

MC215: MATHEMATICAL REASONING
AND DISCRETE STRUCTURES

- **Wednesday, 11/5/08**
 - From last time:
 - **Sorting**
 - Today:
 - **EXAM #2 INFORMATION**
 - **Analysis of Algorithms**
- **READING:**
4.3
- **EXERCISES:**
 - pp. 207-210:
1, 2, 7, 8

Wednesday, 11/5/08, Slide #1
Printed 11/7/2008 10:40 AM

Exam 2: Next Wednesday, 11/12/08
(15% of course grade)

- **Material:**
 - Includes everything through *Friday 11/7/08* class, including all topics covered in class and related readings, all exercises, all hand-in homework
- **Types of Questions:**
 - State definitions, theorems carefully
 - Give examples and counter-examples
 - Solve problems similar to homework and exercises
- **Reference Sheet:**
 - You may bring one reference sheet, 8.5" x 11"
 - OK to write anything you want, on both sides, in any size font.

Wednesday, 11/5/08, Slide #2
Printed 11/7/2008 10:40 AM

Checklist of topics, Chap. 3

- **Chapter 3 (3.1-3.4)**
 - Function, domain, codomain, range, graph of a function, arrow diagrams, mod function, floor and ceiling functions, unary and binary operators, one-to-one, onto, bijections, **inverses, composition**
 - Sequences, increasing and decreasing sequences, subsequences, product notation Π , strings, length, concatenation, substrings; Erdos-Szekeres Theorem (statement only, not proof; see handout from 10/10/08)
 - Relations; graphs of relations; reflexive, symmetric, antisymmetric, transitive relations; partial orders; total orders; equivalence relations; equivalence classes and partitions

Wednesday, 11/5/08, Slide #3
Printed 11/7/2008 10:40 AM

Checklist of topics, Countability

- **Cardinality and Countability** (see handout from 10/13/08)
 - Definitions of cardinality, countable, countably infinite, uncountable; uncountability of \mathbf{R} ; countability of \mathbf{Q} ; definition of $|S| \leq |T|$; Schroeder-Bernstein Theorem (statement only, not proof)

Wednesday, 11/5/08, Slide #4
Printed 11/7/2008 10:40 AM

Checklist of topics, Chap. 4

- **Chapter 4** (material covered by Friday's class, probably 4.1-4.3, but maybe 4.4)
 - Algorithms; properties of algorithms; pseudocode;
 - Linear Search; Text search; Selection Sort; Insertion Sort
 - Analysis of Algorithms, best-case, worst-case, average-case; Big-Oh (order of a functions), Big-theta; growth rates of functions; the order hierarchy
 - Recursive algorithms; n-factorial; Towers of Hanoi, Fibonacci numbers

Wednesday, 11/5/08, Slide #5
Printed 11/7/2008 10:40 AM

Analysis of Selection Sort

start = 1 to n-1	i = start+1 to n	# comps.	#swaps
1	2, 3, ..., n		
2			
...			
n-1			

```

for start = 1 to n-1 {
  small = start
  for i = start+1 to n {
    if (si < ssmall) {
      small = i
    }
  }
  swap(sstart, ssmall)
}
    
```

Wednesday, 11/5/08, Slide #6
Printed 11/7/2008 10:40 AM

Analysis of Insertion Sort

□ Swapping v. assigning

i = 2 to n	j = i-1 down to 1 (at worst)	# comps.	#assgns
2	1		
3	2, 1		
...	...		
n	n-1, ..., 1		

```

for i = 2 to n {
  val = si //save si value
  j = i - 1
  while (j ≥ 1 ∧ val < sj)
    sj+1 = sj
    j = j - 1
  }
  sj+1 = val // insert val
}
    
```

Wednesday, 11/5/08, Slide #7
Printed 11/7/2008 10:40 AM

Analysis of Selection Sort

start = 1 to n-1	i = start+1 to n	# comps.	#swaps
1	2, 3, ..., n	n-1	1
2	3, 4, ..., n	n-2	1
...
n-1	n	1	1
Totals:		$n(n-1)/2$	n-1 swaps

```

for start = 1 to n-1 {
  small = start
  for i = start+1 to n {
    if (si < ssmall) {
      small = i
    }
  }
  swap(sstart, ssmall)
}
    
```

