ITEC2620 Introduction to Data Structures

Lecture 2a

Sorting I

Introduction to Sorting I

- Why is sorting important?
  - Easier to search sorted data sets
  - searching and sorting are primary problems of computer science

Introduction to Sorting II

- Sorting in General
  - Arrange a set of elements by their “keys” in increasing/decreasing order

Introduction to Sorting III

- Sort cards
  - A full deck
  - A set of cards dealt to you

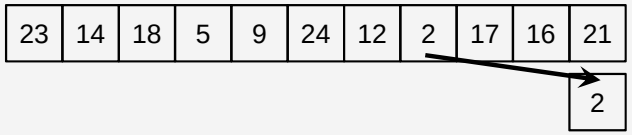
Introduction to Sorting IV

- A full deck
  - Find the smallest value A spades
  - Find the next smallest value 2 spades
  - Find the next smallest value 3 spades
  - Repeat…
    - Selection sort
    - Select the next smallest value each time

Introduction to Sorting V

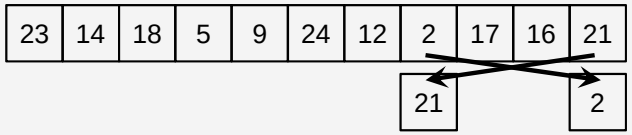
- A set of cards dealt to you
  - Order the first two cards
  - Insert the next card into this subset
  - Insert the next card into this subset
  - Repeat…
    - Insertion sort
    - Insert the new value each time

Selection Sort I



- Find the smallest value and move it into position

Selection Sort II



- What do you do with what was previously there?
  - "swap" it to where the smallest value was
  - sorting can work on the original array

Selection Sort III

23	14	18	5	9	24	12	2	17	16	21
										2
									5	2
								9	5	2
							12	9	5	2

Selection Sort IV

- Pseudocode
  - loop through all elements (have to put n elements into place)
    - ➢for loop
  - loop through all remaining elements and find smallest
    - ➢initialization, for loop, and branch
  - swap smallest element into correct place
    - ➢single method

Insertion Sort I

23	14	18	5	9	24	12	2	17	16	21
										21

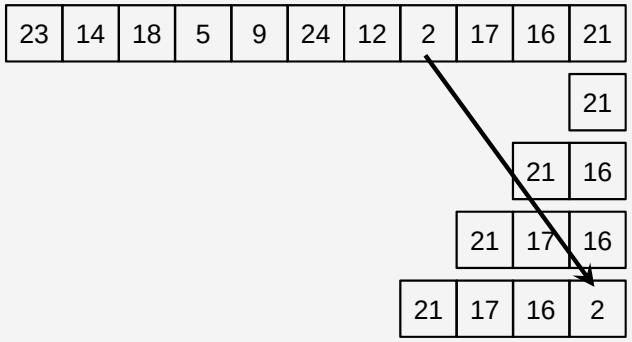
- Get the next value, and push it over until it is “semi” sorted

Insertion Sort II

23	14	18	5	9	24	12	2	17	16	21
										21

- elements in selection sort are in their final position
- elements in insertion sort can still move

Insertion Sort III



Insertion Sort IV

- Pseudocode
  - loop through all elements (have to put n elements into place)
    - ➢for loop
  - loop through all sorted elements, and swap until slot is found
    - ➢while loop
    - ➢swap method

Bubble Sort

- Slow and useless
- Relay race
  - pair-wise swaps until smaller value is found
  - smaller value is then swapped up
- NEVER use this method

Cost of Sorting I

- Selection Sort
  - Is there a best, worst, and average case?
    - ➢two for loops → always the same
  - n elements in outer loop
  - n-1, n-2, …, 2, 1 elements in inner loop
    - ➢average n/2 elements for each pass of the outer loop
  - n * n/2 compares

Cost of Sorting II

- Insertion Sort
  - Worst – same as selection sort, next element swaps till end
    - ➢n * n/2 compares
  - Best – next element is already sorted, no swaps
    - ➢n * 1 compares
  - Average – linear search, 50% of values
    - ➢n * n/4 compares

Value of Sorting I

- Current cost of sorting is roughly n^2 compares
- Toronto phone book
  - 2 million records
  - 4 trillion compares to sort

Value of Sorting II

- 2 million records
- Linear search
  - 1 million compares
- Binary search
  - 20 compares

Value of Sorting III

- 100 searches
  - unsorted array, linear search
    - ➢100 million total compares
  - sorted array, binary search
    - ➢4 trillion and 2 thousand total compares
  - sorting the array first is 4000 times slower

Value of Sorting IV

- 200 million searches (every person does 100 searches)
  - unsorted, linear search
    - ➢200 trillion total compares
  - sorted, binary search
    - ➢4 trillion and 4 billion total compares
  - sorting the array first is 50 times faster

Trade-Offs I

- Buy a parking pass, or pay cash each time?
- Sort in advance, or do linear search each time?
- Trade-offs are an important part of program design
  - How and what to optimize…

Trade-Offs II

- How can we measure the costs of what we are trading-off?
- Need a method to analyze programs…
  - Complexity analysis
  - Next lecture!

Readings and Assignments

- Suggested Readings from Shaffer (third edition)
  - 7.1, 7.2