COMP2521 19T0 lec10 cs2521@ jashankj@

Sorting

# COMP2521 19T0 Week 6, Tuesday: Order! Order! (I)

Jashank Jeremy jashank.jeremy@unsw.edu.au

basic sorting algorithms more sorting algorithms Sortin

MYEXPERIENCE now open! myexperience.unsw.edu.au

PRAC EXAM #1 results look pretty good a majority of people passed the exam! no problem required >10 LoC; if you just threw code at the wall, consider a different strategy next time.

ASSIGNMENT 2 part 1 is underway! views due **20 Jan 2019**, no extensions. view dryruns out now — how does your code do? hunt spec to be released during week07tue lecture

#### COMP2521 19T0 lec10

cs2521@ jashankj@

### Sorting

Problem
Formally
Concretely
Complexity
Elementary S
Bubble

Bubble E Selection

Insertio Shell

# Sorting



cs2521@

jashankj@

#### Sortin

#### Problem

Formally Concretely Complexity

Bubble

Bubble EE

Insertio

## Sorting: The Problem

## Sorting

... arranging a collection of items in order,



Sortin

#### Problem

Concretely Complexity

Elementary S

Bubble EE

Selection Insertion

Shell

## Sorting: The Problem

## Sorting

... arranging a collection of items in order, ... based on some property of an item (a 'key'),



cs2521@ iashanki@

Sorting

#### Problem

Concretely Complexity

Elementary S

Bubble EE

Insertion

Shell

## Sorting: The Problem

## Sorting

... arranging a collection of items in order, ... based on some property of an item (a 'key'), ... using an ordering relation on that property. Elementary Sor Bubble

Bubble EE Selection Insertion

Shell

## Sorting: The Problem

Why? What? Where?

### Why are we interested?

- speeds up subsequent searches;
- arranges data in useful ways (human- or otherwise)
   ... e.q., a list of students in a tutorial
- provides useful intermediate for other algorithms
   ... e.q., duplicate detection/removal; DBMS operations

Why? What? Where?

Why are we interested? Problem

speeds up subsequent searches;

arranges data in useful ways (human- or otherwise)

... e.a., a list of students in a tutorial

 provides useful intermediate for other algorithms ... e.g., duplicate detection/removal; DBMS operations

### What contexts?

- arrays, linked lists (in-memory, internal)
- files (external, on-disk)
- ... distributed across a network (map/reduce)

Why? What? Where?

Why are we interested? Problem

speeds up subsequent searches;

arranges data in useful ways (human- or otherwise)

... e.a., a list of students in a tutorial

 provides useful intermediate for other algorithms ... e.g., duplicate detection/removal; DBMS operations

### What contexts?

- arrays, linked lists (in-memory, internal)
- files (external, on-disk)
- ... distributed across a network (map/reduce)

We'll focus on sorting arrays (and lists)

Elementary S

Bubble El

Selection

Shell

## Sorting: The Problem

(More) 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 same values  $a'[lo] \le a'[lo+1] \le a'[lo+2] \le \cdots \le a'[hi]$ 

#### COMP2521 19T0 lec10

cs2521@ iashanki@

Sorting

#### Formally

Complexity

Elementary S

Bubble F

Selection

Shell

## Sorting: The Problem

Properties: Stability, Adaptive, In-Place

Properties: stable sorts

let x = a[i], y = a[j], where  $\mathrm{KEY}(x) \equiv \mathrm{KEY}(y)$  let the 'precedes' relation be that index  $i \leq j$ . if x 'precedes' y in a, then x 'precedes' y in a'

Elementary S

Bubble E

Insertion

Shell

## Sorting: The Problem

Properties: Stability, Adaptive, In-Place

Properties: stable sorts

let x = a[i], y = a[j], where  $\text{KEY}(x) \equiv \text{KEY}(y)$  let the 'precedes' relation be that index  $i \leq j$ . if x 'precedes' y in a, then x 'precedes' y in a'

Properties: adaptive sorts
where the algorithm's behaviour or performance
is affected by the input data —
that best/average/worst case performance differs
... and can take advantage of existing order

