COMP2521 23T3

·IOLIVALIOII

C-1--4:-- C---

Selection 301

IIISELLIUII 30

Jilett Joilt

Julilliary

Sorting Lists

COMP2521 23T3 Sorting Algorithms (I)

Kevin Luxa cs2521@cse.unsw.edu.au

properties of sorting algorithms elementary sorting algorithms

Motivation

Motivation

Overview

Selection So

Innoution C

Summar

- Sorting enables faster searching
 - Binary search
- Sorting arranges data in useful ways (for humans and computers)
 - For example, a list of students in a tutorial
- Sorting provides a useful intermediate for other algorithms
 - For example, duplicate detection/removal, merging two collections

Overview

Formally Propertie Concrete

Selection 9

msertion sort

Shall Sort

Sullillary

- Sorting involves arranging a collection of items in order
 - Arrays, linked lists, files
- Items are sorted based on some property (called the key), using an ordering relation on that property
 - Numbers are sorted numerically
 - Strings are sorted alphabetically

Overview

We sort arrays of Items, which could be:

- Simple values: int, char, double
- Complex values: strings
- Structured values: struct

The items are sorted based on a key, which could be:

- The entire item, if the item is a single value
- One or more fields, if the item is a struct

COMP2521 23T3

Overview

Overvie

Formally Properties Concretely

election S

Bubble So

Insertion S

Shell Sort

Summary

Sorting

Example: Each student has an ID and a name

5151515	5012345	3456789	5050505	5555555	5432109
John	Jane	Bob	Alice	John	Andrew

Sorting by ID (i.e., key is ID):

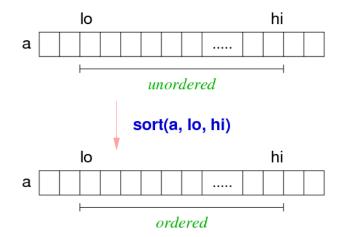
ſ	3456789	5012345	5050505	5151515	5432109	555555
	Bob	Jane	Alice	John	Andrew	John

Sorting by name (i.e., key is name):

5050505	5432109	3456789	5012345	5151515	555555
Alice	Andrew	Bob	Jane	John	John

Motivation Overview

Arrange items in array slice a [lo..hi] into sorted order:



To sort an entire array of size N, lo == 0 and hi == N - 1.

Formally

Pre-conditions: array a[N] of Items lo, hi are valid indices on a (roughly, 0 < lo < hi < N - 1)

Post-conditions: array a [lo..hi] contains the same values as before the sort $a[lo] \le a[lo+1] \le a[lo+2] \le \cdots \le a[hi]$



Properties of Sorting Algorithms

itivatio

Formally

Properties

Adaptabi In-place

Analysis

001001101100

. .

....

311611 301

Summary

Sorting Lists

Properties:

- Stability
- Adaptability
- In-place



Properties of Sorting Algorithms Stability

Motivation

Motivatio

Formal

Stability

Adaptabili In-place Concretely

oloction C

mocreion ooi

Shell Sort

Summar

- A stable sort preserves the relative order of items with equal keys.
- Formally: For all pairs of items x and y where $\mathsf{KEY}(x) \equiv \mathsf{KEY}(y)$, if x precedes y in the original array, then x precedes y in the sorted array.

Properties of Sorting Algorithms Stability

Motivation

Formall

Stability

Stabilit

In-place

Calastian C.

Incombine Com

Summary

Sorting Li

Example: Each card has a value and a suit













A stable sort on value:













Properties of Sorting Algorithms

Stability

Motivation

Example: Each card has a value and a suit

Stability

Adaptability
In-place
Concretely
Analysis

Selection So

Bubble Sort

Insertion 501

Chall Cart

Cumman

Summary













Example of an unstable sort on value:















Properties of Sorting Algorithms Stability

lotivatio

Overview

Stability

Adaptabili In-place

.

Bubble Sort

IIISertion Sor

Shell Sort

Summary

Sorting Lists

When is stability important?

