CS1113
Sorting

Lecturer:

Professor Barry O'Sullivan

Office: 2.65, Western Gateway Building

email: *b.osullivan@cs.ucc.ie*

http://osullivan.ucc.ie/teaching/cs1113/

Algorithms for sorting data

Bubble sort

Summation

Sorting a sequence of items

The task of re-arranging some sequence of items into a specific order is one of the most common sub-tasks in many computer applications. E.g.:

- books listed in order of total sales on Amazon
- web pages listed in order of page rank on Google
- items listed in order of recommendation score in recommender systems
- teams in a sports league listed in order of points
- names in an address book being ordered alphabetically

Re-arranging a sequence into some order is called sorting

The are two main issues:
(i) how to do it, and (ii) how to do it efficiently.

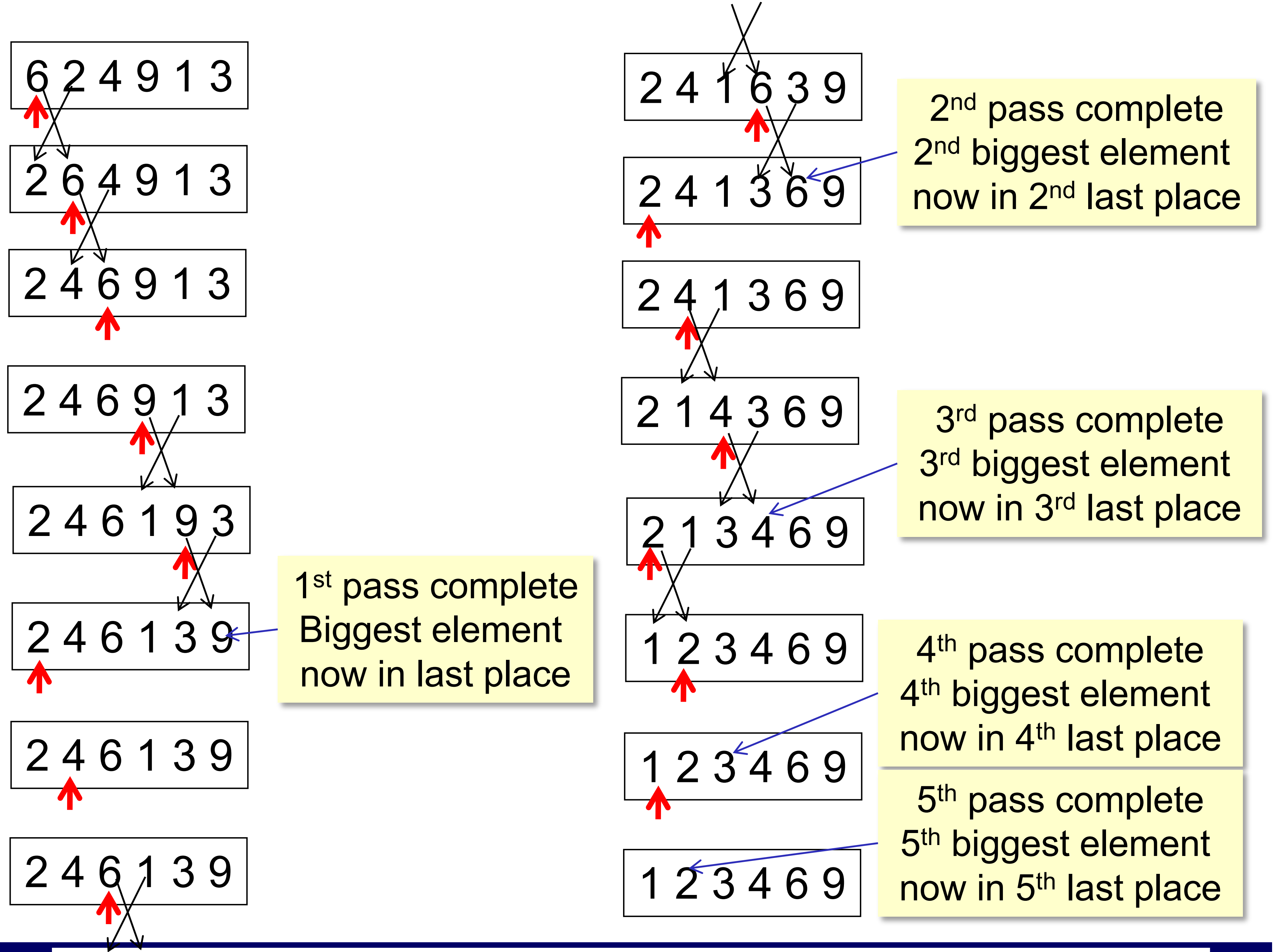
Example

Sort the sequence 6,2,4,9,1,3 in increasing order.

initial sequence:	6 2 4 9 1 3
target sequence:	1 2 3 4 6 9

First method

We will pass through the sequence from front to back swapping each pair that are in the wrong order. This will push the biggest number to the end. We then repeat until all numbers are in the right place.


