Quadratic-time sorting Overview		
Sorting		
Sorting Problem		
Input: An array of comparable elements		
Output : The same elements, sorted in ascending order		
• One of the most well-studied algorithmic problems		
 Has lots of practical applications 		
 You should already know a few algorithms. 		
SI 335 (USNA) Unit 2	Spring 2014	1 / 21
Quadratic-time sorting Overview		

SelectionSort

SI 335 (USNA)

Unit 2

Spring 2014 2 / 21

```
Quadratic-time sorting Overview

InsertionSort

def insertionSort(A):

    for i in range(1, len(A)):

        j = i - 1

    while j >= 0 and A[j] > A[j+1]:

        swap(A, j, j+1)

        j = j - 1

    Stas (USNA) Uni 2 Spring 2014 3/21
```

	Quadratic-time sorting Overview	
Common Features	i i i i i i i i i i i i i i i i i i i	
It's useful to look for la	arger patterns in algorithm desig	sn.
Both InsertionSort and	SelectionSort build up a sorted a	array one element
at a time, in the follow	-	and y one clement
	ment in the unsorted part of the a	arrav
	element into the sorted part of the	-
	element into the solted part of t	ne allay
For both algorithms, o is "hard" $(O(n)$ time).	ne of these is "easy" (constant ti Which ones?	me) and the other
SI 335 (USNA)	Unit 2	Spring 2014 4 / 21
	0.112	Opting 2011 1 / 21
	Quadratic-time sorting Loop analysis with summation	IS
Analysis of Selecti	onSort	
5		
Each loop has $O(n)$ ite	erations, so the total cost is $O(n^2)$	²).
What about a big-Θ b	ound?	
SI 335 (USNA)	Unit 2	Spring 2014 5 / 21
	Quadratic-time sorting Loop analysis with summation	15
Arithmetic Series		

An arithmetic series is one where consecutive terms differ by a constant.

General formula:
$$\sum_{i=0}^{m} (a+bi) = \frac{(m+1)(2a+bm)}{2}$$

So the worst-case of SelectionSort is

This is $\Theta(n^2)$, or **quadratic time**.

Quadratic-time sorting Worst-case family of examples
Worst-Case Family
Why can't we analyze InsertionSort in the same way?
We need a family of examples , of arbitrarily large size, that demonstrate the worst case.
Worst-case for InsertionSort:
Worst-case cost:
SI 335 (USNA) Unit 2 Spring 2014 7 / 21
Quadratic-time sorting Worst-case family of examples
SelectionSort (Recursive Version)
<pre>def selectionSortRec(A, start=0): if (start < len(A) - 1): m = minIndex(A, start) swap(A, start, m) selectionSortRec(A, start + 1)</pre>
minIndex

```
def minIndex(A, start=0):
    if start >= len(A) - 1:
        return start
    else:
        m = minIndex(A, start+1)
        if A[start] < A[m]:
            return start
        else:
            return m
    SI 335 (USNA) Unit 2 Spring 2014 8 / 21
</pre>
```

Quadratic-time sorting Recursive analysis

Analysis of minIndex

Let T(n) be the worst-case number of operations for a size-*n* input array.

We need a **recurrence relation** to define T(n):

$$T(n) = \begin{cases} 1, & n \leq 1 \\ 4 + T(n-1), & n \geq 2 \end{cases}$$

Solving the recurrence:

SI 335 (USNA)

Unit 2

Quadra	atic-time sorting Recursive analysis		
Analysis of recursive S	SelectionSort		
Let $S(n)$ be the worst-case	for SelectionSort		
What is the recurrence?			
SI 335 (USNA)	Unit 2	Spring 2014	10 / 21
	MergeSort Paradigm		
Divide and Conquer			
A new Algorithm Design	Paradigm: Divide and Conq	uer	
Works in three steps:			
Break the problem into	o similar subproblems		
② Solve each of the subp	problems recursively		
③ Combine the results to	o solve the original problem.		
MorgoSort and BinarySoard	th both follow this paradigm.		
(How do they approach eac			
SI 335 (USNA)	Unit 2	Spring 2014	11 / 21
	MergeSort Paradigm		
MergeSort			
<pre>def mergeSort(A):</pre>			
if len(A) <= 1:			
return A else:			

SI 335 (USNA)

m = len(A) // 2
B = A[0 : m]
C = A[m : len(A)]

mergeSort(C)
A[:] = merge(B, C)

mergeSort(B)

Unit 2

	MergeSort Paradigm		
Merge			
<pre>def merge(B, C): A = [] i, j = 0, 0 while i < len(B) if B[i] <= (</pre>	<pre>d(B[i]) 1 d(C[j]) 1): i])):</pre>		
SI 335 (USNA)	Unit 2	Spring 2014	13 / 21

Analysis of Merge

Each while loop has constant cost.

So we just need the total number of iterations through every loop.

	Lower bound	Upper bound	Exact
Loop 1	min(<i>a</i> , <i>b</i>)	a+b	
Loop 2	0	а	
Loop 3	0	Ь	
Total	min(<i>a</i> , <i>b</i>)	2(a+b)	

a is the size of A and b is the size of B.

