CSE 613: Parallel Programming

Lecture 8
(Parallel Quicksort and Selection)

Rezaul A. Chowdhury
Department of Computer Science
SUNY Stony Brook
Spring 2019

Parallel Partition

Input: An array $A[\, q : r \,]$ of distinct elements, and an element x from $A[\, q : r \,]$.

Output: Rearrange the elements of $A[\, q : r \,]$, and return an index $k \in [\, q, r \,]$, such that all elements in $A[\, q : k - 1 \,]$ are smaller than x, all elements in A$[\, k + 1 : r \,]$ are larger than x, and $A[\, k \,] = x$.

```
Par-Partition ( A[ q : r ], x )
    n ← r − q + 1
    if n = 1 then return q
    array  B[ 0: n − 1 ], lt[ 0: n − 1 ], gt[ 0: n − 1 ]
    parallel for i ← 0 to n − 1 do
        B[ i ] ← A[ q + i ]
        if B[ i ] < x then lt[ i ] ← 1 else lt[ i ] ← 0
        if B[ i ] > x then gt[ i ] ← 1 else gt[ i ] ← 0
    lt [ 0: n − 1 ] ← Par-Prefix-Sum (  lt[ 0: n − 1 ],  + )
    gt[ 0: n − 1 ] ← Par-Prefix-Sum (  gt[ 0: n − 1 ],  + )
    k ← q + lt [ n − 1 ],  A[ k ] ← x
    parallel for i ← 0 to n − 1 do
        if B[ i ] < x then A[ q + lt [ i ] − 1 ] ← B[ i ]
        else if B[ i ] > x then A[ k + gt [ i ] ] ← B[ i ]
    return k
```

Parallel Partition

A:

9	5	7	11	1	3	8	14	4	21

$x = 8$

Parallel Partition

***A*:**

9	5	7	11	1	3	8	14	4	21

***x* = 8**

***B*:**

0	1	2	3	4	5	6	7	8	9
9	5	7	11	1	3	8	14	4	21

***lt*:**

0	1	2	3	4	5	6	7	8	9
0	1	1	0	1	1	0	0	1	0

***gt*:**

0	1	2	3	4	5	6	7	8	9
1	0	0	1	0	0	0	1	0	1

Parallel Partition

A:

9	5	7	11	1	3	8	14	4	21

***x* = 8**

B:

0	1	2	3	4	5	6	7	8	9
9	5	7	11	1	3	8	14	4	21

***lt*:**

0	1	2	3	4	5	6	7	8	9
0	1	1	0	1	1	0	0	1	0

***lt*:**

0	1	2	2	3	4	4	4	5	5

prefix sum

***gt*:**

0	1	2	3	4	5	6	7	8	9
1	0	0	1	0	0	0	1	0	1

***gt*:**

1	1	1	2	2	2	2	3	3	4

prefix sum

Parallel Partition

***A*:**	9	5	7	11	1	3	8	14	4	21

***x* = 8**

	0	1	2	3	4	5	6	7	8	9
***B*:**	9	5	7	11	1	3	8	14	4	21

	0	1	2	3	4	5	6	7	8	9
***lt*:**	0	1	1	0	1	1	0	0	1	0
***lt*:**	0	1	2	2	3	4	4	4	5	5

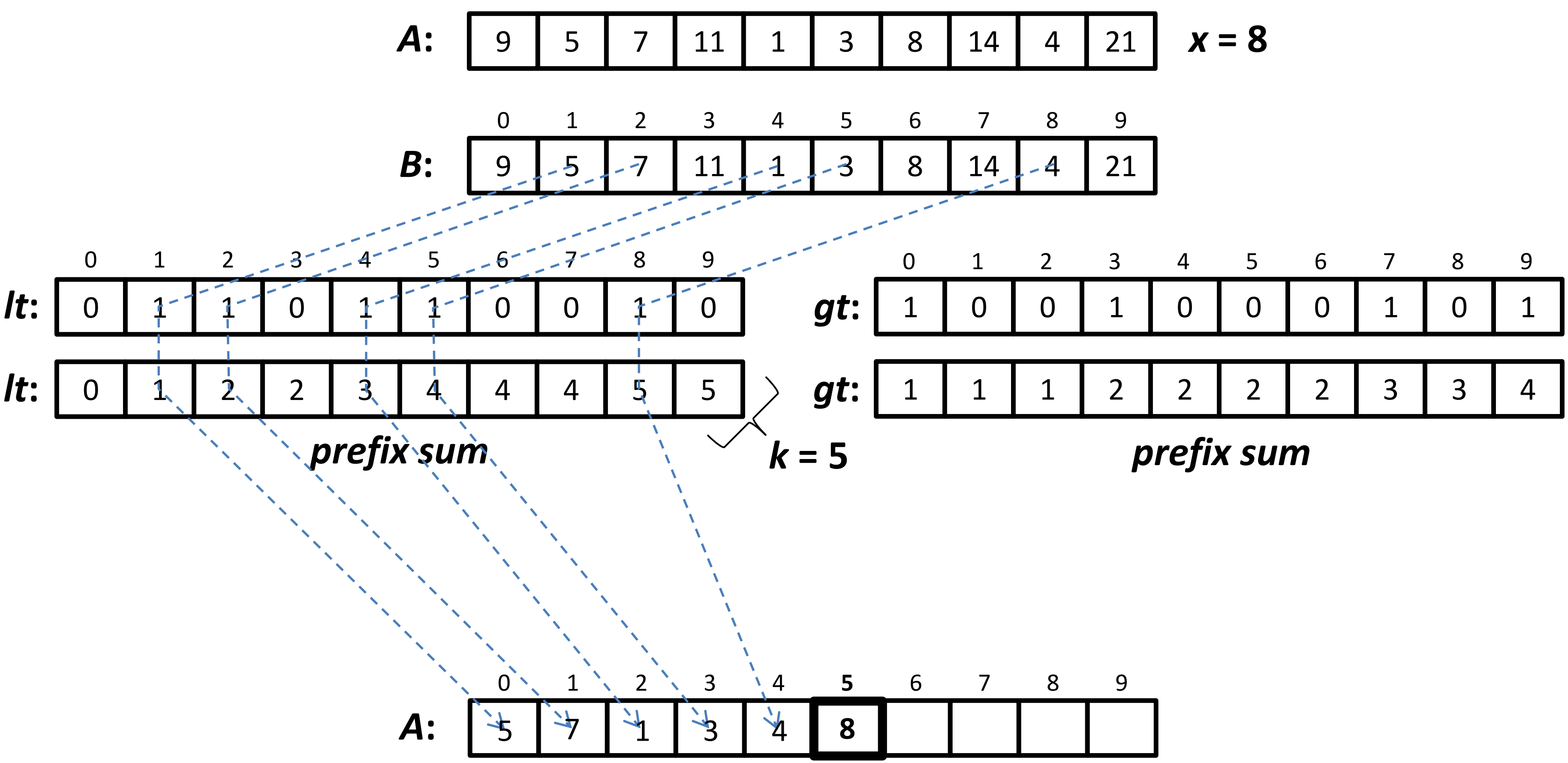
prefix sum

	0	1	2	3	4	5	6	7	8	9
***gt*:**	1	0	0	1	0	0	0	1	0	1
***gt*:**	1	1	1	2	2	2	2	3	3	4

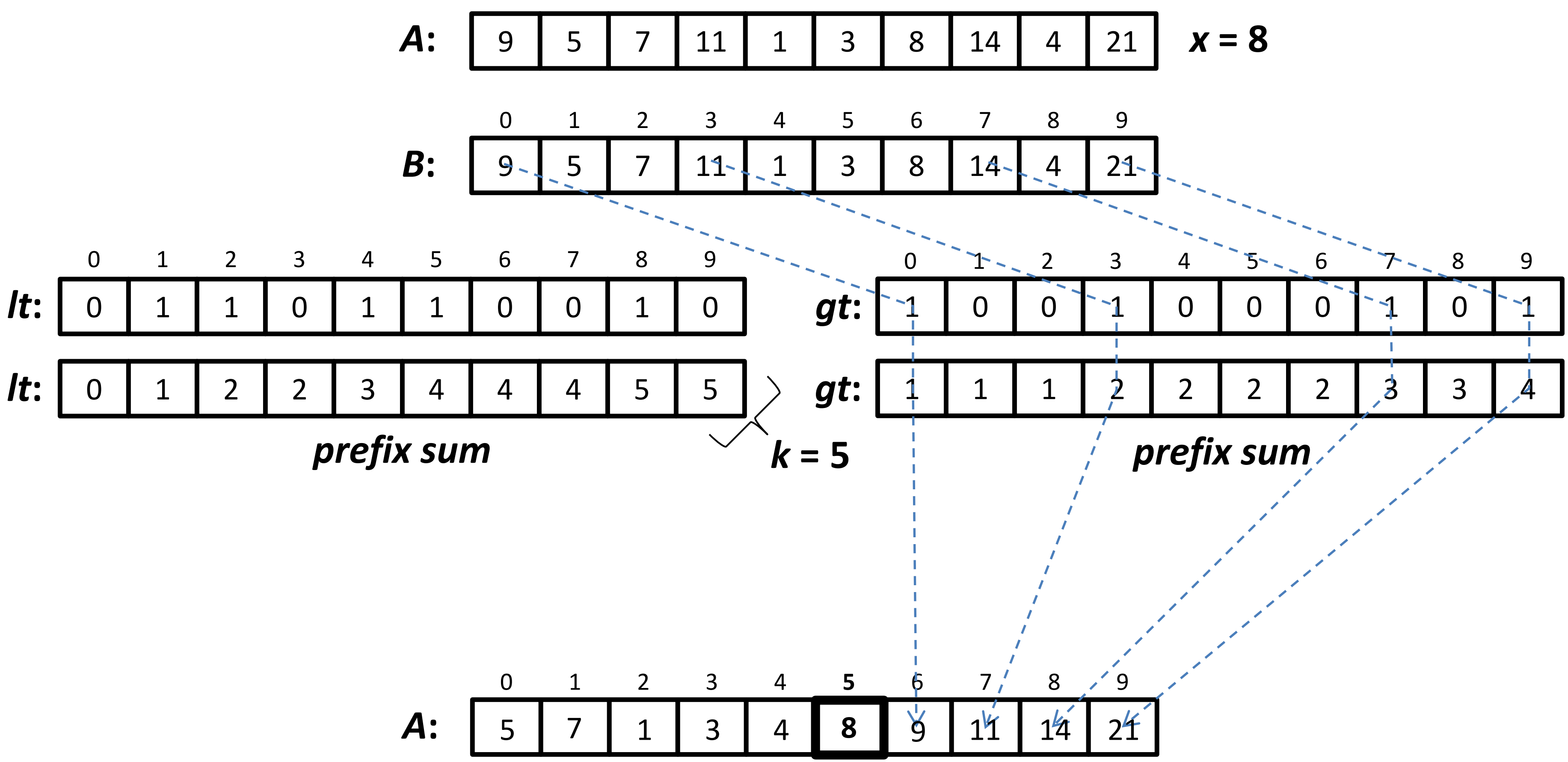
prefix sum

	0	1	2	3	4	5	6	7	8	9
***A*:**										

Parallel Partition



Parallel Partition



Parallel Partition

***A*:**	9	5	7	11	1	3	8	14	4	21

***x* = 8**

	0	1	2	3	4	5	6	7	8	9
***B*:**	9	5	7	11	1	3	8	14	4	21

	0	1	2	3	4	5	6	7	8	9
***lt*:**	0	1	1	0	1	1	0	0	1	0
***lt*:**	0	1	2	2	3	4	4	4	5	5

prefix sum

***k* = 5**

	0	1	2	3	4	5	6	7	8	9
***gt*:**	1	0	0	1	0	0	0	1	0	1
***gt*:**	1	1	1	2	2	2	2	3	3	4

prefix sum

	0	1	2	3	4	**5**	6	7	8	9
***A*:**	5	7	1	3	4	**8**	9	11	14	21

Parallel Partition: Analysis

```
Par-Partition ( A[ q : r ], x )
1.  n ← r − q + 1
2.  if n = 1 then return q
3.  array B[ 0: n − 1 ], lt[ 0: n − 1 ], gt[ 0: n − 1 ]
4.  parallel for i ← 0 to n − 1 do
5.      B[ i ] ← A[ q + i ]
6.      if B[ i ] < x then lt[ i ] ← 1 else lt[ i ] ← 0