- When sorting the same array multiple times on different keys
 - Some sorting algorithms rely on this, for example, radix sort

Properties of Sorting Algorithms

Stability

Motivatio

Stability

Example: Array of first names and last names

Alice	Andrew	Jake	Alice	Andrew	John
Wunder	Bennett	Renzella	Hatter	Taylor	Shepherd

Sort by last name:

Andrew	Alice	Jake	John	Andrew	Alice
Bennett	Hatter	Renzella	Shepherd	Taylor	Wunder

Then sort by first name (using stable sort):

Alice	Alice	Andrew	Andrew	Jake	John
Hatter	Wunder	Bennett	Taylor	Renzella	Shepherd

Dubble 3011

msertion sor



Properties of Sorting Algorithms Stability

Motivatio

Overviev

Stability

Adaptabili In-place Concretely Analysis

Selection So

Bubble Sor

insertion So

Chall Caut

Summary

Julilliary

Sorting Lists

Stability doesn't matter if...

- All items have unique keys
 - Example: Sorting students by ID
- The key is the entire item
 - Example: Sorting an array of integer values



Properties of Sorting Algorithms Adaptability

Motivation

MOLIVALIO

Properties

Adaptability

In-place

Analysis

Selection So

Dubble Cont

Insertion Sor

01101100

Summar

- An adaptive sorting algorithm takes advantage of existing order in its input
 - Time complexity of an adaptive sorting algorithm will be better for sorted or nearly-sorted inputs
- Can be a useful property, depending on whether nearly sorted inputs are common

Properties of Sorting Algorithms In-place

In-place

 An in-place sorting algorithm sorts the data within the original structure, without using temporary arrays

```
Motivation
```

Propertie:

Concretely

.

Dunnie 2011

Insertion Sort

Chall Carr

Summary

. . .

```
// we deal with generic `Item's
typedef int Item;

// abstractions to hide details of items
#define key(A) (A)
#define lt(A, B) (key(A) < key(B))
#define le(A, B) (key(A) <= key(B))
#define ge(A, B) (key(A) >= key(B))
#define gt(A, B) (key(A) > key(B))

// Sort a slice of an array of Items
void sort(Item a[], int lo, int hi);
```

A Concrete Framework

Motivatio

Formally
Properties
Concretely

.

Bubble Sor

Insertion So

Shall Sart

Summary

Cauting

```
This framework can be adapted by...

defining a different data structure for Item;

defining a method for extracting sort keys;

defining a different ordering (less);

defining a different swap method for different Item
```

```
typedef struct {
    char *name;
    char *course;
} Item;

#define key(A) (A.name)
#define lt(A, B) (strcmp(key(A), key(B)) < 0)
#define le(A, B) (strcmp(key(A), key(B)) <= 0)
#define ge(A, B) (strcmp(key(A), key(B)) >= 0)
#define gt(A, B) (strcmp(key(A), key(B)) > 0)
```

Analysis of Sorting Algorithms

Motivatio

Properties

Analysis

Selection S

Bubble Sor

Insertion Sort

Chall Cart

C.....

Sullillary

Sorting Li

In analysing sorting algorithms:

- n: the number of items (hi lo + 1)
- *C*: the number of comparisons between items
- S: the number of times items are swapped

(We usually aim to minimise C and S.)

Cases to consider for input order:

- random order: Items in a [lo..hi] have no ordering
- sorted order: $a[lo] \le a[lo + 1] \le \cdots \le a[hi]$
- reverse-sorted order: $a[lo] \ge a[lo+1] \ge \cdots \ge a[hi]$

COMP2521 23T3

Examples of Sorting Algorithms

Motivatio

Formally Properties

Analysis

election S

Bubble Sor

Insertion So

Chall Cart

C.....

Summary

Sorting List

Elementary sorting algorithms:

- Selection sort
- Bubble sort
- Insertion sort
- Shell sort

More efficient sorting algorithms:

- Merge sort
- Quick sort

Non-comparison-based sorting algorithms:

- Radix sort
- Key-indexed counting sort

Overvie

Selection Sort

Implemen Analysis Properties

Bubble So

IIISertion 30

Shell Sort

Summarv

Summar

Method:

- Find the smallest element, swap it with the first element
- Find the second-smallest element, swap it with the second element
- .
- Find the second-largest element, swap it with the second-last element

Each iteration improves the "sortedness" of the array by one element

Example

Overview

Selection Sor

Implementatio
Analysis
Properties

Dappie 301

Chall Caut

Summary



Example

.

