ICS 443: Parallel Algorithms Fall 2017

Lecture 12 — October 2, 2017

Prof. Nodari Sitchinava *Scribe: Bobby Figgs, Jipeng Huang*

1 Overview

In the last lecture we did a rough over view of how a work-efficient $O(\log n)$ time merging algorithm might work.

In this lecture we will review how the work-efficient algorithm works, and then show the pseudo code to implement this algorithm.

2 Parallel Merging

In this section, we will give an overview of how the parallel merging algorithm should work. We start by stating our assumptions. We are assuming that all elements in both arrays are distinct. We are also assuming that the two arrays are sorted and of the same size. Let's now call our two input arrays A and B. We will call the output array C. What we want this algorithm to do is have each processor grab a chunk of values from A and merge it with the appropriate values from B.

The first step the algorithm takes, is to split array A into sub arrays of size k. Now we want to grab the last element of each of these sub-arrays. We will label the index as $a_k, a_{k+1}, ...$ These values represent the bounds of our chunks of values from A, so as we can see, we have a chunk that we would like to merge that starts at a_k and ends at a_{k+1}. But now we need to find which value in B are appropriate to merge with this chunk.

The next step is to find the values in B that we can safely merge with. To do this we need to use the rank function. The rank function comes in the following notation $rank_Z(x)$. Where the value of this function is equal to the index of the largest value in Z that is less than or equal to x. This is useful for us in finding the bounds of the chunk of values in B. We can find the left bound of the B chunk of values by plugging the value of left-bound of A (a_k) into the rank function. E.g. $rank_B(A[a_k])$. Similarly we can obtain the right-bound of the B chunk with, $rank_B(A[a_{k+1}])$.

Now that we have our chunks defined for both arrays, we can define the chunk in array C lets call it $C[...]$. We can now say that $C[...]$ is equal to the union of our chunk from A and our chunk from B. Formally, $C[...] = A[a_k + 1...a_{k+1}] \cup B[rank_B(A[a_k]) + 1...rank_B(A[a_{k+1}])]$. We have selected our chunk in such a way that, everything before our chunk in A is less than every value in our chunk in C, and every value after our chunk in A is greater than every value in C. Similarly everything before our chunk in B is less than everything in C and everything after our chunk in B is greater than everything in C. Here I will prove that everything before the chunk in A is less than everything in C

2.1 Proof

We want to prove that all elements in A less than or equal to index a_k is less than every element in C. Formally, $\forall i \leq a_k : A[i] < C[...]$. We know that $C[...] = A[a_k + 1...a_{k+1}] \cup B[rank_B(A[a_k]) + 1...rank_B(A[a_{k+1}])]$. Since $C[...]$ consists of a chunk from A and a chunk from B, we need to show that $\forall i \leq a_k : A[i] < A[a_k + 1...a_{k+1}]$ AND $\forall i \leq a_k : A[i] < B[rank_B(A[a_k]) + 1...rank_B(A[a_{k+1}])]$

Let's start by proving, $\forall i \leq a_k : A[i] < A[a_k + 1...a_{k+1}]$. This holds trivially because we know that array A is sorted, and every element is distinct. Every element in a sorted array is less than the elements after it, and since our array starts at $a_k + 1$ then a_k must be less than it.

Now let's prove $\forall i \leq a_k : A[i] < B[rank_B(A[a_k])+1...rank_B(A[a_{k+1}])]$. We know that both of these arrays are sorted, so all we need to do is prove that all of our values are less than the first element. If our values are less than the first element than we know that they are less than the entire array. We also know that a_k is the last element before our chunk in A, meaning that it is also the largest value in that area. Since array A is sorted we just need to show that $A[a_k] < B[rank_B(A[a_k]) + 1]$. We know that the rank function will give us the largest value in B that is less than or equal to a_k. But since every element is distinct it will actually be less than a_k. But if you look at the equation we are not looking at the rank of a_k, we are looking at the element after it. And since we already know that B is sorted, we know that a_k is less than every value in our chunk of B.

Since we have shown that $\forall i \leq a_k : A[i] < A[a_k+1...a_{k+1}]$ AND $\forall i \leq a_k : A[i] < B[rank_B(A[a_k]) + 1...rank_B(A[a_{k+1}])]$ is true. Then $\forall i \leq a_k : A[i] < C[...]$ is also true.

2.2 Blah blah blah

Here is a subsection.

2.2.1 Blah blah blah

Here is a subsubsection. You can use these as well.

2.3 Using Boldface

Make sure to use lots of boldface.

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.