7.      if B[ i ] > x then gt[ i ] ← 1 else gt[ i ] ← 0
8.  lt [ 0: n − 1 ] ← Par-Prefix-Sum ( lt[ 0: n − 1 ], + )
9.  gt[ 0: n − 1 ] ← Par-Prefix-Sum ( gt[ 0: n − 1 ], + )
10. k ← q + lt [ n − 1 ], A[ k ] ← x
11. parallel for i ← 0 to n − 1 do
12.     if B[ i ] < x then A[ q + lt [ i ] − 1 ] ← B[ i ]
13.     else if B[ i ] > x then A[ k + gt [ i ] ] ← B[ i ]
14. return k
```

Work:

$$\begin{aligned} T_1(n) &= \Theta(n) && [\text{ lines } 1-7\text{ }] \\ &+ \Theta(n) && [\text{ lines } 8-9\text{ }] \\ &+ \Theta(n) && [\text{ lines } 10-14\text{ }] \\ &= \Theta(n) \end{aligned}$$

Span:

Assuming $\log n$ depth for *parallel for* loops:

$$\begin{aligned} T_\infty(n) &= \Theta(\log n) && [\text{ lines } 1-7\text{ }] \\ &+ \Theta(\log^2 n) && [\text{ lines } 8-9\text{ }] \\ &+ \Theta(\log n) && [\text{ lines } 10-14\text{ }] \\ &= \Theta(\log^2 n) \end{aligned}$$

Parallelism: $\dfrac{T_1(n)}{T_\infty(n)} = \Theta\left(\dfrac{n}{\log^2 n}\right)$

Randomized Parallel QuickSort

Input: An array $A[\ q : r\ ]$ of distinct elements.

Output: Elements of $A[\ q : r\ ]$ sorted in increasing order of value.

Par-Randomized-QuickSort ($A[\ q : r\ ]$)

1. $n \leftarrow r - q + 1$
2. *if* $n \leq 30$ *then*
3. sort $A[\ q : r\ ]$ using any sorting algorithm
4. *else*
5. select a random element x from $A[\ q : r\ ]$
6. $k \leftarrow$ *Par-Partition* ($A[\ q : r\ ]$, x)