Properties: Stability, Adaptive, In-Place

Sorting

### Formally

Concretely Complexity Elementary So

Bubble El Selection

Insertio Shell

### Properties: stable sorts

let x=a[i], y=a[j], where  $\mathrm{KEY}(x) \equiv \mathrm{KEY}(y)$  let the 'precedes' relation be that index  $i \leq j$ . if x 'precedes' y in a, then x 'precedes' y in a'

Properties: adaptive sorts
where the algorithm's behaviour or performance
is affected by the input data —
that best/average/worst case performance differs
... and can take advantage of existing order

Properties: in-place sorts
sort data within original structure,
using only a constant additional amount of space

## A Concrete Framework

```
cs2521@
jashankj@
```

```
Problem
Formall
```

```
Concretely
Complexity
Elementary Son
```

Bubble EE Selection Insertion

```
// we deal with generic `Item's
typedef int Item:
// abstractions to hide details of items
#define kev(A) (A)
#define less(A,B) (kev(A) < kev(B))</pre>
\#define\ eq(A,B)\ (kev(A) == kev(B))
\#define swap(A,B) { Item t: t = A: A = B: B = t: }
#define cas(A,B) { if (less (A, B)) swap (A, B); }
// cas == Compare And Swap, often hardware assisted
/// Sort a slice of an array of Items.
void sort (Item a[], int lo, int hi);
/// Check for sortedness (to validate functions).
bool sorted p (Item a[], int lo, int hi);
```

#### Problem Formally Concretely

Complexity

Bubble

Bubble EE Selection

Selection Insertion 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 less(A, B) (strcmp(key(A), key(B)) < 0)
#define swap(A,B) { Item t; t = A; A = B; B = t; }
// ... works because struct assignment works in C</pre>
```

## Complexity of Sorting Algorithms

cs2521@ jashankj@

## Sortin

Formally Concrete

#### Complexity Elementary:

Bubble EE Selection Insertion

## In analysing sorting algorithms:

- N: the number of items (hi lo + 1)
- C: the number of comparisons between items
- $\cdot$  S: the number of times items are swapped

(We usually aim to minimise C and S.)

## Complexity of Sorting Algorithms

### cs2521@ iashanki@

Complexity

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-ascending order:  $a[lo] \le a[lo + 1] \le \cdots \le a[hi]$
- sorted-descending order:  $a[lo] > a[lo + 1] > \cdots > a[hi]$

### Sortin

Formal

#### Elementary Sorts

Bubble

Bubble EE Selection

Chall

- Bubble Sort (oblivious and early-exit)
- Selection Sort
- Insertion Sort

Sorting

Problen

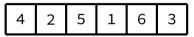
Concretely

Complex Elementar

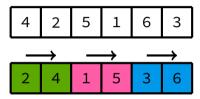
Bubble

Bubble EE Selection

Shell



#### Bubble



Sorting

Problem

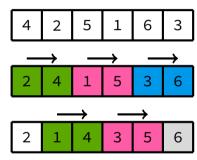
Concrete! Complexi

Complex

Bubble

Selection

Shell



Sorting

Problem

Formally

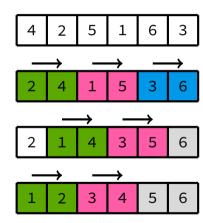
Complex

Element

Bubble

Selection

Insertior Shell



C Implementation — Oblivious

```
cs2521@
iashanki@
```

```
Bubble
```

```
void sort bubble (Item items[], size t lo, size t hi)
    for (size t i = hi; i > lo; i--)
        for (size t i = lo + 1; i <= i; i++)
            if (less (items[i], items[j - 1]))
                swap idx (items, i, i - 1);
```

Bubble

• Outer loop ( $C_0$ )

## Sorting

Formally

Complexit

#### Bubble

Selection Insertion • Outer loop ( $C_0$ )

