125 marks **CS 606 Endsemester** **14:00-17:00, 19/11/14**

There is plenty of time; answers must be written exceedingly neatly, using full sentences.

Problem 1: A priority queue, as you know, supports two operations. $I(x)$ causes an integer x to be inserted into the queue. $R()$ causes the smallest integer in the queue to be removed and returned.

(a)[10 marks] Build a priority queue using an n processor linear array capable of storing upto n elements. It should return the response to $R()$ in $O(1)$ time.

The semisystolic version: linear array should hold the elements in non decreasing order, each processor i holding element y_i. The zero delay edges go to away from the host. 3 marks

When a request $I(x)$ arrives, it is sent right along the zero delay edges. Processor i sends right $I(\max(x, y_i))$ and sets $y_i = \min(x, y_i)$. 3 marks

When $R()$ is received, it is sent to all using the zero delay edges. This causes each processor to send its y_i towards the host in the next step using zero delay edges. 2 marks

It must be slowed down by a factor of two... 2 marks

(b)[10 marks] Implement the same functionality as above, but this time use only a $\log n$ processor linear array. Each processor may have a large memory. Hint: Mimic the uniprocessor implementation of a priority queue. This is a hard problem, attempt only after you have done everything else.

In a sequential heap, we have 2^i nodes at level i of a tree, with the last level possibly having fewer. We keep level i on processor i, of the network as above. Each processor also knows the number n of elements currently in the heap.

Consider an $I(x)$ request entering a level i. Let y denote the value at the parent of element n of the heap in level i. We transmit forward $I(\max(x, y))$, and store in the parent the value $\min(x, y)$. As you can see, this will cause a new key to be added in the nth position of the heap.

Here is what happens on an $R()$ request. A processor i receives the request as $R(m)$ where m refers to the node which will become empty in level $i - 1$. Let x, y be the values at the children of m in level i, and z the value at the parent of node n. Now we send the minimum of these to processor $i - 1$. If the value from node r was sent, then we send $R(r)$ to processor $i + 1$, unless processor i is the last active processor. If processor i is the last active processor, then we move the value at node n into r, unless $r = n$.

Problem 2: A mesh of trees network $MOT(n)$ is defined as follows. First we have 2^{2n} ("leaf") processors arranged in 2^n rows and 2^n columns. Then, on top of each column we construct a tree, with the 2^n processors in the column constituting the leaves of the tree. Likewise we construct a tree on each row.

(a) [10 marks] What is the diameter of the $MOT(n)$?

$4n$, 5 marks for upper and 5 for lower bounds.

(b) [10 marks] Show that a $2^n \times 2^n$ matrix A can be multiplied by a $2^n \times 1$ vector x in time $O(n)$ on $MOT(n)$. State clearly where the data is read and the output $y = Ax$ generated.

Read a_{ij} in leaf ij. Read x_j in root of tree on row j. 5 marks

Send x_j to all leaves of the row tree. Thus leaf ij will get x_j. Leaf ij will calculate $a_{ij}x_j$. Now use the column trees to add up the values at the leaves. Thus the root of column i tree will get $y_i = (Ax)_i$. 5 marks

(c) [10 marks] Show that a $MOT(n)$ can be embedded in a hypercube H_{2n} having $2n$ dimensions with unit dilation such that each node of the hypercube receives at most 1 node from any level of any of the trees. Your answer must contain a line "The ith node in level k of tree ... is placed on ...". Argue that this implies that the matrix multiplication of part (b) can be done on H_{2n} in time $O(n)$.

Number a column/row of the leaves 0 through $2^n - 1$. Place leaf ij at processor $i||j$ of the hypercube. 3 marks

For a tree, place each parent in the same node as its left child. Thus jth node in level k of row tree i will be placed in processor $i||j2^k$. Its children will be at $i||2j2^{k-1} = i||j2^k$ and $i||(2j+1)2^{k-1} = i||j2^k + 2^{k-1}$. Thus a parent will be adjacent to its children. 5 marks.

Since the matrix multiplication happens level by level, the time is the same as in $MOT(n)$, i.e. $O(n)$. 2 marks

(d) [10 marks] Embed a complete directed graph on 2^{2n} vertices in $MOT(n)$ such that the vertices of the complete graph are placed with load 1 on the leaf processors of $MOT(n)$. Minimize the congestion. Let E denote smallest set of edges that must be removed so that $MOT(n)$ separates into two subgraphs, each containing half, i.e. 2^{2n-1} leaf processors. It does not matter how the tree nodes get divided into the subgraphs. Show that $|E| = 2^n$. (use the embedding congestion)

Cutting in the middle gives the upper bound.

