SPRING 2005: **COT 5993** INTRO. TO ALGORITHMS

[HOMEWORK 2; DUE FEB 22 AT START OF CLASS]

Reminder: ADD A SIGNED STATEMENT THAT YOU HAVE ADHERED TO THE COLLABORATION POLICY FOR THIS CLASS AND THAT WHAT YOU ARE PRESENTING IS YOUR OWN WORK.

Problems

9. (**Exercise**) (Exercise 6.2-1, p132)

10. (**Exercise**) (Exercise 6.3-1, p135)

11. (**Exercise**) (Exercise 6.4-1, p136)

12. (**Exercise**) (Exercise 6.5-1, p140)

13. (**Exercise**) (Exercise 6.5-2, p140)

14. (**Exercise**) (Exercise 6.5-2, p140)

15 (**Exercise**) (Exercise 6.5-2, p140)

16. (**Exercise**) (Exercise 7.1-1, p148)

17. (**Exercise**) (Exercise 8.3-1, p173)

18. (**Regular**) For each of the sorting algorithms discussed in class (Insertion Sort, Selection Sort, Bubble Sort, Merge Sort, Quick Sort, Heap Sort, Counting Sort, Bucket Sort, Radix Sort), write down the worst-case and best-case time complexities. Explain your answer in one sentence.

19. (**Regular**) A sorting algorithm is said to be *stable* if the relative order of any equal items in the input list is not changed in the output list. For each of the following sorting algorithms, state which ones are **not** stable by using simple examples: Insertion Sort, Selection Sort, Bubble Sort, Merge Sort, Quick Sort, Heap Sort, Counting Sort, Bucket Sort, Radix Sort. Devise the smallest example you can build. You should consider a sorting algorithm to be stable if the algorithm given to you in the book or the one given to you in class (or a minor variant thereof) is stable.

20. (**Extra Credit**) (Exercise 8.3-4, p173)

21. (**Extra Credit**) (Exercise 8-5, p180)