```
for (size_t i = n - 1; i > 0; i--)
```

## Sorting

Formally Concrete

Complex Elementar Bubble

#### Bubble

Selection Insertion • Outer loop  $(C_0)$   $\Rightarrow N$ for (size\_t i = n - 1; i > 0; i--)

## Sorting

Formally Concretel

Complexity
Elementary So
Bubble

#### Bubble

Selection Insertion • Outer loop  $(C_0)$   $\Rightarrow N$ 

for (size\_t i = n - 1; i > 0; i--)

• Inner loop ( $C_1$ )

### Sorting

Formally

Concretely Complexity Elementary Sor

### Bubble

Selection Insertion

```
• Outer loop (C_0) \Rightarrow N
for (size_t i = n - 1; i > 0; i--)
```

• Inner loop  $(C_1)$ 

```
for (size_t j = 1; j <= i; j++)</pre>
```

## Bubble

• Outer loop  $(C_0) \Rightarrow N$ 

```
for (size t i = n - 1; i > 0; i--)
```

• Inner loop  $(C_1)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 - 1$ for (size t i = 1; i <= i; i++)

#### cs2521@ iashanki@

Bubble

• Outer loop  $(C_0) \Rightarrow N$ 

- for (size\_t i = n 1; i > 0; i--)
- Inner loop  $(C_1)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$ for (size t i = 1; i <= i; i++)
- Comparisons  $(C_2)$

Bubble

- Outer loop  $(C_0) \Rightarrow N$ for (size t i = n - 1; i > 0; i--)
- Inner loop  $(C_1)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$ for (size t i = 1; i <= i; i++)
- Comparisons  $(C_2)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$

### orting

Concretely
Complexity
Elementary So

### Bubble Bubble B

Selection Insertion • Outer loop  $(C_0) \Rightarrow N$ 

Outer toop 
$$(C_0) \Rightarrow N$$
  
for (size\_t i = n - 1; i > 0; i--)

- Inner loop ( $C_1$ )  $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$  for (size\_t j = 1; j <= i; j++)
- Comparisons ( $C_2$ )  $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$
- Swaps  $(C_3)$  (assuming the worst case: we always have to swap)

Bubble

• Outer loop  $(C_0) \Rightarrow N$ for (size t i = n - 1; i > 0; i--)

- Inner loop  $(C_1)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$ for (size t i = 1: i <= i: i++)
- Comparisons  $(C_2)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$
- Swaps  $(C_3)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$ (assuming the worst case: we always have to swap)

### Sorting

Formally Concretel

Complexity Elementary S

#### Bubble Bubble EE

Selection Insertion Shell C Implementation — Analysis

- Outer loop  $(C_0)$   $\Rightarrow N$ for (size\_t i = n - 1; i > 0; i--)
- Inner loop ( $C_1$ )  $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$  for (size\_t j = 1; j <= i; j++)
- Comparisons ( $C_2$ )  $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$
- Swaps ( $C_3$ )  $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$  (assuming the worst case: we *always* have to swap)

$$T(n) = NC_0 + \left(\frac{N^2 + N}{2} - 1\right)C_1 + \frac{N^2 - N}{2}C_2 + \frac{N^2 - N}{2}C_3$$

Bubble

C Implementation — Analysis

- Outer loop  $(C_0) \Rightarrow N$ for (size t i = n - 1; i > 0; i--)
- Inner loop  $(C_1)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 2 = (N^2 + N)/2 1$ for (size t i = 1: i <= i: i++)
- Comparisons  $(C_2)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$
- Swaps  $(C_3)$   $\Rightarrow N + (N-1) + (N-2) + \cdots + 1 + 0 = (N^2 N)/2$ (assuming the worst case: we always have to swap)

$$T(n) = NC_0 + \left(\frac{N^2 + N}{2} - 1\right)C_1 + \frac{N^2 - N}{2}C_2 + \frac{N^2 - N}{2}C_3$$

$$\Rightarrow O(N^2)$$



## Bubble Sort Summary

Droblom

Formally
Concretely
Complexity

Bubble

Bubble

Bubble EE Selection

Shell

How many steps does it take to sort a collection of *N* elements?



cs2521@ jashankj@

## **Bubble Sort**

Summary

Problem
Formally
Concretely
Complexity

#### Bubble

Selection Insertion How many steps does it take to sort a collection of N elements?

For the ith iteration, we have N-i comparisons and best 0, worst N-i swaps (depending on sortedness.)

Summary

#### How many steps does it take to sort a collection of N elements?

For the *i*th iteration, we have N-i comparisons and best 0, worst N-i swaps (depending on sortedness.)

Bubble sort is  $O(n^2)$ . Stable, in-place, non-adaptive.

Bubble



### Improving Bubble Sort

cs2521@ jashankj@

Sortin

Proble

Concrete

Complexity

Elementary So

Bubble

Bubble EE

Selectio

Insertion Shell 'oblivious' bubble-sort continues, even if the list is sorted so what's a better stopping-case than 'we ran out of array'?



## cs2521@

#### Sorting

Problem

Formally

Complexity

Elementary So

Bubble

#### Bubble EE

Insertio

### Improving Bubble Sort

'oblivious' bubble-sort continues, even if the list is sorted so what's a better stopping-case than 'we ran out of array'?

if we complete a whole pass without swaps, we're ordered! this is bubble sort with early exit, or adaptive bubble sort

## **Adaptive Bubble Sort**

C Implementation — Adaptive

