CO322: Data structures and algorithms

Dhammika Elkaduwe

Department of Computer Engineering
Faculty of Engineering

University of Peradeniya

What is on this set of slides?

- Simple sorting algorithms
- Recursive sorting algorithms
- Searching algorithms
- Complexity of sorting algorithms
- Correctness of these algorithms

Why sorting/searching

Simple, yet quite useful operations. Examples:

- Packets that were captured (in CO227 project)
- Words in a dictionary
- Data items in catalogue
- ...

Also, good enough to demonstrate basic concepts.

Simple Sorting Algorithms: Bubble sort

Algorithm (one possible version):

1. Start wit the last item,
2. compare it with the previous item and if the previous item is large, swap the current item and the previous.
3. Move to the next
4. Continue from step 2 until you have reached the first item, now the smallest item will be in the 1st position.
5. Start at the last item, and repeat step 2 through to 4, until you reach the 2^{nd} item, now the 2^{nd} smallest item will be in the 2^{nd} position
6. Repeat until all the item are in order.

Algorithms:

```
public void bubble_sort(int [] data) {
/* Implement above algorithm
*/
}
```

Simple Sorting Algorithms: Bubble sort – code

One possible implementation of bubble sort

```
public void bubble_sort(int [] data) {
int i,j;
for(i=0; i < data.length; i++) {
  for(j = data.length-1; j > i; j--) {
    if(data[j] < data[j-1]) {
      int tmp = data[j];
      data[j] = data[j-1];
      data[j-1] = tmp;
    } // end if
  } // for(j= ...
       } // for(i=...
}
```

Is this the best implementation?

Simple Sorting Algorithms: Bubble sort – Optimisation

Previous implementation is not the best!

```
public void bubble_sort(int [] data) {
int i,j;
for(i=0; i < data.length; i++) {
  for(j = data.length-1; j > i; j--) {
    if(data[j] < data[j-1]) {
      int tmp = data[j];
      data[j] = data[j-1];
      data[j-1] = tmp;
    }
  }
        }
}
```

If you did not do any swaps in one pass, then you are done! (i.e. you do not have to check again!!!)

Simple Sorting Algorithms: Bubble sort – Optimisation

One possible implementation of bubble sort optimisation.

```
static void buble_sort_opt(int [] data) {
  boolean quit = false;
  for(int i=0; i<data.length && !quit; i++) {
    quit = true;
    for(int j=data.length-1; j > i; j--) {
              if(data[j] < data[j-1]) {
                int tmp = data[j];
                data[j] = data[j-1];
                data[j-1] = tmp;
                quit = false;
              }
    }
  }
}
```

How to compare algorithms?

Which implementation is better?

- Of cause both algorithms are correct
- What about space? (memory consumed)
- What about time?
- Does it varies with the input size?
- Does it varies with the input configuration?
- Does it depends on the computer on which it runs?
- Does it depends on the programming language?

So which is the efficient algorithm?

Comparing algorithms

The Random Access Model (RAM)

- *Simple* operations (ex: $+$, $==$, $=$) takes one time step,
- Each memory access takes one time step (ex: no cache),
- Loops and subroutines are collections of *simple* operations

We can compute the number of time steps required for the algorithm *given a data configuration*.

Dealing with the data configurations:

- Analyse the algorithm performance for best case,
- Analyse the algorithm performance for average case,
- Analyse the algorithm performance for worst case.
- (Worst case is the most useful)

(Note: Best, Average and Worst case configuration depends on the algorithm).

Comparing algorithms

The Random Access Model (RAM)

- *Simple* operations (ex: $+$, $==$, $=$) takes one time step,
- Each memory access takes one time step (ex: no cache),
- Loops and subroutines are collections of *simple* operations

