Principles of Distributed Computing 04/05, 2017

Lecture 7

Lecturer: Mohsen Ghaffari *Scribe:*

In this lecture, we will discuss time complexity lower bounds for some distributed coloring problems. That is, we prove that any distributed algorithm that solves the given coloring problem requires at least so many rounds. Before that, we recall the formal definition of the model of distributed algorithms that we are using, called the LOCAL model [Lin87, Lin92].

Definition 1. *(The LOCAL model) We consider an arbitrary n-node graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$, which abstracts the communication network. Unless noted otherwise, G is a simple, undirected, and unweighted graph. There is one process on each node $v \in V$ of the network. At the beginning, the processes do not know the graph G, except for knowing[1] n, and their own unique identifier in $\{1, 2, \ldots, n\}$. The algorithms work in synchronous rounds. Per round, each node/process performs some computation based on its own knowledge, and then sends a message to all of its neighbors, and then receives the messages sent to it by its neighbors in that round. In each graph problem in this model, we require that each node learns its own part of the output, e.g., its own color in a graph coloring.*

Comment: We stress that the model does not assume any limitation on the size of the messages, or on the computational power of the processes. Because of this, it is not hard to see that, any t-round algorithm in the LOCAL model induces a function which maps the t-hop neighborhood of each node to its output (why?). For instance, a t-round algorithm for graph coloring maps the topology induced by vertices within distance t of a vertex v to the coloring of vertex v. The converse of this statement is also true, meaning that if for a given graph problem, such a function exists, then there is also a t-round algorithm for solving that problem. Hence, one can say that the LOCAL model captures the *locality* of graph problems in a mathematical sense.

1 Lower Bound on Coloring Rooted Trees

We start by examining the graph coloring problem on rooted trees, where each node knows its parent. In the first lecture of this class, we saw an $O(\log^* n)$-round algorithm for computing a 3-coloring of any such graph. We now prove that this complexity is optimal. In particular, we show that even in a special case of a rooted path, $\Omega(\log^* n)$ rounds are necessary, for any (deterministic) distributed algorithm that computes a 3-coloring. This result was first proved by [Lin87, Lin92]. We explain a somewhat streamlined proof, based on [LS14]. The lower bound holds also for randomized algorithms [Nao91], but we will not cover that generalization, for the sake of simplicity. Furthermore, essentially the same lower bound can be obtained as a direct corollary of *Ramsey Theory*. We will have a brief explanation about that, at the end of this subsection.

Theorem 2. *Any deterministic algorithm for 3-coloring n-node directed paths needs at least $\frac{\log^* n}{2} - 2$ rounds.*

For the sake of contradiction, suppose that there is an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t < \frac{\log^* n}{2} - 2$. When running this algorithm for t rounds, any node v can see at most the k-neighborhood around itself for $k = 2t + 1$, that is, the vector of identifiers for the nodes up to t hops before itself and up to t hops after itself. Hence, if the algorithm $\mathcal{A}$ exists, there is a mapping from each such neighborhood to a color in $\{1, 2, 3\}$ such that neighborhoods that can be conceivably adjacent are mapped to different colors.

We next make this formal by a simple and abstract definition. For simplicity, we will consider only a restricted case of the problem where the identifiers are set monotonically increasing along the path. Notice this restriction only strengthens the lower bound, as it shows that even for this restricted case, there is no t-round algorithm for $t < \frac{\log^* n}{2} - 2$.

Definition 3. *We say B is a k-ary q-coloring if for any set of identifiers $1 \leq a_1 < a_2 < \ldots < a_k < a_{k+1} \leq n$, we have the following two properties:*

[1] Most often, the algorithms will use only the assumption that nodes know an upper bound N on n such that $N \in [n, n^c]$ for a small constant $c \geq 1$.

P1: $B(a_1, a_2, \ldots, a_k) \in \{1, 2, \ldots, q\}$,

P2: $B(a_1, a_2, \ldots, a_k) \neq B(a_2, \ldots, a_{k+1})$.

Observation 4. *If there exists a deterministic algorithm $\mathcal{A}$ for 3-coloring n-node directed paths in $t < \frac{\log^* n}{2} - 2$ rounds, then there exists a k-ary 3-coloring B, where $k = 2t + 1 < \log^* n - 3$.*

Proof. Suppose that such an algorithm $\mathcal{A}$ exists. We then produce a k-ary 3-coloring B by examining $\mathcal{A}$. For any set of identifiers $1 \leq a_1 < a_2 < \ldots < a_k \leq n$, define $B(a_1, a_2, \ldots, a_k)$ as follows. Simulate algorithm $\mathcal{A}$ on an imaginary directed path where a consecutive portion of the identifiers on the path are set equal to $a_1, a_2, \ldots, a_k$. Then, let $B(a_1, a_2, \ldots, a_k)$ be equal to the color in $\{1, 2, 3\}$ that the node a_{t+1} receives in this simulation.