Selection Sor

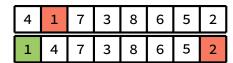
Implementation Analysis

Bubble So

Insertion So

Shell Sort

Summary



Example

Motivation

Selection Sor

Example

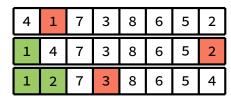
Properties

Bubble Soi

Insertion S

Shell Sort

Summary



Motivation

Overview

Selection Sor

Example

Properties

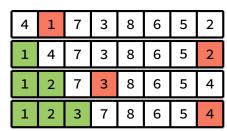
Bubble So

Insortion S

Shall Sort

C.....

Julilliary



Motivation

Salaction Sor

Selection Sor Example

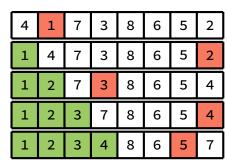
Implementatio Analysis Properties

Bubble So

Insertion S

Shell Sort

Summary



Motivation

_

Selection So

Example

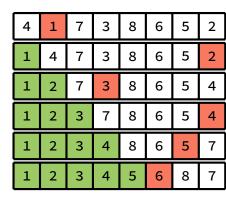
Analysis
Properties

Bubble Soi

Insertion S

Shell Sor

Summary



Motivation

11106114061011

Selection So

Example

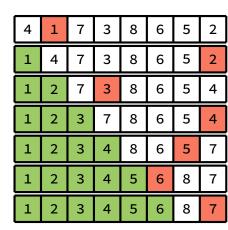
Analysis Properties

Rnppre 20

III3CI CIOII S

Shell Sor

Summary



Motivation

Overview

Selection So

Example

Analysis Properties

Bubble So

Insertion S

Shell Sor

Summary

4	1	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	2	7	3	8	6	5	4
1	2	3	7	8	6	5	4
1	2	3	4	8	6	5	7
1	2	3	4	5	6	8	7
1	2	3	4	5	6	8	7
1	2	3	4	5	6	7	8

Selection Sort C Implementation

```
Motivation
```

Selection So

Implementation

Properties

Bubble So

Insertion 9

Shall Sort

Summarv

```
void selectionSort(Item items[], int lo, int hi) {
    for (int i = lo; i < hi; i++) {</pre>
        int min = i;
        for (int j = i + 1; j <= hi; j++) {
            if (lt(items[j], items[min])) {
                min = j;
        swap(items, i, min);
```

MOLIVALIOI

Selection So Example

Analysis

_ ...

Insertion So

.. .. .

C.....

Sullillary

Sorting Lis

Cost analysis:

- In the first iteration, n-1 comparisons, 1 swap
- In the second iteration, n-2 comparisons, 1 swap
- ..
- In the final iteration, 1 comparison, 1 swap
- $C = (n-1) + (n-2) + \ldots + 1 = \frac{1}{2}n(n-1) \Rightarrow O(n^2)$
- S = n 1

Cost is the same, regardless of the sortedness of the original array.

Selection Sort Properties

Motivatioi

Overview

ection Sort

Properties

11130101011 301

SHELLS

Summar

Sorting List

Selection sort is unstable

- Due to long-range swaps
- Example:

2a 2b 1a

1a 2b 2

Properties

MOLIVALIO

.

xample

Analysis Properties

.

.

61 116 1

Summan

Combinertie

Due to long-range swaps

Non-adaptive

Unstable

Performs same steps, regardless of sortedness of original array

In-place

Sorting is done within original array; does not use temporary arrays

Bubble Sort

Method:

- Make multiple passes from left (lo) to right
- On each pass, swap any out-of-order adjacent pairs
- Elements "bubble up" until they meet a larger element
- Stop if there are no swaps during a pass
 - This means the array is sorted