7. *spawn* *Par-Randomized-QuickSort* ($A[\ q : k - 1\ ]$)
8. *Par-Randomized-QuickSort* ($A[\ k + 1 : r\ ]$)
9. *sync*

Randomized Parallel QuickSort: Analysis

```
Par-Randomized-QuickSort ( A[ q : r ] )
1.  n ← r − q + 1
2.  if n ≤ 30 then
3.      sort A[ q : r ] using any sorting algorithm
4.  else
5.      select a random element x from A[ q : r ]
6.      k ← Par-Partition ( A[ q : r ], x )
7.      spawn Par-Randomized-QuickSort ( A[ q : k − 1 ] )
8.      Par-Randomized-QuickSort ( A[ k + 1 : r ] )
9.      sync
```

Lines 1—6 take $\Theta(\log^2 n)$ parallel time and perform $\Theta(n)$ work.

Also the recursive spawns in lines 7—8 work on disjoint parts of $A[\, q : r\,]$. So the upper bounds on the parallel time and the total work in each level of recursion are $\Theta(\log^2 n)$ and $\Theta(n)$, respectively.

Hence, if D is the *recursion depth* of the algorithm, then

$$T_1(n) = \mathrm{O}(nD) \text{ and } T_\infty(n) = \mathrm{O}(D \log^2 n)$$

Randomized Parallel QuickSort: Analysis

```
Par-Randomized-QuickSort ( A[ q : r ] )
n ← r − q + 1
if n ≤ 30 then
    sort A[ q : r ] using any sorting algorithm
else
    select a random element x from A[ q : r ]
    k ← Par-Partition (  A[ q : r ],  x )
    spawn Par-Randomized-QuickSort ( A[ q : k − 1 ] )
    Par-Randomized-QuickSort ( A[ k + 1 : r ] )
    sync
```

We will show that w.h.p. recursion depth, $D = \mathrm{O}(\log n)$.