```
iashanki@
```

**Bubble FF** 

```
void sort bubble ee (Item items[], size t lo, size t hi)
   bool no swaps = false:
   for (size t i = hi; i > lo && !no swaps; i--) {
        no swaps = true;
        for (size_t j = lo + 1; j <= i; j++)
            if (less (items[i], items[i - 1])) {
                swap_idx (items, j, j - 1);
                no_swaps = false;
```



cs2521@ jashankj@

## Adaptive Bubble Sort

Analysis; Summary

#### Problem

Problem

Concrete

Complexity

Etementary S

#### Bubble EE

Bubble E

Insertion

How many steps does it take to sort a collection of *N* elements?

Analysis; Summary

#### orting

Problen

Concrete

Complexity

Elementary So

Bubble FF

#### Dubbte t

Insertio Shell How many steps does it take to sort a collection of *N* elements?

Each traversal does N comparisons.

Best case: exit after one iteration
(if the collection is already sorted.)

Worst case: N traversals still necessary.

$$T_{\text{worst}}(N) = N - 1 + N - 2 + \dots + 1 \approx N^2$$
  
 $T_{\text{best}}(N) = N$ 

## Adaptive Bubble Sort

Analysis; Summary

#### Sorting

Problen

Concret

Elomontary S

Eternemary 50

#### Bubble EE

Selectio

Insertion Shell How many steps does it take to sort a collection of N elements?

Each traversal does N comparisons.

Best case: exit after one iteration
(if the collection is already sorted.)

Worst case: N traversals still necessary.

$$T_{\text{worst}}(N) = N - 1 + N - 2 + \dots + 1 \approx N^2$$
  
 $T_{\text{best}}(N) = N$ 

Bubble-sort with early exit is still  $O(N^2)$ . Stable, in-place, adaptive (!).

#### cs2521@ jashankj@

Sorting

Problen

Formal

Comple

Flammatan

Dubbbo

Bubble I

Selection

Shall

Select the smallest element. Swap it with the first position.

Select the next smallest element. Swap it with the second position.

... continue until sorted!



cs2521@ jashankj@

Sorting

Proble

Formall

Concret

Flomentanic

Dubbb

Bubble

#### Selection

Insertion

4 1 7

\_\_\_\_\_

\_\_\_

\_

\_

6

\_

cs2521@ jashankj@

Sorting

Proble

Formally

Concrete

Comple

Elementa

Bubble

Bubble

#### Selection

Insertior

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

cs2521@ jashankj@

Sortin

Proble

Formally

Concrete

Complexit

Elementary

Dubbbo

Selection

Selectio

Insertion

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

cs2521@ jashankj@

Sorting

Problem

Formally

Concrete

Elementary

Dubble

Bubble

Selection

Insertio

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

cs2521@ jashankj@

Sorting

Problem

Formally

Complexi

Elementary S

pubble

Selection

Setectio

Shell

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

cs2521@ jashankj@

Sorting Problem

Formally Concretel

Elementary S

Bubble

Selection

Shell

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

cs2521@ jashankj@

Selection

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

cs2521@ jashankj@

Sorting

Problem

Formally

Complexi

Elementary S

Bubble

Selection

Shell

Į	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
ſ	1	2	3	4	8	6	5	7
į	1	2	3	4	5	6	8	7
	1	2	3	4	5	6	8	7

cs2521@ jashankj@

Sorting Problem

Problem Formally Concrete

Elementary :

Bubble

Selection

Shell

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
1	2	3	4	5	6	7	8

#### Selection Sort C Implementation

Selection

