Homework Assignment #3
Due: Tuesday 11/03/09 in class (code demo/hardcopy due on Friday 10/30/09)
(total: 100 points)

Problem 1 (4 points) Textbook, Exercise 11.2-2 (page 229)
Problem 2 (4 points) Textbook, Exercise 11.2-5 (page 229)
Problem 3 (4 points) Textbook, Exercise 11.4-1 (page 244)
Problem 4 (4 points) Textbook, Exercise 6.1-1 (page 129) and Exercise 6.1-6 (page 130)
Problem 5 (5 points) Textbook, Exercise 6.3-1 (page 135)
Problem 6 (9 points) Textbook, Problem 6-2 (page 143)
Problem 7 (4 points) Textbook, Exercise 12.1-2 (page 256)
Problem 8 (4 points) Textbook, Exercise 12.2-4 (page 260)
Problem 9 (4 points) Textbook, Exercise 12.3-5 (page 264)
Problem 10 (4 points) Textbook, Exercise 7.2-4 (page 153)
Problem 11 (4 points) Read chapter 8 section 8.1 and argue in your own words that the lower bound on any sorting scheme that uses comparisons is $n \log n$.

Programming assignments

Problem 12 (25 points) Hashing
Write Matlab code that for a given long string finds the 10 most frequent triplets (alphabet of 26 English letters is considered) and outputs them together with their number of occurrences. Examples of triplets are ACT, TGT, RRR. Consider the following three types of hashing with open addressing: (i) linear probing, (ii) quadratic probing, and (iii) double hashing. Your solution to the hashing problem should use no more than 30,000 buckets which sets a limit on the load factor to 0.59, however make sure that all slots are considered during probing.

1. (15 points) Try at least three (but preferably more) different values for the number of buckets $m$. For example, a power of 2, a prime number close to the power of 2, and a prime number far from the power of 2. Compare all three hashing schemes in terms of the number of rehashing per visit. Compare these numbers with a theoretical bound and then rank all schemes you tried according to their quality. Also describe how you measured the number of rehashes independently of the machine on which you were running your software.

2. (10 points) Now implement the following brute-force scheme. For each of the 17576 triplets, go over the given long string (incrementing by 1), and using the `strcmp` function, find the 10 most frequent triplets. What can you comment on the speed of both algorithms in terms of
the big-O notation and time in seconds. Are the results using both brute-force and hashing versions identical?

A file containing the long string can be downloaded from the class home page.

**Problem 13** (25 points) Sorting.

1. (7 points) Build a function that implements the heapsort algorithm

   ```matlab
   function [s, c] = heapsort(a)
   
   where \(a\) is an unsorted array, \(s\) is a sorted array, and \(c\) is the number of comparisons performed.
   
2. (7 points) Build a function that implements the quicksort algorithm

   ```matlab
   function [s, c] = quicksort(a)
   
   where \(a\) is an unsorted, \(s\) is a sorted array, and \(c\) is the number of comparisons performed. To simplify the code, you may consider defining a *global* array variable that will be shared within all recursively called quicksort routines.
   
3. (7 points) Build a function that implements the insertion sort algorithm that instead of linear search in the sorted subarray performs binary search

   ```matlab
   function [s, c] = insertion_sort_binary(a)
   
   where \(a\) is an unsorted, \(s\) is a sorted array, and \(c\) is the number of comparisons performed.
   
4. (4 points) Make a wrapper code that generates 10,000 unsorted arrays of various lengths and then sends them to insertion sorts (both linear and binary versions), mergesort (appropriately modify your function from Assignment 1), heapsort and quicksort to sort (all three algorithms should sort the same unsorted arrays for fairness). Comment on the relative speed between all three algorithms, both in terms of numbers of comparisons and time in seconds. Could you estimate the constant factor hidden in the big-O notation? Does this constant factor change on different machines with different processors in expected ways?

Please turn in the code for Problems 12-13 as hardcopies and demonstrate that it works to the teaching assistant. All files should have a .m extension (default Matlab extension).

For homework assignment policies please see the class syllabus located at [http://www.informatics.indiana.edu/predrag/classes/2009falli500/syllabus.htm](http://www.informatics.indiana.edu/predrag/classes/2009falli500/syllabus.htm)

Good luck!