Hence, with high probability,

$$T_1(n) = \mathrm{O}(n \log n) \text{ and } T_\infty(n) = \mathrm{O}(\log^3 n)$$

Randomized Parallel QuickSort: Analysis

Approach: We will show the following

1. For any specific element v, the sizes of the partitions containing v in any two consecutive levels of recursion decrease by a constant factor with a certain probability.
2. With probability $1 - \mathrm{O}\left(\frac{1}{n^6}\right)$, the partition containing v will be of size 30 or less after $\mathrm{O}(\log n)$ levels of recursion.
3. With probability $1 - \mathrm{O}\left(\frac{1}{n^5}\right)$, the partition containing every element will be of size 30 or less after $\mathrm{O}(\log n)$ levels of recursion.

Randomized Parallel QuickSort: Analysis

Lemma 1: Let v be an arbitrary element of the original input array A of size $n = n_0$, and let n_j be the size of the partition containing v after partitioning at recursion depth $j \geq 1$. Then for any $j \geq 0$,

$$\Pr\left[n_{j+1} \geq \frac{7}{8} n_j\right] \leq \frac{1}{4}.$$

Proof: Suppose at recursion depth $j + 1 \geq 1$ element x was chosen as the pivot element.

One of the new partitions will have at least $\frac{7}{8} n_j$ elements provided x is among the smallest or largest $\frac{1}{8} n_j$ elements in the old partition.

The probability that x is among the smallest or largest $\frac{1}{8} n_j$ elements in the old partition is clearly $\leq \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$.

Randomized Parallel QuickSort: Analysis

Lemma 2: In $20 \log n$ levels of recursion, the probability that an element goes through $20 \log n - \log(n/30)$ *unsuccessful partitioning steps* (i.e., partitioning steps with $n_{j+1} \geq (7/8) n_j$) is $\mathrm{O}\left(1/n^6\right)$.

[all logarithms are to the base 8/7]

Proof: The events consisting of the partitioning steps being successful can be modeled as *Bernoulli trials*.

Let X be a random variable denoting the number of unsuccessful partitioning steps among the $20 \log n$ steps. Then

$$\Pr[X > 20 \log n - \log(n/30)] \leq \Pr[X > 19 \log n]$$

$$\leq \sum_{j > 19 \log n} \binom{20 \log n}{j} \left(\frac{1}{4}\right)^{j} \left(\frac{3}{4}\right)^{20 \log n - j}$$

Randomized Parallel QuickSort: Analysis

Lemma 2: In $20 \log n$ levels of recursion, the probability that an element goes through $20 \log n - \log(n/30)$ *unsuccessful partitioning steps* (i.e., partitioning steps with $n_{j+1} \geq (7/8)n_j$) is $\mathrm{O}\left(1/n^6\right)$.

[all logarithms are to the base 8/7]

Proof: $\Pr[X > 20 \log n - \log(n/30)] \leq \Pr[X > 19 \log n]$

$$\leq \sum_{j > 19 \log n} \binom{20 \log n}{j} \left(\frac{1}{4}\right)^j \left(\frac{3}{4}\right)^{20 \log n - j}$$

$$\leq \sum_{j > 19 \log n} \left(\frac{20e \log n}{j}\right)^j \left(\frac{1}{3}\right)^j = \sum_{j > 19 \log n} \left(\frac{20e \log n}{3j}\right)^j$$

$$\leq \sum_{j > 19 \log n} \left(\frac{20e \log n}{57 \log n}\right)^j = \sum_{j > 19 \log n} \left(\frac{20e}{57}\right)^j = \mathrm{O}\left(\frac{1}{n^6}\right)$$

Randomized Parallel QuickSort: Analysis

Theorem 1: The recursion depth of *Par-Randomized-Quicksort* is $\leq 20\log_{8/7}n$ with probability $1 - \mathrm{O}\left(1/n^5\right)$.

