Parallel Prefix

Note: These notes derived from S. Akl, *Parallel Computation: Models and Methods*, chapt. 4.

Parallel Prefix (review): Given $X = \{x_1, x_2, \ldots, x_n\}$ and a binary associative operator $\otimes$, which is closed with respect to X, compute $x_1, x_1 \otimes x_2, x_1 \otimes x_2 \otimes x_3, \ldots$ where the k^{th} item $x_1 \otimes x_2 \otimes \ldots \otimes x_k$ is called the k^{th} prefix.

- **Binary**: $x_i \otimes x_j$
- **Closed**: $x_i \otimes x_j$ is in $X = \{x_1, x_2, \ldots, x_n\}$
- **Associative**: $(x_i \otimes x_j) \otimes x_k = x_i \otimes (x_j \otimes x_k)$
- Note: $\otimes$ is not necessarily **commutative**, i.e., there is no requirement that $x_i \otimes x_j = x_j \otimes x_i$

Assume that $\otimes$ is a **constant time** computable function

Simple Examples of $\otimes$ (discuss):
1. Addition, multiplication, max, or min (on integers or reals)
2. Concatenation (on symbols) - discuss what is wrong with this
3. Logical operations (and, or, XOR) on Boolean values

Lower Bound number of operations required to perform a complete parallel prefix operation is $\Omega(n)$ since the n^{th} prefix involves operating on n values.

Parallel Prefix Operation is sometimes referred to as *scan* or *sweep*.

RAM Algorithm: Demonstrate straightforward sequential algorithm. Discuss time and space.

Parallel Algorithms (Assumptions): Assume that the data in initially stored in contiguous memory locations - either in contiguous cells in the shared memory of a PRAM or in contiguous processors on a distributed memory machine. We will deal with linked lists at a later time. Note that this situation is analogous to the one just discussed for the **RAM**.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Parallel Prefix on the PRAM

1. Algorithm 1 (naive)
   a) Assume n processors, $P_0, P_1, \ldots, P_{n-1}$, where processor P_i is associated with x_i.
   b) Demonstrate recursive-halving type algorithm - combine pairs, then pairs of pairs, then ....
   c) for $j = 0$ to (log n) - 1 do
      for $i = 2^j$ to n-1 do in parallel
         $x_i \leftarrow x_{i-2^j} \otimes x_j$
      end for
      end for
   d) Running time: O(log n)
   e) Cost: $c(n) = p(n) \times t(n)$
      $= n \times O(\log n)$
      $= O(n \log n)$
      which is **not** optimal
2. Algorithm 2 (cost-optimal)
   a) Options for optimality:
      i) reduce running time to O(1), while keeping the number of processors at n, **or**
      ii) reduce the number of processors by a factor of log n, while keeping the running time at O(log n).
   b) Discuss an optimal 2-phase algorithm
   c) Cost optimal: Discuss the running time and number of processors

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Maximum Sum Subsequence

Defn.: Given a sequence $X =< x_0 x_1 \ldots x_{n-1} >$, find indices *u* and *v*, $u \le v$, such that the subsequence $< x_u x_{u+1} \ldots x_v >$ has the largest possible sum $x_u + x_{u+1} + \ldots + x_v$ among all such subsequences in *X*.

Note: if all values of *X* are positive, then trivial: the entire sequence is the solution.

RAM Solution:

1. Initialize the maximum to x_0
2. Solve the problem for $< x_0 x_1 \ldots x_{i-1} >$.
3. Extend the solution to include x_i. The maximum sum in $< x_0 x_1 \ldots x_i >$ is the maximum of the following:
   a) The sum of a maximum sum subsequence in $< x_0 x_1 \ldots x_{i-1} >$, which is referred to as the maximum sum seen so far (called *Maxseen*)
   b) The sum of a subsequence ending with x_i (called *Maxhere*)

```
Maxseen ← x_0          {Initialize Maxseen}
u ← 0
v ← 0

Maxhere ← x_0          {Initialize Maxhere}
q ← 0

for i=1 to n-1 do      {Traverse list and do updates}
    if Maxhere ≥ 0
    then Maxhere ← Maxhere + x_i
    else Maxhere ← x_i
         q ← i
    end if

    if   Maxhere > Maxseen
    then Maxseen ← Maxhere
         u ← q
         v ← i
    end if
end for
```

