ICS 443: Parallel Algorithms Fall 2017

Lecture 9 — September 25, 2017

Prof. Nodari Sitchinava *Scribe: Isaac DeMello, Brandon Doan, Robert Figgs*

1 Overview

In the last lecture we detailed an algorithm that finds the minimum of an array in $O(\log_2(\log_2(n)))$ time, and with $O(n(\log_2(\log_2(n)))$ work.

In this lecture we define a work-efficient algorithm that finds the minimum in $O(\log_2(\log_2(n)))$ and with $O(n)$ work. We also begin to look at searching algorithms. ....

2 Work Efficient Min

We begin by breaking down the algorithm into parts of size k, where k = $O(\log_2(\log_2(n)))$

2.1 WE-CRCW-min

In order to break down the size of the array and apply the previously defined CRCW-min algorithm covered in lecture 9. We use the following algorithm:

Algorithm 1 WE-CRCW-min(A, n)

```
k = ⌈log2(log2(n))⌉
n' = ⌈n/k⌉
B = new array of size n'
for i = 1 to n' in parallel do
    B[i]=A[(i-1)k+1]
    for j = (i − 1)k + 2 to ik in parallel do
        B[i] = min(B[i], A[j])
    end for
end for
return CRCW-min(B, 1, n')
```

2.1.1 WE-CRCW-min Algorithmic Analysis

The first 3 lines of the algorithm can be done in constant time, this means that the runtime of the algorithm is dominated by the for-loop.

Work Analysis : Work of this algorithm can be shown as follows:

$\sum_{i=1}^{n} \sum_{j=(i-1)k+2}^{ik} O(1) = \theta(n' * k) = \theta((n/k) * k) = \theta(n)$

3 Searching

We start with a trivial parallel searching algorithm. Note that this algorithm assumes that every element in the array is distinct. This algorithm will also return a value of -1 if the element is not found in the array.

Algorithm 2 parallel-search(A, n)

```
answer = -1
for i = 1 to n in parallel do
  if A[i] == x then
    answer = i
  end if
end for
return answer
```

We can see that because this all runs in parallel, our run time is $T_\infty(n) = O(1)$ and our work is $W(n) = O(n)$

If we plug these values into Brent's scheduling principle we get $T_P(n) = O(\frac{n}{p} + 1)$ This isn't a fantastic run time so lets see if we can make this any better. We'll start by reviewing the sequential algorithms for Binary Search, and Binary Search Trees. Note that these two algorithms assume that the input is sorted.

3.1 Binary Search

Binary search is a sequential algorithm. It can only be used on a pre-sorted array. This algorithm returns the index of the of the element if it is found between the left and right indices . Otherwise it will return -1.

Algorithm 3 Binary-Search(A, l, r, x)

```
if r < l then
  return -1
else
  mid = ⌊(l+r)/2⌋
  if A[mid] == x then
    return mid
  else if x < A[mid] then
    return Binary-Search(A, l, mid − 1, x)
  else
    return Binary-Search(A, mid + 1, r, x)
  end if
end if
```

The run time for this algorithm is $T(n) = T(\frac{n}{2}) + O(1)$. Using master theorem we get $T(n) = O(\log n)$

3.2 Binary Search Tree

A binary search tree is a special type of binary tree. The only difference is that in binary search tree, the children on the left of the node all are less than the value of the node, and the children on the right side of the node are all greater than the node. To search through the Binary Search Tree we can use the following sequential algorithm. Note that this algorithm returns nil if the element is not found.

Algorithm 4 BST-Search$(root, x)$

```
if root == nil or root.key == x then
  return root
else if x < root.key then
  return BST-Search(root.left, x)
else
  return BST-Search(root.right, x)
end if
```

The run-time of this algorithm depends on how well balanced the binary search tree was before running this algorithm.

Question: How would you use boldface?

Example: This is an example showing how to use boldface to help organize your lectures.

Some Formatting. Here is some formatting that you can use in your notes:

- *Item One* – This is the first item.

- *Item Two* – This is the second item.
- . . . and here are other items.

If you need to number things, you can use this style:

1. *Item One* – Again, this is the first item.
2. *Item Two* – Again, this is the second item.
3. . . . and here are other items.