B1 Algorithms (25 pts)

Complete a, b and c.

a) Complete all four portions of this question.

For each of the following, outline an algorithm that finds the numbers within the given amount of time. For example, if S = {6, 13, 19, 3, 8} then 19 - 3 maximizes the difference while 8 - 6 minimizes the difference.

In the algorithm outlines that you provide, assume only operations provided in a programming language like C or Pascal (but no library functions). The outline may be in English prose or pseudo-code.

- [2 points] Let S be a sorted array of n integers. Give an algorithm that finds the pair x,y in S that maximizes $|x - y|$. Your algorithm must run in O(1) worst case time.
- [2 points] Let S be an unsorted array of n integers. Give an algorithm that finds the pair x,y in S that maximizes $|x - y|$. Your algorithm must run in O(n) worst case time.
- [2 points] Let S be a sorted array of n integers. Give an algorithm that finds the pair x,y in S that minimizes $|x - y|$, for $x \neq y$. Your algorithm must run in O(n) worst case time.
- [4 points] Let S be an unsorted array of n integers. Give an algorithm that finds the pair x,y in S that minimizes $|x - y|$. Your algorithm must run in O(n lg n) worst case time.

b) Complete both portions of this question.

- [5 points] Give pseudo-code for Prim's minimum spanning tree algorithm.
- [5 points] Explain how to implement that algorithm in time O(e + n log n), where e is the number of edges in the graph and n is the number of vertices.

  Note: If you use another algorithm as a subroutine in that implementation you do not have to explain how the subroutine is implemented. You do have to explain how it is used. For instance, if your implementation uses AVL trees, then you need to say "this is where we do an AVL insert", but you would not have to explain either what an AVL tree is, nor how to do an insert in an AVL tree.

c) [5 points] In the analysis of Union/Find using only the Ranking rule, carefully explain why the cost of a Find in this situation does not exceed O(log n).

B2 Algorithms (25 pts)

Complete a and b.

a) [15 points] Let n = 2k and consider two polynomials:

$P = a_{2k-1}x^{2k-1} + \ldots + a_k x^k + a_{k-1}x^{k-1} + \ldots + a_1 x + a_0$
$Q = b_{2k-1}x^{2k-1} + \ldots + b_k x^k + b_{k-1}x^{k-1} + \ldots + b_1 x + b_0$ where a_i, b_i are integers, $0 \le i \le 2k$.

Devise a "fast" algorithm (i.e., one with a worst case computing time that is $o(n^2)$ using arithmetic complexity) for polynomial multiplication of two polynomials of degree n-1. Your method should *not* be based on the FFT. Your algorithm should be in the form of pseudo assuming existing functions for all linear operations (however clearly specify what functions you are assuming).

Hint: Let $P_1 = a_{2k-1}x^{k-1} + \ldots + a_k$, and $P_0 = a_{k-1}x^{k-1} + \ldots + a_0$
Let Q_1 and Q_0 be defined analogously

Then $P*Q = (P_1 x^k + P_0)(Q_1 x^k + Q_0)$
$= P_1Q_1 x^{2k} + (P_1Q_0 + P_0Q_1) x^k + P_0Q_0$

Note that $P_1Q_0 + P_0Q_1 = (P_1 + P_0)(Q_1 + Q_0) - P_1Q_1 - P_0Q_0$.

Use this along with divide and conquer to devise the fast algorithm.

b) [10 points] Give a recurrence relation for the worst case computing time for your algorithm. Now solve the recurrence relation to give a tight upper bound for the worst case computing time.

B3 Algorithms (25 pts)

Complete a, b and c.

a) [10 points] Give an example of dynamic programming not related to shortest paths or transitive closure. Provide enough details in your explanation that the reader can understand that the example actually works.

b) [5 points] Define each of the following.

- P
- NP
- polynomially reducible
- NP-complete

c) [10 points] Prove that the following DOUBLE_CLIQUE problem is NP-complete, using a reduction from CLIQUE.

INSTANCE: An undirected graph G and a positive integer k.
QUESTION: Does G contain two disjoint k-cliques?

Note: Disjoint cliques means that the two cliques do not have any common vertices.

B4 Algorithms (25 pts)

Complete a and b.

a) Complete all three portions of this question.

- [2 points] Describe the prefix sums problem.
- [5 points] Give an O(log n) algorithm for computing prefix sums on an EREW PRAM.
- [8 points] Give an O(log n) algorithm on an EREW PRAM for solving the following problem:

  Given a sequence of integers $x_1, x_2, \ldots, x_n$, we will say that x_i, $i < n$, is ***locally big*** if $x_i > x_{i+1}$. Your algorithm should output the number of locally big values in the sequence.

  Example: The output for 12 8 3 2 6 5 is 4 since 12, 8, 3, and 6 are all locally big.

b) For each of the following items *state* what data structure and/or algorithm could be used to solve the problem using the least worst case time complexity. If there are several possibilities, just give one. Note that you should just name the data structure/algorithm. You should NOT explain how it is used. You should also *state* the worst case time complexity. Again, you should NOT explain how the time complexity is computed.

- [2.5 points] finding the connected components of an undirected graph
- [2.5 points] processing a sequence of n min, insert and delete_min operations
- [2.5 points] finding single source shortest paths in directed graphs
- [2.5 points] processing a sequence of n member, insert and delete operations