Wednesday, 11/5/08, Slide #8
Printed 11/7/2008 10:40 AM

Analysis of Insertion Sort

□ **Swapping v. assigning?**

start = 2 to n	j = i-1 down to 1 (at worst)	# comps. (at worst)	#assgns (at worst)
2	1	1	1+2=3
3	2, 1	2	2+2=4
...
n	n-1, n-2, ..., 2, 1	n-1	n-1+2=n+1
Totals:		$n(n-1)/2$ at worst	$(n+1)(n+2)/2 - 3$ assigns at worst

```

for i = 2 to n {
  val = si //save si value
  j = i - 1
  while (j ≥ 1 ∧ val < sj)
    sj+1 = sj
    j = j - 1
  }
  sj+1 = val // insert val
}
    
```

Wednesday, 11/5/08, Slide #9
Printed 11/7/2008 10:40 AM

Comparing Selection Sort and Insertion Sort

Selection Sort

- # comparisons:
 $n^2/2 - n/2$
- # assignments
(= 3 x #swaps)
 $3n - 3$

Insertion Sort

- # comparisons:
 $n^2/2 - n/2$ (worst case);
1 (best case);
 $n^2/4 - n/4$ (avg. case)
- # assignments:
 $n^2/2 + 3n/2 - 2$ (worst case);
2 (best case);
 $n^2/4 - 3n/4 - 1$ (avg. case)

Wednesday, 11/5/08, Slide #10
Printed 11/7/2008 10:40 AM

Comparing growth rates of functions

- "Growth (or order) analysis" is a way of comparing algorithm efficiency, but ignoring small differences.
- We *don't care* about "small" differences between functions
 - Ignore constant factors:
E.g. $f(n) = 3n^2 + 5$ and $g(n) = n^2$
 - Ignore behavior for "small" values of n:
E.g. $f(n) = 3n^2 + 5$ and $g(n) = 2^n$ if $n \leq 100$; $g(n) = n^2$ if $n > 100$
- We only count "main" operations
 - E.g. comparisons for search, or comparisons and assignments for sorting
- For order analysis, we *do care* about "large" differences:
 - E.g. $f(n) = 4n^3 + 10$ and $g(n) = 1000 n^2$
 - $g(n)$ larger for small n
 - As n gets large, ratio $f(n)/g(n)$ gets large

Wednesday, 11/5/08, Slide #11
Printed 11/7/2008 10:40 AM

"Big-Oh:" $f(n) = O(g(n))$

- "Big-Oh" notation – a rigorous way to say that the growth of $f \leq$ growth of g
- Definition. $f(n) = O(g(n))$ means there is a constant C such that $|f(n)| \leq C|g(n)|$, for all but finitely many positive integers n .
- "All but finitely many positive integers n " means there is some *threshold*, n_0 , such that for $n \geq n_0$, $|f(n)| \leq C|g(n)|$.

Wednesday, 11/5/08, Slide #12
Printed 11/7/2008 10:40 AM

Examples and a theorem

- Example: Show that $n^2 = O(3n^2 + 5)$
- Example: Show that $3n^2 + 5 = O(n^2)$
- Example: Show that $n = O(3n^2 + 5)$
- Example: Show that $3n^2 + 5 \neq O(n)$

- **Theorem.** If $p(n)$ is any polynomial with degree k , then $n^k = O(p(n))$ and $p(n) = O(n^k)$
 - This theorem says, with regard to order of growth, any two polynomials with the same degree are equivalent

Wednesday, 11/5/08, Slide #13
Printed 11/7/2008 10:40 AM

Example: Selection Sort and Insertion Sort

- **Selection Sort:**
 - # comparisons = $\Theta(n^2)$
 - # assignments = $\Theta(n)$
 - **Overall:** Selection Sort is an algorithm with $\Theta(n^2)$ running time
- **Insertion Sort (average analysis):**
 - # comparisons = $\Theta(n^2)$
 - # assignments = $\Theta(n^2)$
 - **Overall:** Insertion Sort is an algorithm with $\Theta(n^2)$ running time

Wednesday, 11/5/08, Slide #14
Printed 11/7/2008 10:40 AM
