	count	val	pos
a: 5 12 19 7	3	7	2
0 1 2 3			
a: 5 12 19 7	3	7	1
0 1 2 3			
a: 5 12 12 19	3	7	0
0 1 2 3			
a: 5 7 12 19			
0 1 2 3			
a: 5 7 12 19			
0 1 2 3			

15

Insertion Sort Analysis

- What is the time complexity of this algorithm?
- Worst case > Average case > Best case
- Each iteration inserts an element at the start of the array, shifting all sorted elements along
 - second element 2 +
 - ...
 - penultimate element N-1 +
 - last element N

• Total $(2+N)(N-1)/2 = O(N^2)$

16

Bubble Sort

- Basic idea (*lighter bubbles rise to the top*):
 - Exchange neighbouring items until the largest item reaches the end of the array
 - Repeat for the rest of the array

17

Bubble Sort -- Example

a: 19 5 12 7	a: 5 12 7 19
0 1 2 3	0 1 2 3
a: 5 19 12 7	a: 5 7 12 19
0 1 2 3	0 1 2 3
a: 5 12 19 7	a: 5 7 12 19
0 1 2 3	0 1 2 3
a: 5 12 7 19	a: 5 7 12 19
0 1 2 3	0 1 2 3
a: 5 12 7 19	a: 5 7 12 19
0 1 2 3	0 1 2 3

18

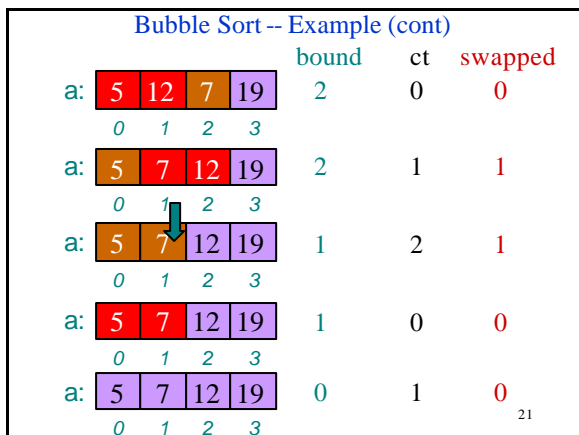
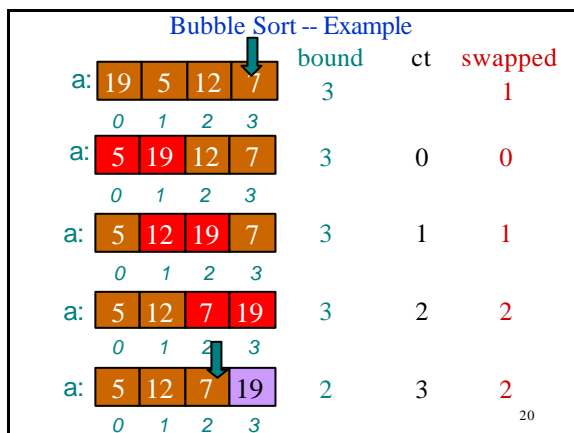
Bubble Sort: Algorithm and Code

```

bubbleSort( array, N )
{
  set bound to N-1
  set swapped to 1
  set count to 0
  while ( swapped > 0 )
  {
    set swapped to 0
    while ( count < bound )
    {
      if ( array[count] >
          array[count+1] )
      {
        swap array[count] and
          array[count+1]
        set swapped to count
      }
      increment count by 1
    }
    set bound to swapped
  }
}

void bubbleSort(int *arr,
                int N)
{
  int ct, temp, bound = N-1;
  int swapped = 1;
  while (swapped > 0 )
  {
    swapped = 0;
    for(ct=0;ct<bound;ct++)
    {
      if ( arr[ct] >
          arr[ct+1] )
      { /* swapping items */
        temp = arr[ct];
        arr[ct] = arr[ct+1];
        arr[ct+1] = temp;
        swapped = ct;
      }
    }
    bound=swapped;
  }
}
    
```

19



Bubble Sort Analysis

- What is the time complexity of this algorithm?
- Worst case > Average case > Best case
- Each iteration compares all the adjacent elements, swapping them if necessary
 - first iteration N +
 - second iteration N-1 +
 - ...
 - last iteration 1

- Total $N(1+N)/2 = O(N^2)$

22

Summary

- Insertion, Selection and Bubble sort:
 - Worst case time complexity is $O(N^2)$

Bestsorting routines are $O(N \log(N))$
In CSE1303 next semester

23

Reading

- Forouzan and Gilberg, Section 8.5
- Selection sort --
 - Knuth, *Sorting and Searching*, pages 139-142 (ed 1), pages 138-141 (ed 2)
- Insertion sort --
 - Brookshear, pages 154-157
 - Knuth, *Sorting and Searching*, pages 80-82
- Bubble sort --
 - Goldschlager & Lister, Section 2.7
 - Knuth, *Sorting and Searching*, pages 105-108

24