Discuss time and space of RAM algorithm.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

PRAM Algorithm:

1. Compute the prefix sums $\{s_0, s_1, \ldots, s_{n-1}\}$ of $X = \{x_0, x_1, \ldots, x_{n-1}\}$.
2. Compute the **postfix** maximum so that for each *i*, the maximum $s_j, j \geq i$, is determined, along with the value *j*. So, a postfix-max is computed that drags the index along with it. Let m_i denote the value of the postfix-max at positions *i*, and let a_i be the associated index.
3. For each *i*, compute $b_i = m_i - s_i + x_i$. Note that this is the maximum prefix value of anything to the right minus the prefix sum plus the current value (which was in both terms so needs to be added back).
   a) Show example with some data of positive and negative values chosen at random.
4. The maximum of these b_i represents the solution, where *u* is the index of the position where the maximum of the *b's* are found and $v = a_u$.

Discuss time and space and number of processors. Discuss an efficient algorithm vs. an optimal algorithm. Also, discuss the implementation of this algorithm on network models (MCC).

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Array Packing

Defn.: Given an array of items, call it *X*, where some of the items are labeled, rearrange the array so that all of the labeled items come before all of the unlabeled items. That is, *pack* the labeled items at the front of the array. This problem is equivalent to sorting a set of 0s and 1s into nonincreasing order.

RAM Solution: Sort the items:

1. Discuss using a $\Theta(n \log n)$ time sort
2. Discuss using a $\Theta(n)$ time modified counting sort (remember, need to rearrange the items, not just the labels). Discuss the need for an auxiliary data structure.
3. Discuss using the *partition* routine associated with quicksort (array version with pointers starting at both ends and each looking for an item that doesn't belong and then swapping). Discuss the running time of this routine.

PRAM Solution:

1. Let each labeled item use a "1" for its value
2. Let each unlabeled item use a "0" for its value
3. Perform a parallel prefix sum on the values. Also, perform a parallel postfix to broadcast the value of the maximum of these values (i.e., the total number of labeled entries).
4. Each labeled entry now knows its position in the array
5. Reverse the 1's and 0's and perform prefix-sum so that all unlabeled items now know their position w.r.t. the other unlabeled items. Add the max that was broadcast at the end of step 3 to this value to determine the overall position.
6. Move the labeled values to the positions determined in step 3. Move the unlabeled items to the positions determined in step 5.

Discuss time and space on a PRAM. Discuss problems/possibilities on network models. Discuss traditional solution on a network model (sorting). Discuss why that isn't typically done on the PRAM even though sorting can now be done in logarithmic time.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Discuss problem of selection (determining the k^{th} item) from an array on a RAM (modification of quicksort/partition) to note that it uses packing in some sense.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Interval Broadcasting

Defn.: Given an **array** X of n elements, with a subset of the elements marked as "leaders," broadcast the values associated with the leaders to the subsequent elements, up to, but not including, the next leader.

Show example of an array layout with a subset of the elements *marked*, and then show the result of this operation.

Modification to normal parallel prefix:

1. Each leader in processor P_i, which initially contains data element d_i, creates the record (i, d_i).
2. Each nonleader in a processor P_i, which initially contains data element d_i, creates the record $(-1, dummy)$.
3. The prefix operator $\otimes$ is defined as:

$$(i,a) \otimes (j,b) \rightarrow (i,a) \text{ if j < i}$$

$$(i,a) \otimes (j,b) \rightarrow (j,b) \text{ otherwise}$$

Discussion:

1. Discuss the operator and its properties.
2. Time and space and *cost* on RAM, PRAM, Mesh.

Examples:

1. Do **quicksort** with prefix operations (RAM/PRAM) - discuss running times
2. Discuss modification of interval broadcasting to *interval prefix computation*, where the operator is applied to all of the elements within each interval. This requires a minor modification to the operation defined above. Give examples in terms of some database applications.