Embed a complete directed graph. Path from (i, j) to (l, k) goes through column tree and row tree. Row tree i will have congestion from half the 2^n elements of the row, going to all l in the other half, for arbitrary k. Thus the congestion is $2^{n-1} \times 2^{n-1} \times 2^n = 2^{3n-2}$. Thus the bisection is $2^{4n}/4C = 2^{4n}/4 \cdot 2^{3n-2} = 2^n$.

Problem 3: Suppose we are given as usual a list ranking/suffix problem on n elements, i.e. we are given arrays $NEXT[1..n]$ and $VALUE[1..n]$ stored in the memory of a p processor PRAM (whatever type you like), where $p = n/\log^2 n$.

Suppose each processor i independently randomly selects an element a_i of the list, and marks it. Let a_j be the next marked element if any encountered as the list is traversed from a_i. The elements between a_i (inclusive) and a_j (exclusive) are said to constitute the *load* of processor i.

(a)[5 marks] What is the probability that a given processor i has load at least L?

Other processors must select only from $n - L$ elements. So the probability is $(1 - L/n)^{p-1} \leq (1 - L/n)^{p/2}$.

(b)[5 marks] Show that the load of any processor is $O(\log^3 n)$ with high probability.

Substituting we get $(1 - k\log^3 n/n)^{p/2} \leq \exp(-k \log n/2) = n^{-k/2}$.

(c)[10 marks] Suppose that you have a (magical!) procedure that picks a_i such that all processors have load $\log^2 n$. Show that in this case you will have an easy algorithm that does list ranking in time $O(\log^2 n)$.

Each processor i eliminates all the nodes in its load except for a_i. This requires $\log^2 n$ time. 5 marks

Then we run Wyllies algorithm on the a_i, which requires $O(\log p) = O(\log n)$ time. 5 marks.

Problem 4: Consider a butterfly network with $N = 2^n$ inputs. Some of the inputs hold a single packet. Let r_i denote the number of inputs among $0 \dots i$ that hold packets.

(a)[10 marks] Show that it is possible to establish vertex disjoint paths from every input i holding a packet to output $r_i - 1$.

Suppose the packet paths starting at inputs i, j meet in node (x, y). Subsequently, the y lsbs of the vertices on the path will not change, i.e. they will equal the y lsbs of x. The paths finally reach vertices $r_i - 1$ and $r_j - 1$. Thus $|(r_i - 1) - (r_j - 1)| \geq 2^y$, i.e. $|r_i - r_j| \geq 2^y$.

On the other hand, when the paths meet at (x, y), only the least significant y bits can have changed, i.e. the remaining bits must be identical in i, j. Thus $|i - j| < 2^y$.

But each node can hold at most one packet. Thus $|r_i - r_j| \leq |i - j|$. Thus we have a contradiction.

(b)[5 marks] We wish to establish paths between node i and $r_i - 1$ on a N node hypercube. Based on the relationships you know between the hypercube and the butterfly, state how you can get paths of low congestion. State what congestion you get.

If we collapse each row of the butterfly we get the hypercube. The cross edges become edges of the hypercube, except that we have two butterfly edges mapping onto a single hypercube edge. But given that the Butterfly had congestion 1, in the hypercube also we have congestion 1 in each direction at most.

(c)[10 marks] Same as part (b), but for the N node deBruijn graph.

The Butterfly is isomorphic to the Omega, so vertex disjoint paths in the former remain so in the latter. 5 marks.

The rows of the latter when collapsed give us the deBruijn. Thus we get at most $\log N$ congestion in each edge. 5 marks.

Problem 5:(a)[5 marks] Give a lower bound on the area required to layout a 2^n node hypercube.

The area $= \Omega(BW^2) = \Omega(2^{2n-2}) = O(2^{2n})$.

(b)[15 marks] Give a layout for a 2^n node hypercube. You may assume that each node is layed out in an $n \times n$ grid. You get full credit only for a layout matching the lower bound (to within constants), but will also get some credit if you give any layout and evaluate its area requirement. Assume n is even.

Do a divide and conquer layout. Each time remove two dimensions. We will have to add $O(2^n)$ wires. Thus the recurrence will be $S(n) = 2^n + 2S(n-2)$. 7 marks

The base case is $S(1) = n$. 3 marks

Thus we get $S(n) = 2^n + 2^{n-1} + 4S(n-4) = 2^n + 2^{n-1} + \dots + 1 + 2^{n/2}S(1) = O(2^n)$. 5 marks.