SI 335 (USNA)

Unit 2

Spring 2014 14 / 21

	MergeSort	Analysis		
Analysis of MergeSort				
SI 335 (USNA)	Un	it 2	Spring 2014	15 / 21

	Lower Bound for Sorting	τ		
		,		
Complexity of S	Sorting			
Algorithms we have	e seen so far			
Algorithms we have				
	Sort SelectionSort	Worst-case cost $\Theta(n^2)$		
	InsertionSort	$\Theta(n^2)$		
	MergeSort	$\Theta(n \log n)$		
	HeapSort	$\Theta(n \log n)$		
Million dollar que	e stion : Can we do	b better than $\Theta(n)$	og n)?	
SI 335 (USNA)	l	Jnit 2	Spring 2014	16 / 21
	Lower Bound for Sorting	3		
Comparison Mo	odel			
Elements in the inp	out array can only	, be accessed in tw	0 Ways.	
Lienients in the inp	out unity cun only			
Moving them			o ways.	
	(swap, copy, etc.))	o wayo.	
)	o wayo.	
• Comparing tw Every sorting algor	(swap, copy, etc.) to of them (<, >, rithm we have see) =, etc.) en uses this model.		
 Comparing tw Every sorting algorithm It is a very general 	(swap, copy, etc.) to of them (<, >, rithm we have see) =, etc.) en uses this model.		
• Comparing tw Every sorting algorithm It is a very generation anything else.	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin) =, etc.) en uses this model. g strings or integer		
 Comparing tw Every sorting algorithm of the second se	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin) =, etc.) en uses this model. g strings or integer		
• Comparing tw Every sorting algorithm It is a very genera anything else.	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin) =, etc.) en uses this model. g strings or integer		
• Comparing tw Every sorting algorithm It is a very generation anything else.	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin) =, etc.) en uses this model. g strings or integer		
 Comparing tw Every sorting algorithm of the second se	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algorithm of the second se	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer		17 / 21
 Comparing tw Every sorting algorithm of the second se	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algoring to a very general anything else. What operations and SI 335 (USNA) 	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algoring to a very general anything else. What operations and state of the state o	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algoring to a very general anything else. What operations and SI 335 (USNA) 	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algoring to a very general anything else. What operations and SI 335 (USNA) 	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or	17 / 21
 Comparing tw Every sorting algoring to a very general anything else. What operations an SI 335 (USNA) Permutations 	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in) =, etc.) en uses this model. g strings or integer this model?	rs or floats or Spring 2014	17 / 21
 Comparing tw Every sorting algorithm is a very general anything else. What operations and 	(swap, copy, etc.) o of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in Lower Bound for Sorting) =, etc.) en uses this model. g strings or integer this model? J ^{nit 2}	s or floats or Spring 2014	17 / 21
 Comparing tw Every sorting algoring the second second	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in Lower Bound for Sorting Lower Bound for Sorting $n! = n \times (n - 1)$ comparison-based s) =, etc.) en uses this model. g strings or integer this model? Junit 2 Junit 2 s	s or floats or Spring 2014 e elements? 2 × 1.	
 Comparing tw Every sorting algor It is a very genera anything else. What operations an SI 335 (USNA) Permutations How many ordering <i>n</i> factorial, written Observation: A co <i>A</i>, not the actual of 	(swap, copy, etc.) to of them ($<$, $>$, rithm we have see I model for sortin re <i>not</i> allowed in Lower Bound for Sorting tower Bound for Sorting $n! = n \times (n - 1)$ comparison-based so) =, etc.) en uses this model. g strings or integer this model? Juit 2 s ions) are there of n $x \times (n-2) \times \cdots \times$ sort is only sensitiv	s or floats or Spring 2014 e elements? 2 × 1.	
 Comparing tw Every sorting algoring the second second	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in Lower Bound for Sorting Lower Bound for Sorting $n! = n \times (n - 1)$ comparison-based scontents.) =, etc.) en uses this model. g strings or integer this model? Juit 2 Juit 2 $(n-2) \times \cdots \times$ sort is only sensitiv same things on	s or floats or Spring 2014 e elements? 2 × 1.	
 Comparing tw Every sorting algoring the second second	(swap, copy, etc.) to of them (<, >, rithm we have see I model for sortin re <i>not</i> allowed in Lower Bound for Sorting Lower Bound for Sorting $n! = n \times (n - 1)$ comparison-based scontents.) =, etc.) en uses this model. g strings or integer this model? Juit 2 Juit 2 $(n-2) \times \cdots \times$ sort is only sensitiv same things on	s or floats or Spring 2014 e elements? 2 × 1.	

Lower Bound	for Sorting		
Logarithms			
Recall some useful facts about	t logarithms:		
• $\log_b b = 1$	t logarithino.		
• $\log_b b = 1$ • $\log_b ac = \log_b a + \log_b c$			
• $\log_b a^c = c \log_b a$			
• $\log_b a = (\log_c a)/(\log_c b)$)		
Now how about a lower bound	d on lg <i>n</i> !?		
SI 335 (USNA)	Unit 2	Spring 2014	19 / 21
Lower Bound	for Sorting		
Lower Bound on Sorting	2		
A correct algorithm must possible input permutation		ions for each of the	
② The choice of actions is a	determined only by	comparisons.	
③ Each comparison has two	o outcomes.		
An algorithm that perform actions.	ms <i>c</i> comparisons	can only take 2 ^c differ	ent
5 The algorithm must perfect to the second secon	orm at least lg <i>n</i> ! o	comparisons.	
Therefore ANY compariso	on-based sort is Ω	$P(n \log n)$	
SI 335 (USNA)	Unit 2	Spring 2014	20 / 21
Conclusions			
Any sorting algorithm that on $\Omega(n \log n)$ steps in the worst of		ns must take at least	

- This means that sorts like MergeSort and HeapSort couldn't be much better they are **asymptotically optimal**.
- What if I claimed to have a O(n) sorting algorithm?
 What would that tell you about my algorithm (or about me)?
- Remember what we learned about summations, recursive algorithm analysis, and logarithms.