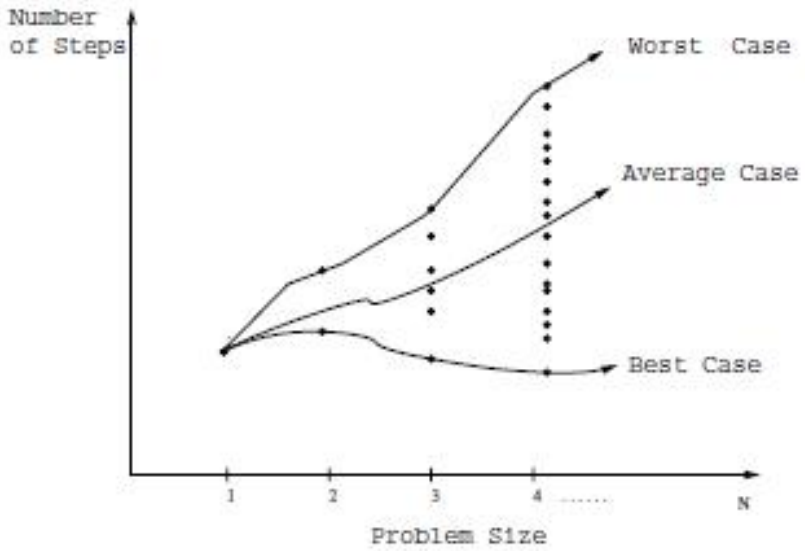
We can compute the number of time steps required for the algorithm *given a data configuration*.

Dealing with the data configurations:

- Analyse the algorithm performance for best case,
- Analyse the algorithm performance for average case,
- Analyse the algorithm performance for worst case.
- (Worst case is the most useful)

(Note: Best, Average and Worst case configuration depends on the algorithm).

Best, average and worst case complexities



- Worst-case complexity is the maximum number of steps taken in any instance of size N.
- Average-case complexity is the average number of steps taken in any instance of size N.
- Best-case complexity is the best number of steps taken in any instance of size N.

Bubble sort – analysis

```
public void bubble_sort(int [] data) {
int i,j;
for(i=0; i < data.length; i++) {
  for(j = data.length-1; j > i; j--) {
    if(data[j] < data[j-1]) {
      int tmp = data[j];
      data[j] = data[j-1];
      data[j-1] = tmp;
    } // end if
  } // for(j= ...
        } // for(i=...
}
```

What is the best-case and worst-case for this algorithm?

Bubble sort – numbers

Using RAM, the bubble sort has following complexities:

Operation	Best-case	Worst-case	Average-case
Comparisons	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$
Swaps	0	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{4}$
Total	$\frac{N(N-1)}{2}$	$N(N-1)$	$\frac{3N(N-1)}{4}$

Too much details! Do we need all this? (later).

Bubble sort – numbers

Using RAM, the bubble sort has following complexities:

Operation	Best-case	Worst-case	Average-case
Comparisons	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$
Swaps	0	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{4}$
Total	$\frac{N(N-1)}{2}$	$N(N-1)$	$\frac{3N(N-1)}{4}$

Too much details! Do we need all this? (later).

Bubble sort – numbers

Using RAM, the bubble sort has following complexities:

Operation	Best-case	Worst-case	Average-case
Comparisons	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$
Swaps	0	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{4}$
Total	$\frac{N(N-1)}{2}$	$N(N-1)$	$\frac{3N(N-1)}{4}$

Too much details! Do we need all this? (later).

Bubble sort – Optimisation

```
static void buble_sort_opt(int [] data) {
  boolean quit = false;
  for(int i=0; i<data.length && !quit; i++) {
    quit = true;
    for(int j=data.length-1; j > i; j--) {
              if(data[j] < data[j-1]) {
                int tmp = data[j];
                data[j] = data[j-1];
                data[j-1] = tmp;
                quit = false;
              }
    }
  }
}
```

Analyse the complexity.

Optimised Bubble sort – numbers

Using RAM, the optimised bubble sort has following complexities:

Operation	Best-case	Worst-case	Average-case
Comparisons	$N-1$	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$
Swaps	0	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{4}$
Total	$N-1$	$N(N-1)$	$\frac{3N(N-1)}{4}$

Too much details! We have additional comparisons right?.

Optimised Bubble sort – numbers

Using RAM, the optimised bubble sort has following complexities:

Operation	Best-case	Worst-case	Average-case
Comparisons	$N-1$	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{2}$
Swaps	0	$\frac{N(N-1)}{2}$	$\frac{N(N-1)}{4}$
Total	$N-1$	$N(N-1)$	$\frac{3N(N-1)}{4}$

Too much details! We have additional comparisons right?.