Proof: The probability that one or more elements of *A* go through $20\log_{8/7}n - \log_{8/7}(n/30)$ unsuccessful partitioning steps is

$$\mathrm{O}\left(n \times \frac{1}{n^6}\right) = \mathrm{O}\left(\frac{1}{n^5}\right)$$ [using Lemma 2]

Hence, the probability that at least $\log_{8/7}(n/30)$ of the $20\log_{8/7}n$ partitioning steps is successful for all elements is $1 - \mathrm{O}\left(1/n^5\right)$.

After $t = \log_{8/7}(n/30)$ successful partitioning steps involving an element, the element belongs to a partition of size $\left(\frac{7}{8}\right)^t n = 30$.

Hence, *Par-Randomized-Quicksort* terminates in $\leq 20\log_{8/7}n$ levels of recursion with probability $1 - \mathrm{O}\left(1/n^5\right)$.

Parallel Selection

Input: A subarray $A[\ q : r\ ]$ of an array $A[\ 1 : n\ ]$ of n distinct elements, and a positive integer $k \in [1, r - q + 1]$.

Output: An element x of $A[\ q : r\ ]$ such that $rank(x, A[\ q : r\ ]) = k$.

```
Par-Selection ( A[ q : r ], n, k )
n′ ← r − q + 1
if n′ ≤ n / log n then
    sort A[ q : r ] using a parallel sorting algorithm and return A[ q + k - 1 ]
else
    partition A[ q : r ] into blocks B_i's each containing log n consecutive elements
    parallel for i ← 1 to ⌈ n′ / log n ⌉ do
        M[ i ] ← median of B_i using a sequential selection algorithm
    find the median m of M[ 1 : ⌈ n′ / log n ⌉ ] using a parallel sorting algorithm
    t ← Par-Partition ( A[ q : r ], m )
    if k = t − q + 1 then return A[ t ]
    else if k < t − q + 1 then return Par-Selection ( A[ q : t − 1 ], n, k )
        else return Par-Selection ( A[ t + 1 : r ], n, k − t + q − 1 )
```

Parallel Selection

Lemma 3: In *Par-Selection* (lines 11—12)

$$|A[q{:}\,t-1]| \leq \frac{3n'}{4} \text{ and } |A[t+1{:}\,r]| \leq \frac{3n'}{4}.$$

Proof: It suffices to show that $\frac{n'}{4} \leq rank(m, A[q{:}\,r]) \leq \frac{3n'}{4}$.

Since m is the median of $M[i]$'s, it is larger than one half of the $M[i]$'s. But each $M[i]$ is larger than $\frac{\log n}{2}$ elements in B_i.

Hence, $rank(m, A[q{:}\,r]) \geq \frac{n'}{2\log n} \times \frac{\log n}{2} = \frac{n'}{4}$.

Similarly, one can show that $rank(m, A[q{:}\,r]) \leq \frac{3n'}{4}$.

Parallel Selection

Lemma 4: In *Par-Selection* $n' \le \frac{n}{\log n}$ after at most $\log_{4/3} \log n$ levels of recursion.

Proof: It follows from Lemma 3 that $n' \le \left(\frac{3}{4}\right)^k n$ after k levels of recursion.

Hence, for reaching $n' \le \frac{n}{\log n}$, we need

$$\frac{n}{\log n} \ge \left(\frac{3}{4}\right)^k n \Rightarrow k \le \log_{4/3} \log n.$$

Deterministic Parallel Selection

Par-Selection ($A[\,q : r\,]$, n, k)

1. $n' \leftarrow r - q + 1$
2. *if* $n' \le n / \log n$ *then*
3. sort $A[\,q : r\,]$ using a *parallel sorting algorithm* and *return* $A[\,q + k - 1\,]$
4. *else*
5. partition $A[\,q : r\,]$ into blocks B_i's each containing $\log n$ consecutive elements
6. *parallel for* $i \leftarrow 1$ *to* $\lceil n' / \log n \rceil$ *do*
7. $M[\,i\,] \leftarrow$ median of B_i using a *sequential selection algorithm*
8. find the median m of $M[\,1 : \lceil n' / \log n \rceil\,]$ using a *parallel sorting algorithm*
9. $t \leftarrow$ *Par-Partition* ($A[\,q : r\,]$, m)
10. *if* $k = t - q + 1$ *then return* $A[\,t\,]$
11. *else if* $k < t - q + 1$ *then return* *Par-Selection* ($A[\,q : t - 1\,]$, n, k)
12. *else return* *Par-Selection* ($A[\,t + 1 : r\,]$, n, $k - t + q - 1$)

Step 7: Use a linear time (worst-case) sequential selection algorithm (see Section 9.3 of “Introduction to Algorithms”, 3rd Ed. by Cormen et al.).

