NOTE: check the newsgroup and website regularly for hints or corrections, if any.


Read sections 1.1-1.6, and 2.1-2.2 in the Xeroxed copy of the book (available at Gnomen.)

1. (4 points) Algorithm Performance: Answer the following questions as briefly as possible (in no more than three lines):
   
   a. What does it mean for an algorithm to run in $O(n \log n)$ time? Is such an algorithm considered a polynomial-time algorithm?

   b. Say you are given a randomly ordered list of $2n$ integers, and are told that each number occurs exactly twice in the list. Describe an algorithm that returns only the $n$ unique numbers. What is its time complexity in big-oh notation?

   c. Consider a randomized algorithm which runs for a fixed number of steps and finds the correct solution to an optimization problem with probability greater than $\varepsilon = 0.01$, regardless of the input. How many independent executions of this algorithm suffice to raise the probability of finding the correct solution to at least 0.9, regardless of the input?

2. (6 points) NP-Completeness: A decision problem is one that has a yes or no answer. The decision version of the travelling salesman problem (TSP) from example 1.6 in the text is the following: "for the given graph and associated edge lengths, is there a tour of length at most $C$ (an arbitrary constant)?" The decision version of the Hamiltonian cycle problem (HCP) in example 1.7 is: "does the given graph contain a Hamiltonian cycle?" We will prove that TSP is NP-hard, assuming it is already known that HCP is NP-hard:

   a. Show that if someone presents you with a "certificate" that a particular sequence of vertices on a graph (whose edges are labeled with lengths) constitutes a tour of length at most $C$, you can verify whether this certificate is true in polynomial time. This proves that the decision version of TSP is in the class of computational problems called NP.
b. Now assume you are given a particular instance of HCP, i.e. some graph G. Show how you can assign lengths to the edges of this graph and come up with a bound C such that solving the decision version of TSP on this labeled graph is equivalent to solving the decision version of HCP on G (i.e. either both are answered by yes or both have the solution no). This proves that TSP is at least as hard as HCP. Since HCP is NP-hard, this means that TSP is also NP-hard. Along with part a, this suffices to show that the decision version of TSP is NP-complete.

c. Suggest an algorithm to solve HCP. What is its time complexity in big-oh notation?

3. (10 points) **Randomized Search for Optimization:** We wish to minimize the following simple one-dimensional cost function:

\[ J(s) = (400 - (s - 21)^2) \times \sin(s\pi/6) \]

**Constraints:** \(s\) integer-valued, 0 ≤ s ≤ 32

We will use the following neighborhood function: For each element \(s\) in the space, pick a neighboring solution randomly between max(s-10,0) and min(s+10, 32). Note: for this problem, assume \(s\) is not a neighbor of itself.

a. Write a MATLAB function `cost.m` that implements the cost function, i.e. accepts input \(s\), and returns \(J(s)\). Submit a plot of the given cost function with respect to \(s\).

b. Write a MATLAB function `neighbor.m` that implements the neighborhood function, taking a single input \(s\), and outputting a neighboring value \(s_{new}\).

c. Write a MATLAB function `RW.m` which implements the RandomWalk algorithm described in exercise 1.9 of the book. The function should have two inputs: a starting point \(s_{initial}\), and a maximum number of iterations to run \(maxiter\). The output of the function is a matrix `solution` with each row corresponding to the results for each iteration and columns corresponding to the following for each row.

- \(i\) – the iteration number (same as the row number)
- \(s_{current}\) – the value of \(s\) at iteration \(i\)
- \(s_{best}\) – the value of \(s\) that had the lowest cost in the history of the search (best solution to date).
- \(J_{current}\) – the current cost, i.e. \(J(s_{current})\)
- \(J_{best}\) - the lowest cost in the history of the search

This function should call `cost.m` and `neighbor.m` where needed. The header of this function should thus read:

    function solution = RW(sinital, maxiterations)
d. Write a MATLAB function RS.m which implements the RandomSampling algorithm described in exercise 1.10 of the book. Follow the same conventions as in part c.

e. Call the RW and RS functions 30 times (runs) with $s_{initial} = 10$, $max iterations = 10$. Submit plots of $s_{current}$, $s_{best}$ vs. iterations for one run on one figure, and the average of $J_{current}$, average of $J_{best}$ vs. iterations on another figure. Compute and report the average and standard deviation of $J_{best}$ after 10 iterations for each algorithm. How often did each algorithm find the right solution? Comment on these plots and results. Which algorithm performs better?

SUBMIT ALL CODE AS WELL