6 2 4 9 1 3
2 6 4 9 1 3
2 4 6 9 1 3
2 4 6 9 1 3
2 4 6 1 9 3
2 4 6 1 3 9
1st pass complete
Biggest element
now in last place
2 4 6 1 3 9
2 4 6 1 3 9
2 4 1 6 3 9
2 4 1 3 6 9
2nd pass complete
2nd biggest element
now in 2nd last place
2 4 1 3 6 9
2 1 4 3 6 9
2 1 3 4 6 9
3rd pass complete
3rd biggest element
now in 3rd last place
1 2 3 4 6 9
4th pass complete
4th biggest element
now in 4th last place
1 2 3 4 6 9
5th pass complete
5th biggest element
now in 5th last place
1 2 3 4 6 9

Algorithm: Bubble Sort

Input: a sequence $x_1, x_2, \ldots, x_n$ (where $n > 1$)

Output: a sequence $y_1, y_2, \ldots, y_n$, which is an ordering of the x_i

1. for each i from 1 to $n-1$
2.   for each j from 1 to $n-i$
3.   if $x_j > x_{j+1}$
4.   then swap the positions of x_j and x_{j+1}

1. each time round outer loop, we will push i^{th} biggest element into i^{th} last place
2. so must consider each of the first (n-i) positions in turn
3. if the current element is bigger than the one that comes after it
4. swap the current element with the one that comes after it

Bubble Sort run-time is O(n^2)

<u>Proof</u>

The outer loop considers each value of i from 1 to n-1, and calls the inner loop each time.

Each time the inner loop is called, the value of i is fixed, and we then make (n-i) comparisons (and at most (n-i) swaps).

The number of comparisons is then

$(n-1) + (n-2) + (n-3) + ... + (n-(n-2)) + (n-(n-1))$

$= (n-1) + (n-2) + (n-3) + ... + (2) + (1)$

$= (n-1)*n/2$ [proof of this on next slides]

$= (n^2-n)/2$

$= n^2/2 – n/2$

which is O(n^2) [by the result proved in previous lecture, since we have a polynomial with n^2 as the highest power of n]

Summation

Often when we are analysing algorithms, or analysing memory requirements (and even specifying problems), we will want to write down a series of numbers or variables which have to be added together. It is tedious to write out the series each time.

Instead, if there is an easy pattern to the series, we will use a shorthand based on the Greek capital letter sigma:

$$\sum_{i=low}^{high} pattern$$

means add together all the values that match the pattern, where *i* ranges from *low* to *high*. *i* is called the index of the summation

$$\sum_{i=1}^{n} i = 1+2+3+\ldots+(n-1)+n$$

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + x_3 + \ldots + x_{n-1} + x_n$$

Examples

Expand the following:

$$\sum_{k=1}^{5} 1/k$$

$$\sum_{j=0}^{8} y_j$$

$$\sum_{i=0}^{n} a_i x^i$$

$$\sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i$$

What are the values of the following?

$$\sum_{k=1}^{8} k$$

$$\sum_{i=1}^{4} i^2$$

$$\sum_{i=1}^{n} i = \frac{1}{2} * n * (n+1)$$

Proof

Expand the left hand side to get 1+2+3+...+(n-1)+n

Add that expression to itself, but write it out backwards underneath the first version, aligning the values, and then add each aligned pair:

$$\begin{array}{lllllll}
1 & +2 & +3 & + \ldots & +(n-1) & + \ n \\
+ \ n & +(n-1) & +(n-2) & + \ldots & + \ 2 & + \ 1 \\
\hline
(n+1) & +(n+1) & +(n+1) & + \ldots & +(n+1) & +(n+1)
\end{array}$$

There are n of these terms, so this sum gives n*(n+1).

But we had to double our original series to get this, so our original expression must be n*(n+1)/2

Next lecture ...

insertion sort

induction