Bubble Sort Example

Motivatio

Overview

Selection So

Example

Implmentation

Propertie

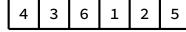
Insertion Sc

Shell Sort

Summary

Sorting Lists

Example



Bubble Sort Example

Motivation

Overview

Example

Analysis Properties

IIISEILIOII 3

Shell Sor

Summary

Sorting Lists

First pass

4	3	6	1	2	5
3	4	6	1	2	5
3	4	6	1	2	5
3	4	1	6	2	5
3	4	1	2	6	5
3	4	1	2	5	6

Bubble Sort Example

Motivation

Overview

Selection 50

Example

Analysis

Properties

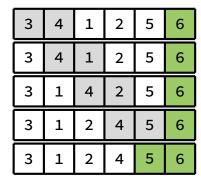
Insertion So

Shell Sor

Summary

Sorting Lists

Second pass



Bubble Sort Example

Motivation

Overview

Selection Sc

Example

Implmentation

Propertie

msertion so

Shell Sor

Summary

Sorting Lists

Third pass

3	1	2	4	5	6
1	3	2	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

Bubble Sort Example

Motivation

Ovoniow

Selection So

Bubble So

Example

Implmentatio

Propertie

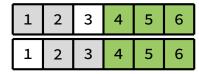
1113611101130

Shell Sort

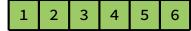
Summary

Sorting Lists

Fourth pass



No swaps made; stop



Bubble Sort C Implementation

```
Overview
```

Dubble Co.

Example

Implmentation

Properties

Jilett Joil

Summary

```
void bubbleSort(Item items[], int lo, int hi) {
    for (int i = hi; i > lo; i--) {
        bool swapped = false;
        for (int j = lo; j < i; j++) {
            if (gt(items[i], items[i + 1])) {
                swap(items, j, j + 1);
                swapped = true;
        if (!swapped) break;
```

Bubble Sort

Analysis

Motivation

Selection So

Example

Analysis Propertie

Insertion Sor

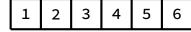
Shell Sort

Summary

Sorting Lis

Best case: Array is sorted

- Only a single pass required
- n-1 comparisons, no swaps
- Best-case time complexity: O(n)



Analysis

Worst case: Array is reverse-sorted

- n-1 passes required
 - First pass: n-1 comparisons
 - Second pass: n-2 comparisons

 - Final pass: 1 comparison
- Total comparisons: $(n-1) + (n-2) + ... + 1 = \frac{1}{2}n(n-1)$
- Every comparison leads to a swap $\Rightarrow \frac{1}{2}n(n-1)$ swaps
- Worst-case time complexity: $O(n^2)$



Bubble Sort Analysis

Analysis

Average-case time complexity: $O(n^2)$

• Can show empirically by generating random sequences and sorting them

Bubble Sort Properties

Motivation

Overview

Selection Sc

Bubble So

Implmentatio

Analysis Properties

.,....

Summary

Sorting List

Stable

Comparisons are between adjacent elements only Elements are only swapped if out of order

Adaptive

Bubble sort is $O(n^2)$ on average, O(n) if input array is sorted

In-place

Sorting is done within original array; does not use temporary arrays

.

Overviev

Selection S

Bubble So

Insertion Sort

Implementation
Analysis
Properties

Snell Sor

Summar

Sorting Lists

Method:

- Take first element and treat as sorted array (of length 1)
- Take next element and insert into sorted part of array so that order is preserved
 - This increases the length of the sorted part by one
- Repeat for remaining elements

Example

.

Overview

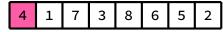
Insertion Sor

Example

Analysis

Shell Sort

Summary



Example

Overview

Example

4	1	7	3	8	6	5	2
1	4	7	3	8	6	5	2

Example

. .

Overview

Selection Sor

Incortion Cor

Example Example

Implementati

Properties Properties

Shall Sor

Summary

4	1	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	4	7	3	8	6	5	2

Example

.