The Big Oh notation

- Provides a simpler way of comparing algorithm performance.
- Gross over lots of details (which are not that relevant at the end of the day)
- We care about how the algorithm behaves for large problems ($n \to \infty$)

Definition

$f_{(n)} = O(g_{(n)})$ means that there exists some constant C such that $c.g_{(n)} \geq f_{(n)}$, when $n \to \infty$ (for large enough n or when $n \geq n_0$)

Example: $O(\frac{N(N-1)}{2}) = N^2$

The Big Oh notation

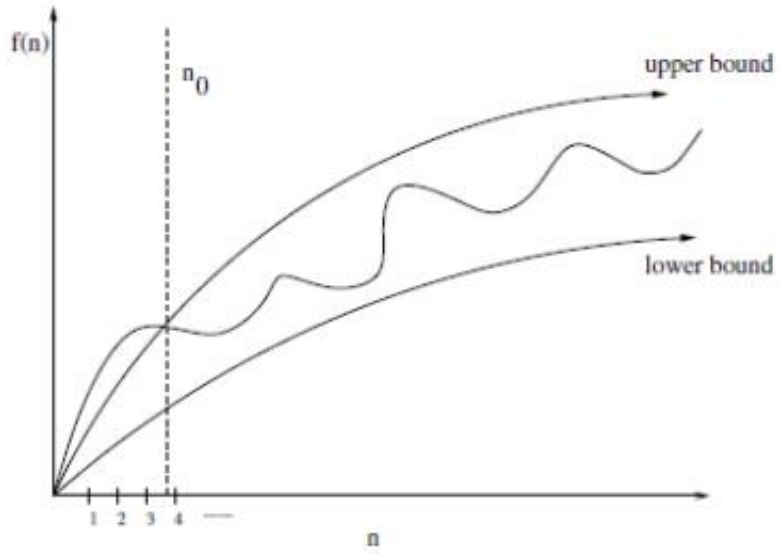
- Provides a simpler way of comparing algorithm performance.
- Gross over lots of details (which are not that relevant at the end of the day)
- We care about how the algorithm behaves for large problems ($n \to \infty$)

Definition

$f_{(n)} = O(g_{(n)})$ means that there exists some constant C such that $c.g_{(n)} \geq f_{(n)}$, when $n \to \infty$ (for large enough n or when $n \geq n_0$)

Example: $O(\frac{N(N-1)}{2}) = N^2$

Other bounds

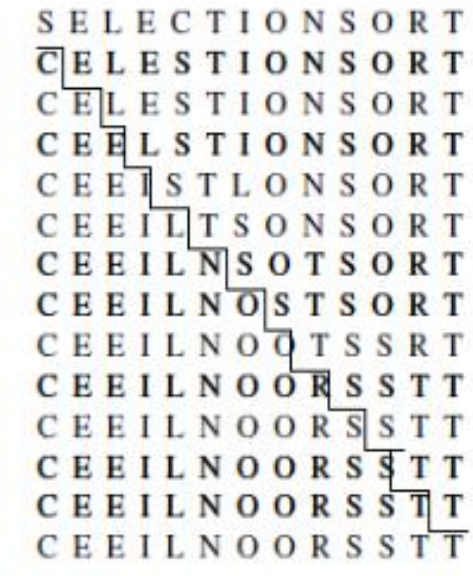


- $f_{(n)} = O(g_{(n)})$ means that there exists some constant C such that $c.g_{(n)} \geq f_{(n)}$, when $n \to \infty$ (for large enough n or when $n \geq n_0$) (Upper bound, worst case)
- $f_{(n)} = \Omega(g_{(n)})$ means that there exists some constant C such that $c.g_{(n)} \leq f_{(n)}$, when $n \to \infty$ (for large enough n or when $n \geq n_0$) (Lower bound)

Exercise

1. Find BigOh and Ω of the following functions:
   1.1 2^{n+1}
   1.2 $n^3 + 3n^2 + 4$
   1.3 $(x^2 + y^3)$
   1.4 $n^6 + n^5 + n^{\frac{1}{2}} + 1$
2. Express the run-time complexity of Bubble sort and Bubble sort – Optimisation using BigOh
   2.1 Which is better?
   2.2 Explain your answer.