```
void sort selection (Item items[], size t lo, size t hi)
    for (size t i = lo; i < hi; i++) {</pre>
        size t low = i:
        for (size t j = i + 1; j <= hi; j++)
            if (less (items[i], items[low]))
                low = i:
        swap idx (items, i, low);
```



cs2521@ iashanki@

### **Selection Sort**

Analysis; Summary

#### Problem

Formally

Concrete

FlomentanyCo

2101110111011

Dubbb F

Bubble

Selection

Insertio

How many steps does it take to sort a collection of *N* elements?

... picking the minimum of a sequence of N elements: N steps. ... inserting at the right place: 1.

Analysis; Summary

Selection

How many steps does it take to sort a collection of N elements?

... picking the minimum of a sequence of N elements: N steps. ... inserting at the right place: 1.

$$T(N) = N + (N-1) + (N-2) + \dots + 1 = \frac{1}{2}N(N+1)$$

Analysis; Summary

#### Sorting

Problem

Formall

Comple

Elementary So

Bubble

Bubble

Selection

chall

How many steps does it take to sort a collection of *N* elements?

 $\dots$  picking the minimum of a sequence of N elements: N steps.

... inserting at the right place: 1.

$$T(N) = N + (N-1) + (N-2) + \dots + 1 = \frac{1}{2}N(N+1)$$

Selection sort is  $O(N^2)$ . Unstable, in-place, oblivious.s



cs2521@ jashankj@

Sorting

Problem Formall

Concret

Complexity

Elementary So

Bubble F

Selection

Insertion

Take the first element, insert into the first position.

This starts our 'sorted sublist'.

Take the next element.
Insert it into the sorted sublist in the right spot!

Repeat until sorted!

cs2521@ jashankj@

Sortin

Proble

Formall

Concret

Elomonts

Bubble

Bubble E

#### Insertion

SHELL



cs2521@ jashankj@

Sorting

Proble

Formall

Concrete

Complexi

Etementar

Rubble F

Selection

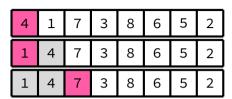
Insertion

-----

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

cs2521@ jashankj@

Insertion



cs2521@ jashankj@

Formally
Concretely
Complexity
Elementary Sort

Insertion

 4
 1
 7
 3
 8
 6
 5

 1
 4
 7
 3
 8
 6
 5

 1
 4
 7
 3
 8
 6
 5

 1
 3
 4
 7
 8
 6
 5

cs2521@ jashankj@

Problem
Formally
Concretely
Complexity
Elementary Sort

Insertion

 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

cs2521@ jashankj@

Sorting
Problem
Formally
Concretely
Complexity
Elementary Sort
Bubble

Insertion

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

cs2521@ jashankj@

Forting

Problem

Formally

Concretely

Complexity

Elementary Sort

Bubble

Insertion

```
cs2521@
jashankj@
```

Formally
Concretely
Complexity
Elementary Sort

Insertion

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

cs2521@ jashankj@

Sorting
Problem
Formally
Concretely
Complexity
Elementary Sort
Bubble

Insertion

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

