

CSC C78F 2000

Assignment 4

due: Wednesday, November 22, 2000

1. Consider the ADT MINMAXPQ that consists of a set of elements from an ordered universe and supports the following operations:

INSERT(S, x): $S \leftarrow S \cup \{x\}$.

DELETEMIN(S): returns the minimum element in S and deletes it from S .

DELETEMAX(S): returns the maximum element in S and deletes it from S .

An SMM HEAP is a data structure that consists of an array A of size m . This is viewed as a full binary tree, as in a heap, except that the root node is empty. Instead, $A[1]$ contains the number of elements in S .

Every node x in the tree satisfies the following two properties:

- the minimum element among all the descendants of x is located at the left child of x , if x has a descendant, and
- the maximum element among all the descendants of x is located at the right child of x , if x has at least 2 descendants.

- (a) Give all three SMM HEAPS for the set of elements 1,2,3,4,5. Draw the corresponding full binary trees.

- (b) Prove that an array A is an SMM HEAP if and only if

- $0 \leq A[1] < \text{length}(A)$,
- for every location $1 \leq i \leq \lfloor A[1]/2 \rfloor$ we have $A[\text{left}(i)] \leq A[\text{right}(i)]$, and
- for every location $3 < i \leq A[1] + 1$ we have $A[i] \geq A[\text{left}(\text{parent}(\text{parent}(i)))]$ and $A[i] \leq A[\text{right}(\text{parent}(\text{parent}(i)))]$.

- (c) Describe how to perform INSERT, DELETEMIN, and DELETEMAX on an SMM HEAP in $O(\log n)$ time, where n is the size of S . Justify that your algorithms run in the required time and that, after any of your algorithms is performed, the resulting array is an SMM HEAP.

2. A *self-stabilizing data structure* is a data structure that corrects itself automatically after its data has been corrupted. Data corruption can occur, for example, as a result of a power outage and can cause the contents of any number of array elements and the values of any number of variables to change arbitrarily. After data corruption has occurred, operations on a self-stabilizing data structure may give incorrect results, but, eventually, if no more data corruptions occur, only correct results will be produced. The *stabilization time* of a self-stabilizing data structure is the number of operations that can give incorrect results after the last corruption has occurred. Note that the stabilization time may depend on the size of the data structure immediately after the last data corruption occurred.

Operations do not know when the last data corruption occurred, so a trivial solution such as reinitializing the data structure (to be empty) is not allowed.

Describe a self-stabilizing variant of the HEAP data structure such that the worst case time complexities of the operations DELETEMAX and INSERT change by no more than a constant factor and the stabilization time is $O(m)$, where m is the size of your set immediately after the last data corruption occurred.

Justify that your self-stabilizing heap has the required stabilization time and worst case DELETEMAX and INSERT times. Hint: show that the data in your self-stabilizing heap will eventually satisfy the priority conditions.

3. Many European countries have currencies that fluctuate together on the world currency markets.

Consider a disjoint sets data structure that represents the following currencies:

BEF Belgian Franc
DEM German Deutsche Mark
DKK Danish Kroner
ESP Spanish Pesetas
FIM Finnish Markka
FRF French Franc
ISK Icelandic Krona
ITL Italian Lira
NLG Dutch Guilder
NOK Norwegian Kroner
SEK Swedish Krona

Suppose the following imaginary sequence of operations are performed:

UNION(DEM,FRF)
UNION(BEF,NLG)
UNION(FRF,BEF)
FIND(NLG)
UNION(ITL,ESP)
UNION(DKK,SEK)
UNION(DKK,NOK)
UNION(NOK,ISK)
UNION(FIM,SEK)
UNION(NOK,DEM)
FIND(ISK)
UNION(FRF,SEK)
UNION(ESP,NLG)

Draw each of the following disjoint set data structures for this collection of eleven currencies after this sequence of operations, assuming that, initially, all sets are singleton sets.

- (a) linked list (as described on page 443 of CLR)
 - (b) linked list with weighted union
 - (c) disjoint set forest
 - (d) disjoint set forest with union by weight
 - (e) disjoint set forest with union by rank
 - (f) disjoint set forest with path compression
 - (g) disjoint set forest with union by weight and path compression
 - (h) disjoint set forest with union by rank and path compression
4. A large biological laboratory has a project to determine the complete DNA sequence of Human chromosome 7. The project will take several years to complete and will generate huge amounts of experimental data. This data is generated by chopping up very long strands of Human 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 is generated. Eventually, when enough clones are generated to cover the entire chromosome, and when enough overlaps are detected, there will be just one contig.
- The laboratory assigns a unique integer id number to each clone and it generates a stream of data items of the following two forms:

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

OVERLAP(A, B): Clones A and B 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, \dots, 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) Describe an efficient data structure for representing the contigs and performing NEW and OVERLAP. Determine the worst case cost of processing a sequence of m NEW and n OVERLAP data items. What is the amortized cost per data item? Justify your answer.
- (b) Implement your data structure. To test the correctness of your code, you need to implement an extra operation:

PRINT(I): prints all the clones in the contig I belongs to. This command has no effect if I is not a recognized clone id.

Properly document your code and test cases.

Details about input and output format and submitting your program can be found on the course website.

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.