15499 Assignment 1: solutions

January 28, 2009

1 Karatsuba Integer multiplication

1.1 A

Each invokation of **km** with size n integers calls invokes **km** with problem size $n/2$ thrice, all in parallel. It also performs a constant number of additions and shift operations that put together takes $O(n)$ work and $O(logn)$ depth.
$W(n) = 3W(n/2) + O(\log n) \implies W(n) = O(n^{\log_2 3})$
$D(n) = D(n/2) + O(n) \implies D(n) = O(n)$

1.2 B

While the work remains the same, depth is $D(n) = D(n/2) + O(\log n) = O(\log^2 n)$

2 UnSchur Inversion

Work: $W(n) = 2W(n/2) + O(n^3) = O(n^3)$.

Notice that D^{-1} is needed for computing S^{-1}. These operations can not be done in parallel. Therefore, $D(n) = 2D(n/2) + O(\log n) = O(n)$.

3 Parallel Merging

There are several ways to do this. I will describe one and briefly describe other variants.

Algorithm 1 MERGE$(C, s_C, A, s_A, l_A, B, s_B, l_B)$

$A[s_A : s_A + l_A - 1]$ and $B[s_B : s_B + l_B - 1]$ are merged in to C, starting at location $C[s_C]$
if $l_A == 0$ **then**
  Copy $B[s_B : s_B + l_B - 1]$ in to $C[s_C : s_C + l_B - 1]$
  RETURN null
else if $l_B == 0$ **then**
  Copy $A[s_A : s_A + l_A - 1]$ in to $C[s_C : s_C + l_A - 1]$
  RETURN null
end if
if $l_A == 1$ **then**
  Binary search B to find $p_B, s_B \leq p_B \leq s_B + p_B - 1$ such that $B[p_B - 1] \leq A[s_A] \leq B[p_B]$
  Copy $B[s_B : s_B + p_B - 1]$ in to $C[s_C + p_C - 1]$
  $C[s_C + p_C] \leftarrow A[s_A]$
  Copy $B[s_B + p_B : s_B + l_B - 1]$ in to $C[s_C + p_C + 1]$
  RETURN null
else if $l_B == 1$ **then**
  Similar as above
  RETURN null
end if
if $l_A \geq l_B$ **then**
  Binary search B to find $p_B, s_B \leq p_B \leq s_B + p_B - 1$ such that $B[p_B - 1] \leq A[s_A + \lceil l_A/2 \rceil] \leq B[p_B]$.
  MERGE$(C, s_C, A, s_A, \lfloor l_A/2 \rfloor, B, s_B, p_B)$
  MERGE$(C, s_C + \lfloor l_A/2 \rfloor + p_B - 1, A, s_A + \lfloor l_A/2 \rfloor, l_A - \lfloor l_A/2 \rfloor, B, s_B + p_B, l_B - p_B)$
  RETURN null
else
  Similar as above, except that we split B segment in half
  RETURN null
end if

This algorithm takes the median element m of the larger of the two inputs $A[s_A : s_+ l_A - 1]$ and $B[s_B : s_B + l_B - 1]$, and searches the smaller input for m to get pivot p. After that, it uses m as pivot for the larger input and p as pivot for the smaller input to cut the two inputs in two parts each. It then pairs off the two segments with the smaller elements and merges them recursively. The two segments with larger elements are similarly handled.

These means that in two level of resursion, both the input arrays are halved in size.

I will add the analysis for this by tonight.

3.1 Variation 1

Instead of taking the median of the larger of the inputs, we can keep computing the median of the fist input A and use this to search the pivot of the second input B.

This results in skewed sizes for the second input across the lower levels of recursion. However, we will show that this is not a problem.

To see how, lets start with arrays A and B, both of length n. In the recursion tree, at depth d, there are 2^k calls to the merge procedure, say MERGE$(A_{d,i}, B_{d,i}), 1 \leq i \leq 2^d$. Each of the $A_{d,i}$ is exactly of size $n/2^d$, and $B_{d,i}$s are such that $\sum_i |B_{d,i}| = n$. To split the problem MERGE$(A_{d,i}, B_{d,i})$, we need a binary search on $B_{d,i}$ which costs $\log |B_{d,i}|$. Therefore,

$$W(n,n) = \sum_{d=1}^{\log n} \left[\sum_{i} \log |B_{d,i}| + c \right], \text{where } \ c \ \text{ is a constant} \tag{1}$$

However, since $\log x$ is a convex function, by Jensen's inequality:

$$\sum_{i=1, \sum_i x_i = n}^{2^k} \log x_i \leq \sum_{i=1}^{2^k} \log n/2^k = 2^k \log n/2^k \tag{2}$$

Therefore,

$$W(n,n) < \sum_{d=1}^{\log n} 2^d (\log n/2^d + c) = O(n). \tag{3}$$

That depth is $O(\log^2 n)$ is clear from the fact that each recursion level has depth $O(\log n)$ and that the recursion stops whenever elements $A_{d,i}$ becomes smaller than 2, which happens by recursion level $O(1 + \lceil \log_2 n \rceil)$.

3.2 Variation 2

Instead of taking the median of one input and searching the other, it is possible to compute the median of the union of both the inputs in logarithmic work and depth. We can then use this common median as pivot to split both the inputs and proceed as usual. If we let n denote the combined sizes of both the inputs to such as algorithm,
$W(n) = 2W(n/2) + O(\log n)$ and
$D(n) = D(n/2) + O(\log n)$.
These equations solve out to $W(n) = O(n)$ and $D(n) = O(\log^2 n)$.