Salaction Sor

Setection Sor

Insertion Sor

Example

Analysis

rioperties

_

Julilliary

4	1	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	3	4	7	8	6	5	2

Motivation

Overview

Selection 301

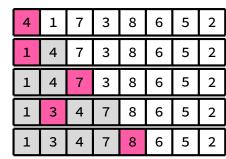
Insertion Sor

Example

Analysis

Shell Sor

Summary



Motivation

Selection 301

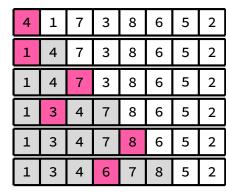
Insertion Sor

Example

Analysis

Shell Son

Summary



Motivation

Coloction Con

Setteetion Son

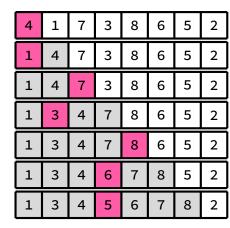
Insertion Sor

Example

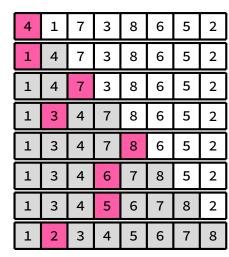
Analysis

Shell Son

Summary



Example



Motivation

Soloction Sou

Setection Sor

Insertion Soi

Example

Analysis

Shell Sor

Summary

4	1	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	4	7	3	8	6	5	2
1	3	4	7	8	6	5	2
1	3	4	7	8	6	5	2
1	3	4	6	7	8	5	2
1	3	4	5	6	7	8	2
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Insertion Sort C Implementation

Implementation

```
void insertionSort(Item items[], int lo, int hi) {
    for (int i = lo + 1; i <= hi; i++) {
        Item item = items[i];
        int i = i;
        for (; j > lo && lt(item, items[j - 1]); j--) {
            items[j] = items[j - 1];
        items[i] = item;
```

Analysis

MOLIVALIOI

Overview

Jetection Joi

Insertion Sc

Example

Analysis

Propertie

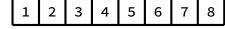
Shell Sor

Summar

Sorting List

Best case: Array is sorted

- Inserting each element requires one comparison
- n-1 comparisons
- Best-case time complexity: O(n)



Motivation

Overview

Dubble 301

Example

Analysis

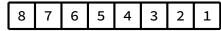
Properties

Shell So

Summar

Worst case: Array is reverse-sorted

- ullet Inserting i-th element requires i comparisons
 - Inserting index 1 element requires 1 comparison
 - Inserting index 2 element requires 2 comparisons
 - ..
- Total comparisons: $1 + 2 + ... + (n-1) = \frac{1}{2}n(n-1)$
- Worst-case time complexity: $O(n^2)$





Analysis

Insertion Sort

Average-case time complexity: $O(n^2)$

• Can show empirically by generating random sequences and sorting them

Insertion Sort Properties

Motivation

Overview

Selection So

Rupple 201

Example Implementation

Properties

Shell Sor

Summary

Sorting Lists

Stable

Elements are always inserted to the right of any equal elements

Adaptive

Insertion sort is $O(n^2)$ on average, O(n) if input array is sorted

In-place

Sorting is done within original array; does not use temporary arrays

Shell Sort

Overview

Selection So

Bubble Sc

Insertion Sc

Shell Sort

Evamnle

impleme

n .:

Cumman

Summary

Sorting Lists

Bubble sort and insertion sort really only consider *adjacent* elements.

If we make longer-distance exchanges, can we be more efficient?

What if we consider elements that are some distance apart?

Shell sort, invented by Donald Shell

Shell Sort

Overviev

Selection S

Bubble So

Insertion S

Shell Sort

Implementation
Analysis

Summary

Sorting Lis

Idea:

- An array is h-sorted if taking every h-th element yields a sorted array
- An h-sorted array is made up of $\frac{n}{h}$ interleaved sorted arrays
- ullet Shell sort: h-sort the array for progressively smaller h, ending with h=1

.

Overview

Selection S

Incortion S

Shell Sort

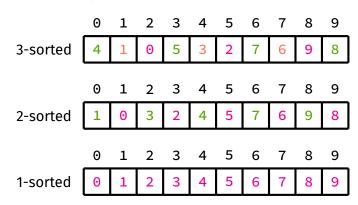
Implementa Analysis

Properties

Summary

Sorting Lists

Example of h-sorted arrays:



Shell Sort Example

Overview

Example

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	
unsorted	4	1	7	3	8	6	5	2	

Shell Sort Example

Motivation

Overview

Selection 30

Incortion So

Example

Analysis

Janimary

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
unsorted	4	1	7	3	8	6	5	2
h=3 passes	3			4			5	
		1			2			8
			6			7		
3-sorted	3	1	6	4	2	7	5	8

Shell Sort

Example

Motivation

Selection 30

Incombion Co.