Steps 3 and 8: Use the parallel mergesort with parallel merge (see Lecture 7) that runs in $O(\log^3 n)$ parallel time and performs $O(n \log n)$ work in the worst case.

Deterministic Parallel Selection

```
Par-Selection ( A[ q : r ], n, k )
1.  n′ ← r − q + 1
2.  if n′ ≤ n / log n then
3.      sort A[ q : r ] using a parallel sorting
        algorithm and return A[ q + k - 1 ]
4.  else
5.      partition A[ q : r ] into blocks B_i's each
        containing log n consecutive elements
6.      parallel for i ← 1 to ⌈ n′ / log n ⌉ do
7.          M[ i ] ← median of B_i using a
            sequential selection algorithm
8.      find the median m of M[ 1 : ⌈ n′ / log n ⌉ ]
        using a parallel sorting algorithm
9.      t ← Par-Partition ( A[ q : r ], m )
10.     if k = t − q + 1 then return A[ t ]
11.     else if k < t − q + 1 then
            return Par-Selection ( A[ q : t − 1 ], n, k )
12.     else return Par-Selection ( A[ t + 1 : r ],
                                    n, k − t + q − 1 )
```

Last Level of Recursion

Work: $O\left(\frac{n}{\log n}\log\left(\frac{n}{\log n}\right)\right) = O(n)$

Span: $O\left(\log^3\left(\frac{n}{\log n}\right)\right) = O(\log^3 n)$

Any Other Recursion Level (except last)

Work: $O\left(\frac{n'}{\log n} \times \log n\right)$ [lines 5 – 7]

$+ O\left(\frac{n'}{\log n}\log\left(\frac{n'}{\log n}\right)\right)$ [line 8]

$+ O(n')$ [line 9]

$= O(n')$

Span: $O\left(\log\left(\frac{n'}{\log n}\right) + \log n\right)$ [lines 5 – 7]

$+ O\left(\log^3\left(\frac{n'}{\log n}\right)\right)$ [line 8]

$+ O(\log^2 n')$ [line 9]

$= O(\log^3 n)$

Deterministic Parallel Selection

Par-Selection ($A[\,q : r\,]$, n, k)

1. $n' \leftarrow r - q + 1$
2. *if* $n' \leq n \,/\, \log n$ *then*
3. sort $A[\,q : r\,]$ using a *parallel sorting algorithm* and *return* $A[\,q + k - 1\,]$
4. *else*
5. partition $A[\,q : r\,]$ into blocks B_i's each containing $\log n$ consecutive elements
6. *parallel for* $i \leftarrow 1$ *to* $\lceil\, n' \,/\, \log n \,\rceil$ *do*
7. $M[\,i\,] \leftarrow$ median of B_i using a *sequential selection algorithm*
8. find the median m of $M[\,1 : \lceil\, n' \,/\, \log n\,\rceil\,]$ using a *parallel sorting algorithm*
9. $t \leftarrow$ *Par-Partition* ($A[\,q : r\,]$, m)
10. *if* $k = t - q + 1$ *then return* $A[\,t\,]$
11. *else if* $k < t - q + 1$ *then*
    return *Par-Selection* ($A[\,q : t - 1\,]$, n, k)
12. *else* *return* *Par-Selection* ($A[\,t + 1 : r\,]$, n, $k - t + q - 1$)

Overall

Work:

$$T_1(n) = O\left(n + \sum_{i=0}^{\log_{4/3} \log n} \left(\frac{3}{4}\right)^i n\right)$$
$$= O(n)$$

Span:

$$T_\infty(n) = O\left(\left(\log_{4/3} \log n\right) \log^3 n\right)$$
$$= O(\log^3 n \log \log n)$$