CSC 212: Data Structures and Abstractions  
Basic Sorting Algorithms

Marco Alvarez  
Department of Computer Science and Statistics  
University of Rhode Island  
Spring 2018

Administrativia

- Autograder is up and running  
  - submit functions.cc and functions.h (read Piazza post)

- Weekly Assignments
  - new due date: before lecture every Thursday
  - videos: must be sent through Piazza no later than Thursday (only 1 award per week)

- Videos of Advanced Concepts
  - before starting on your video, seek approval of instructor

Total Order

- Every pair of items must be comparable according to a total order, that satisfies:

  Antisymmetry: if \( k_1 \leq k_2 \) and \( k_2 \leq k_1 \) then \( k_1 = k_2 \)

  Transitivity: if \( k_1 \leq k_2 \) and \( k_2 \leq k_3 \) then \( k_1 \leq k_3 \)

  Totality: \( k_1 \leq k_2 \) or \( k_2 \leq k_1 \)

Partial Order

Set of all subsets of a three-element set \( \{x, y, z\} \), ordered by inclusion. Not every pair is comparable.
Sorting

- Given \( n \) elements that can be compared according to a total order relation
  - we want to rearrange them in non-increasing / non-decreasing order, for example:

  - input: sequence \( A = [k_1, k_2, \ldots, k_n] \) of items
  - output: permutation \( B \) of \( A \) s.t. \( B[1] \leq B[2] \leq \ldots \leq B[n] \)

Central problem in computer science

Selection Sort

- Array is divided into sorted and unsorted parts
  - algorithm scans array from left to right

- Invariants
  - elements in sorted are fixed and in ascending order
  - no element in unsorted is smaller than any element in sorted

\[
\begin{array}{cccccccc}
1 & 3 & 4 & 5 & 9 & 6 & 10 & 15 & 7 \\
\end{array}
\]

\begin{array}{c}
\text{sorted} \\
\text{unsorted}
\end{array}

void selectionsort(ul_int *A, ul_int n) {
ul_int i, j, min, temp;
// grows the left part (sorted)
for (i = 0; i < n; i++) {
  min = i;
  // find min in unsorted part
  for (j = i+1; j < n; j++) {
    if (A[j] < A[min]) {
      min = j;
    }
  }
  // swap A[i] and A[min]
  temp = A[i];
  A[i] = A[min];
  A[min] = temp;
}
}

Number of comparisons?
Number of exchanges?
Analysis — Selection Sort (comparisons)

- Worst-case?
- Best-case?
- Average-case?
- Running time is quadratic
  - insensitive to the input (quadratic in all cases)
  - linear number of exchanges (minimal data movement)

Insertion Sort

- Array is divided into sorted and unsorted parts
  - algorithm scans array from left to right
- Invariants
  - elements in sorted are in ascending order
  - elements in unsorted have not been seen

Invariants

<table>
<thead>
<tr>
<th>2</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>9</th>
<th>1</th>
<th>10</th>
<th>3</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorted</td>
<td>unsorted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of comparisons?

Number of exchanges?

```c
1 void insertionsort(ul_int *A, ul_int n) {
2    ul_int temp, i, j;
3    // grows the left part (sorted)
4    for (i = 0 ; i < n ; i ++) {
5        // inserts A[j] in sorted part
6            for (j = i ; j > 0 ; j --) {
7                if (A[j] < A[j-1]) {
8                    temp = A[j];
10                   A[j-1] = temp;
11                }
12            } else
13                break;
14        }
15    }
```
Analysis — Insertion Sort (comparisons)

• Running time depends on the input
• Worst-case?
  ✓ input reverse sorted
• Best-case?
  ✓ input already sorted
• Average-case?
  ✓ expect every element to move $O(n/2)$ times

Partially sorted arrays

• An inversion is a pair of keys that are out of order

```
1 3 4 5 2 6 10 15 7
```

“array is partially sorted if the number of pairs that are out-of-order is $O(n)$”

For partially-sorted arrays, insertion sort runs in linear time.

$\Theta(n)$

Summary

<table>
<thead>
<tr>
<th></th>
<th>Best-Case</th>
<th>Average-Case</th>
<th>Worst-Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Sort</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
</tr>
</tbody>
</table>