```
void sort insertion (Item items[], size t lo, size t hi)
            for (size t i = lo + 1; i <= hi; i++) {
                Item item = items[i]:
Insertion
                size t i = i:
                for (/* j */; j > lo; j--) {
                    if (! less (item, items[i - 1])) break;
                    items[j] = items[j - 1];
                items[j] = item;
```

Insertion

How many steps does it take to sort a collection of N elements?

For every element (of N elements): 1 step to pick an element; insert into a N' < N sequence: up to N steps.

$$T_{\text{worst}}(N) = 1 + 2 + \dots + N = \frac{N}{2}(N+1)$$
  
 $T_{\text{best}}(N) = 1 + 1 + \dots + 1 = N$ 

Insertion

How many steps does it take to sort a collection of N elements?

For every element (of N elements): 1 step to pick an element: insert into a N' < N sequence: up to N steps.

$$T_{\text{worst}}(N) = 1 + 2 + \dots + N = \frac{N}{2}(N+1)$$
  
 $T_{\text{best}}(N) = 1 + 1 + \dots + 1 = N$ 

Insertion sort is  $O(N^2)$ . Stable, in-place, adaptive.



cs2521@ jashankj@

#### **Shell Sort**

One Small Swap for a Sort  $\dots$ 

Formally
Concretely
Complexity

Elementary So

Bubble E

Insertion

Shell

Bubble- and Insertion-Sort really only consider *adjacent* elements.

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



Problem Formally

Concrete

Elementary Soi

Bubble

Selection

Shell

#### Shell Sort

One Small Swap for a Sort ...

Bubble- 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? ... sort sublists of mod-h indices, for decreasing h until h=1?

### Shell Sort

cs2521@ jashankj@

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

Sorting Problem

Formally Concrete

Complexity Elementary Sor

Bubble

Selection

Insertior

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

Problem
Formally
Concretely
Complexity

Elementary Sor Bubble Bubble EE

Selection Insertion

	[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

## Forblem Formally Concretely Complexity Elementary Sorts Bubble

Bubble Bubble EE Selection

Insertion

Sneu

	[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		

Forming

Problem

Formally

Concretely

Complexity

Elementary Sorts

Bubble

	[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**

cs2521@ jashankj@

Forblem
Formally
Concretely
Complexity
Elementary Sorts
Bubble
Bubble EE

	[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	

Forblem

Formally

Concretely

Complexity

Elementary Sorts

Bubble

Bubble EE

	[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

Forblem

Formally

Concretely

Complexity

Elementary Sorts

Bubble

Bubble EE

	[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

Forblem
Formally
Concretely
Complexity
Elementary Sorts
Bubble
Bubble EE

	[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

```
void sort_shell (Item items[], size_t lo, size_t hi)
             size t h:
             for (h = 1; h \le (n - 1) / 9; h = (3 * h) + 1);
             for (/* h */; h > 0; h /= 3) {
Shell
                 // when `h' = 1, this is an insertion sort.
                 for (size t i = h; i < n; i++) {
                     Item item = items[i];
                     size t i = i:
                     for (/* i */; i >= h &\& item < items[i - h]; i -= h)
                          items[j] = items[j - h];
                     items[i] = item:
```



cs2521@ iashanki@

#### Shell Sort

... One Giant Leap for Complexity?

# Forking Problem Formally Concretely Complexity Elementary Sort Bubble Bubble EE Selection

Shell

The exact complexity-class depends on the h-sequence. Probably safe to assume that  $O (\leq n^2)$ , because otherwise what's the point? Lots of h-value sequences are  $O \left( n^{\frac{3}{2}} \right)$ .

No 'general' analysis exists.

#### Shell Sort

... One Giant Leap for Complexity?

Problem
Formally
Concretely
Complexity
Elementary Sort
Bubble
Bubble EE
Selection

Shell

The exact complexity-class depends on the h-sequence. Probably safe to assume that  $O (\leq n^2)$ , because otherwise what's the point? Lots of h-value sequences are  $O \left( n^{\frac{3}{2}} \right)$ .

No 'general' analysis exists.

Shell Sort is  $O (\leq n^2)$ . It is unstable, adaptive, in-place.

#### cs2521@ iashanki@

Shell

Bubble traverse list: if curr > next, swap.

Selection delete selected element, insert at head of sorted list.

Insertion delete first element, do order-preserving insertion.

Shell (screaming)