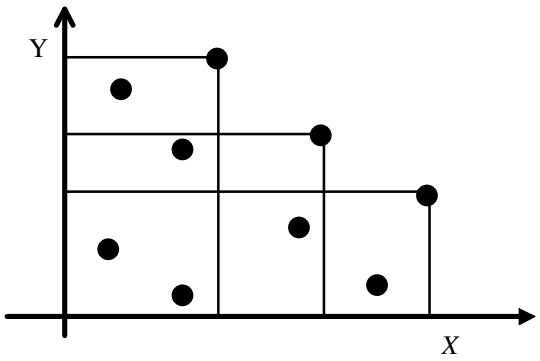
Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

(Simple) Point Domination Query

Defn.: A point $q_1 = (x_1, y_1)$ is said to dominate a point $q_2 = (x_2, y_2)$ *if and only if* $x_1 > x_2$ and $y_1 > y_2$. Given a set $Q = \{q_1, q_2, \ldots, q_n\}$ of planar points, find all points *q* that are **not** dominated by any *Q*.

Discuss the relevance of the problem.

Brief discussion of the field of **computational geometry**: that branch of computer science concerned with designing efficient algorithms for solving geometric problems - i.e., problems involving points, lines, polygons, and other geometric figures.



Input:
1. A set of *n* points ordered by *x*-coordinate.
2. No two points have the same *x* or *y*-coordinate.

Algorithm:
1. Perform parallel postfix with operator *max-y*.
2. All points q_i that have a greater *y*-coordinate than the postfix value $q_{i+1} \otimes \ldots \otimes q_n$, are the set of points that are not dominated by any others.

Discuss time, space, and cost for RAM, PRAM, and Mesh.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Computing Overlapping Line Segments

Defn.: Assume that a set of n uniquely labeled line segments, all of which lie along the same horizontal line, are given. The representation is that each endpoint is represented by a record containing the x-coordinate of an endpoint, the label of the line segment, and a flag indicating whether the point is the Left or Right endpoint of the line segment. Further, assume that the $2n$ records are given ordered with respect to the x-coordinate of the records.

1. Give an efficient algorithm to determine whether or not the line segments completely cover the horizontal line in a given range $[A,B]$.
   a) Discuss RAM, PRAM, and Mesh.
   b) Discuss optimality in terms of time, space, and cost.
2. Give an efficient algorithm which will identify a point of the line that is covered by the most line segments under each of the following assumptions. (Note that such a point must necessarily be one of the given endpoints.)
   a) Discuss RAM, PRAM, and Mesh.
   b) Discuss optimality in terms of time, space, and cost.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Computing Cliques of Circular Arcs

Defn.: Given a sequence of arcs $R = < r_1 r_2 \ldots r_n >$, find a point on the circle that has maximum overlap. That is, find a point *q* that has a maximum number of arcs that overlap it - notice that while *q* is not necessarily unique the maximum number of arcs that overlap a point is.

Look at the problem in its simplest form.

Input:

1. No arc is contained in any other.
2. No two arcs share an endpoint.
3. The endpoints of the arcs are given completely sorted is clockwise order.
   a) The tail point of an arc only appears following the head of its arc and the points are given so they go around twice.
   b) Note that each arc is represented by two points (records) in the sorted list.
   c) Draw example.

Algorithm: Associate with each head the value of +1 and with each tail the value of -1 and perform a prefix-sum on the array. (*Note*: this problem is similar to the parenthesis matching/nesting problem - point out the similarity.)

Do time/space/cost analysis of RAM, PRAM, and Mesh.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.