Simple sorting – Selection Sort

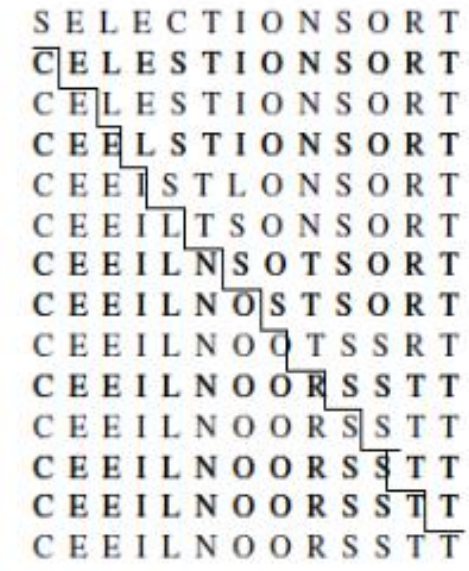


Idea: Find the smallest item in the *unsorted* list and put it at the end of the sorted list (*swap smallest with end of sorted list*). Continue until the list is sorted.

Exercises:

1. Implement *static void selection_sort(int [] data)* in Java
2. What is the run-time complexity of selection sort?
3. Which algorithm is better, selection or bubble sort?
4. What if your data items are really large (difficult to swap)

Simple sorting – Selection Sort

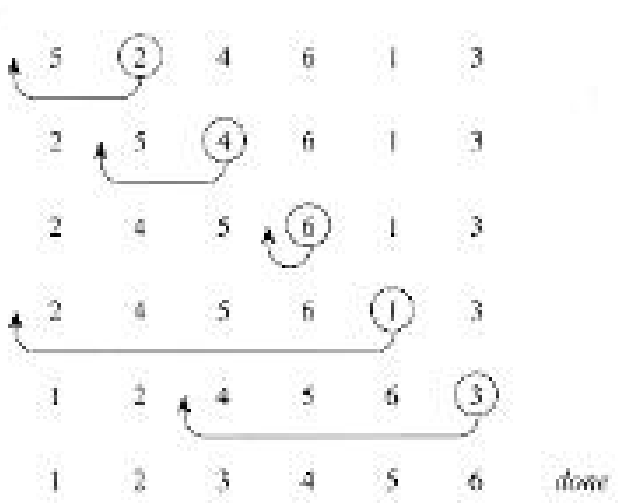


Idea: Find the smallest item in the *unsorted* list and put it at the end of the sorted list (*swap smallest with end of sorted list*). Continue until the list is sorted.

Exercises:

1. Implement *static void selection_sort(int [] data)* in Java
2. What is the run-time complexity of selection sort?
3. Which algorithm is better, selection or bubble sort?
4. What if your data items are really large (difficult to swap)

Simple sorting – Insertion Sort

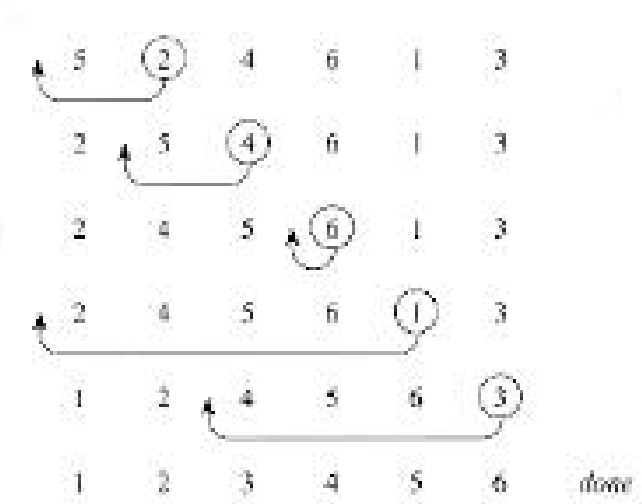


Select the i^{th} item and insert it into the correct place amongst the already sorted (i-1) items. (*you can find space by moving all items one location upwards)*

Exercises:

1. Implement *static void insertion_sort(int [] data)* in Java
2. What is the run-time complexity of insertion sort?
3. Which algorithm is better, insertion or bubble sort?

Simple sorting – Insertion Sort



Select the i^{th} item and insert it into the correct place amongst the already sorted (i-1) items. (*you can find space by moving all items one location upwards)*

Exercises:

1. Implement *static void insertion_sort(int [] data)* in Java
2. What is the run-time complexity of insertion sort?
3. Which algorithm is better, insertion or bubble sort?

Simple sorting algorithms

Fill the following table on complexities of simple sorting algorithms:

Algorithm	Best-case	Worst-case	Average-case
Bubble			
Selection			
Insertion			

which is better? can we do better?

Simple sorting algorithms

Fill the following table on complexities of simple sorting algorithms:

Algorithm	Best-case	Worst-case	Average-case
Bubble			
Selection			
Insertion			

which is better? can we do better?