Computer Science B63 | July 20, 2008
Scarborough Campus | University of Toronto

Homework Assignment #3/4
Due: July 31, 2008, by 1:10 pm

1. *Please complete and attach (with a staple) an assignment cover page to the front of your assignment.*
2. *If you do not know the answer to a question, and you write "I do not know the answer to this question", you will receive 20% of the marks of that question. If you just leave a question blank with no such statement, you get 0 marks for that question.*
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

Question 1. (20 marks) A biological laboratory has a project to determine the complete DNA sequence of chromosome 3 of the ornithorhynchus anatinus. The project will take several years to complete and will generate huge amounts of experimental data. These data are generated by chopping up very long strands of DNA into millions of shorter strands (called *clones*) and by performing experiments to determine which pairs of clones overlap. A very important part of the process is analyzing the data as it is generated to discover sets of clones that cover contiguous regions of the chromosome. Such sets are called *contigs*. The set of contigs is not fixed, but changes as more and more data are generated. Eventually, when enough clones are generated to cover the entire chromosome, and when enough overlaps are detected, there will be just one contig.

Each clone is assigned a unique integer id number, but these ids may not be contiguous and may have a large range. The laboratory generates a stream of data items of the following two forms:

NEW(I): A new clone has been analyzed and given an integer id I.

OVERLAP(A, B): Clones A and B have been discovered to overlap. If either A or B does not exist, then this operation has no effect.

Two clones, I and I', belong to the same contig if and only if there is a sequence of clones $I_1, I_2, ..., I_n$ such that clone I overlaps clone I_1, clone I' overlaps clone I_n, and clone I_{i-1} overlaps clone I_i for all $1 < i \leq n$.

Each time a data item is generated, the set of contigs needs to be updated.

a. (12 marks) Describe an efficient data structure for representing the contigs, and give algorithms for processing NEW and OVERLAP data items. If you base your solution on data structures we studied in class, concentrate on how you use or modify them rather than describing textbook details.

Note: Part of your grade will depend on the efficiency of your data structure, i.e., more efficient data structures will receive more marks and less efficient data structures will receive fewer.

Hint: Be sure to also describe any auxilary data structures you may require, such as for determining whether a particular id exists or where it is located within your data structure.

b. (8 marks) Determine the worst-case cost of processing a sequence of n NEW and m OVERLAP data items. What is the amortized cost per data item? Justify your answer.

Question 2. (20 marks) Consider the following data structure for representing a *set* of elements and implementing SEARCH and INSERT operations:

- The elements in the set are stored in a singly linked list of sorted arrays, where the number of elements in each array is a power of 2 and the sizes of all the arrays in the list are different. Each element in the set occurs in exactly one array. The arrays in the linked list are kept in order of increasing size. For example, the linked list might contain arrays of size $2^0, 2^3, 2^4$ and 2^7, in this order.

- To perform a SEARCH, iteratively perform binary search separately on each array in the list until either the desired element is found or all arrays have been considered.
  For the above example, we'd first search the array with 2^0 elements, then the array with 2^3 elements, and so on.

- To INSERT a new element x into the set (given the precondition that x is not already in the set),

  create a new array of size 1 containing x
  insert this new array at the beginning of the linked list
  while the linked list contains 2 arrays of the same size **do**
      merge the 2 arrays into one array of twice the size

a. (5 marks) What is the worst-case time, to within a constant factor, for performing SEARCH when the set has size n? Justify your answer.

b. (5 marks) What is the worst-case time, to within a constant factor, for performing INSERT when the set has size n? Justify your answer.

c. (10 marks) Use the accounting method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(\log n)$.

Question 3. (5 marks) An undirected multigraph is a generalization of an undirected graph in that the edges form a multiset: for any pair of vertices u and v, there may be zero, one, or more edges joining u and v. The number of edges between u and v is the *multiplicity* of (u, v).

a. (3 marks) Describe how to represent a multigraph using each of these approaches: an adjacency lists representation, and an adjacency matrix representation. Be precise. What is the space usage for each approach, in terms of the number of vertices n and the number of edges m?

b. (2 marks) Suppose each edge in an undirected multigraph has a weight associated with it (edges between the same vertices may each have *different* weights). Describe an efficient way to represent such a weighted multigraph using adjacency lists. What is the complexity of answering the question "is there an edge between u and v with weight x?"

Question 4. (20 marks) A bipartite graph $G = (V, E)$ is a graph where V can be partitioned into two sets V_1 and V_2 (one of which may be empty) such that there are no edges between any two nodes in V_1 or between any two nodes in V_2 (that is, all edges of G have one endpoint in each set). There are many fast algorithms on bipartite graphs, so it is important to be able to recognize whether a given graph is bipartite.

a. (10 marks) Prove that a graph is bipartite if and only if it does not contain an odd cycle. Recall that an *odd cycle* is a simple cycle that contains an odd number of vertices.

b. (10 marks) Show how to modify DFS so that, given any graph G, it proves in $\Theta(n+m)$ time whether the graph is bipartite or not. If the graph *is* bipartite, your algorithm should return two sets V_1 and V_2 as described above, but if the graph *is not* bipartite, your algorithm should return an odd cycle in G.

Question 5. (marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.