11150101011

Example

Analysis

Summary

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
unsorted	4	1	7	3	8	6	5	2
h=3 passes	3			4			5	
		1			2			8
			6			7		
3-sorted	3	1	6	4	2	7	5	8
h=2 passes	2		3		5		6	
		1		4		7		8
2-sorted	2	1	3	4	5	7	6	8

Shell Sort Example

Motivation

Overview

Selection So

msertion 50

Example

Implementa

Properties

Summary

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
unsorted	4	1	7	3	8	6	5	2
h=3 passes	3			4			5	
		1			2			8
			6			7		
3-sorted	3	1	6	4	2	7	5	8
h=2 passes	2		3		5		6	
		1		4		7		8
2-sorted	2	1	3	4	5	7	6	8
h=1 pass	1	2	3	4	5	6	7	8

Shell Sort C Implementation

MOLIVALIOI

Selection So

. . . .

Insertion Soi

Shell Sort

Example Implementation

Analysis

Summary

· · · · · ·

```
void shellSort(Item items[], int lo, int hi) {
   int size = hi - lo + 1;
   // find appropriate h-value to start with
   int h;
   for (h = 1; h \le (size - 1) / 9; h = (3 * h) + 1);
   for (; h > 0; h /= 3) {
        for (int i = lo + h; i <= hi; i++) {
            Item item = items[i];
            int j = i;
            for (; j >= lo + h && lt(item, items[j - h]); j -= h) {
                items[i] = items[i - h];
            items[j] = item;
```

Analysis

Motivation

Overview

Selection 50

Incortion Sc

111361 (1011 30

Example

Implementat

Analysis

...,....

Summary

- Efficiency of shell sort depends on the h-sequence
- Effective h-sequences have been determined empirically
- Many h-sequences have been found to be $O(n^{\frac{3}{2}})$
 - For example: 1, 4, 13, 40, 121, 364, 1093, ...

•
$$h_{i+1} = 3h_i + 1$$

- Some h-sequences have been found to be $O(n^{\frac{4}{3}})$
 - For example: 1, 8, 23, 77, 281, 1073, 4193, ...

Shell Sort Properties

Motivation

Overview

selection 50

Insertion S

Shell Sort

Implementati

Properties

Summary

Sorting Lists

Unstable

Due to long-range swaps

Adaptive

Shell sort applies a generalisation of insertion sort (which is adaptive)

In-place

Sorting is done within original array; does not use temporary arrays

Summary of Elementary Sorts

Overview
Selection Sor
Bubble Sort
Insertion Sort
Shell Sort

Summary

	Tir	ne complex	Properties		
	Best	Average	Stable	Adaptive	
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	No
Bubble sort	O(n)	$O(n^2)$	$O(n^2)$	Yes	Yes
Insertion sort	O(n)	$O(n^2)$	$O(n^2)$	Yes	Yes
Shell sort	depends	depends	depends	No	Yes

COMP2521 23T3

Aside: Sorting Linked Lists

Overviev

Selection So

Insertion So

Shell Sort

Summar

Sorting Lists

Selection sort:

- Let L = original list, S = sorted list (initially empty)
- Repeat the following until L is empty:
 - Find the node V containing the largest value in L, and unlink it
 - Insert V at the front of S

Bubble sort:

- Traverse the list, comparing adjacent values
 - If value in current node is greater than value in next node, swap values
- Repeat the above until no swaps required in one traversal

Insertion sort:

- Let L = original list, S = sorted list (initially empty)
- For each node in *L*:
 - Insert the node into S in order

Aside: Sorting Linked Lists

Overviev

ection 5

Insertion S

Shell Sort

Summarv

Sorting Lists

Shell sort:

- Difficult to implement efficiently
- Can't access specific index in constant time
 - Have to traverse from the beginning

Motivatio

Selection So

Bubble 2011

Insertion So

311011 301

Sorting Lists

https://forms.office.com/r/aPF09YHZ3X