We now argue that B as defined above is a k-ary 3-coloring. Property P1 holds trivially. We now argue that property P2 also holds. For the sake of contradiction, suppose that it does not, meaning that there exists a set of identifiers $1 \leq a_1 < a_2 < \ldots < a_k < a_{k+1} \leq n$ such that $B(a_1, a_2, \ldots, a_k) = B(a_2, \ldots, a_{k+1})$. Then, imagine running algorithm $\mathcal{A}$ on an imaginary directed path where a consecutive portion of identifiers are set equal to $a_1, a_2, \ldots, a_{2t+2}$. Then, since $B(a_1, a_2, \ldots, a_k) = B(a_2, \ldots, a_{k+1})$, the algorithm $\mathcal{A}$ assigns the same color to a_{t+1} and a_{t+2}. This is in contradiction with $\mathcal{A}$ being a 3-coloring algorithm. □

To prove Theorem 2, we show that a k-ary 3-coloring B where $k < \log^* n - 3$ cannot exist. The proof is based on the following two lemmas:

Lemma 5. *There is no 1-ary q-coloring with $q < n$.*

Proof. A 1-ary q-coloring requires that $B(a_1) \neq B(a_2)$, for any two identifiers $1 \leq a_1 < a_2 \leq n$. By the Pigeonhole principle, this needs $q \geq n$. □

Lemma 6. *If there is a k-ary q-coloring B, then there exists a $(k-1)$-ary 2^q-coloring B'.*

Proof. For any set of identifiers $1 \leq a_1 < a_2 < \ldots < a_{k-1} \leq n$, define $B'(a_1, a_2, \ldots, a_{k-1})$ to be the set of all possible colors $i \in \{1, \ldots, q\}$ for which $\exists a_k > a_{k-1}$ *such that* $B(a_1, a_2, \ldots, a_{k-1}, a_k) = i$.

Notice that B' is a subset of $\{1, \ldots, q\}$. Hence, it has 2^q possibilities, which means that B' has property P1 and it assigns each set of identifiers $1 \leq a_1 < a_2 < \ldots < a_{k-1} \leq n$ to a number in 2^q. Now we argue that B' also satisfies property P2.

For the sake of contradiction, suppose that there exist identifiers $1 \leq a_1 < a_2 < \ldots < a_k \leq n$ such that $B'(a_1, a_2, \ldots, a_{k-1}) = B'(a_2, a_3, \ldots, a_k)$. Let $q^* = B(a_1, a_2, \ldots, a_k) \in B'(a_1, a_2, \ldots, a_{k-1})$. Then, we must have $q^* \in B'(a_2, a_3, \ldots, a_k)$. Thus, $\exists a_{k+1} > a_k$ such that $q^* = B(a_2, a_3, \ldots, a_k, a_{k+1})$. But, this means $B(a_1, a_2, \ldots, a_k) = q^* = B(a_2, a_3, \ldots, a_k, a_{k+1})$, which is in contradiction with B being a k-ary q-coloring. Having reached a contradiction by assuming that B' does not satisfy P2, we conclude that it actually does satisfy P2. Hence, B' is a $(k-1)$-ary 2^q-coloring. □

Proof of Theorem 2. For the sake of contradiction, suppose that there is an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t < \frac{\log^* n}{2} - 2$. As stated in Observation 4, if there exists an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t < \frac{\log^* n}{2} - 2$, then there exists a k-ary 3-coloring B, where $k = 2t + 1 < \log^* n - 3$. Using one iteration of Lemma 6, we would get that there exists a $(k-1)$-ary 8-coloring. Another iteration would imply that there exists a $(k-2)$-ary 2^8-coloring. Repeating this, after $k < \log^* n - 3$ iterations, we would get a 1-ary coloring with less than n colors. However, this is in contradiction with Lemma 5. Hence, such an algorithm $\mathcal{A}$ cannot exist. □

An Alternative Lower Bound Proof Via Ramsey Theory:
Let us first briefly recall the basics of Ramsey Theory. The simplest case of Ramsey's theorem says that for any ℓ, there exists a number $R(\ell)$ such that for any $n \geq R(\ell)$, if we color the edges of the n-node complete graph K_n with two colors, there exists a monochromatic clique of size ℓ in it, that is, a set of ℓ vertices such that all of the edges between them have the same color. A simple example is that among any group of at least $6 = R(3)$ people, there are either at least 3 of them which are friends, or at least 3 of them no two of which are friends.

A similar statement is true in hypergraphs. Of particular interest for our case is coloring hyperedges of a complete n-vertex hypergraph of rank k, that is, the hypergraph where every subset of size k of the vertices defines one hyperedge. By Ramsey theory, it is known that there exists an n_0 such that, if $n \geq n_0$, for any way of coloring hyperedges of the complete n-vertex hypergraph of rank k with 3 colors,

there would be a monochromatic clique of size $k+1$. That is, there would be a set of $k+1$ vertices a_1, ..., a_k in $\{1,\ldots,n\}$ such that all of their $\binom{k+1}{k} = k$ subsets with cardinality k have the same color.

In particular, consider an arbitrary k-ary coloring B, and let B define the colors of the hyperedges $\{a_1,\ldots,a_k\}$ when $1 \leq a_1 < a_2 < \ldots < a_k \leq n$. For the remaining hyperedges, color them arbitrarily. By Ramsey's theorem, we would get the following: there exist vertices $1 \leq a_1 < a_2 < \ldots < a_k < a_{k+1} \leq n$ such that B assigns the same color to hyperedges $\{a_1,\ldots,a_k\}$ and $\{a_2,\ldots,a_{k+1}\}$. But this is in contradiction with the property P2 of B being a k-ary coloring. The value of n_0 that follows from Ramsey theory is such that $k = O(\log^* n_0)$. In other words, Ramsey's theorem rules out $o(\log^* n)$-round 3-coloring algorithms for directed paths. See [CFS10] for more on hypergraph Ramsey numbers.

2 Coloring Unrooted Trees

In the first lecture of the class, we saw that on a rooted tree, where each node knows its parent, a 3-coloring can be computed distributedly in $O(\log^* n)$ rounds. In the previous section, we proved that this round complexity is optimal. This $O(\log^* n)$-round algorithm for rooted trees heavily relies on each node knowing its parent.

We next prove that no such result is possible in unrooted trees, when nodes do not know which neighbor is their parent. More concretely, we prove that any deterministic algorithm for coloring unrooted trees that runs in $o(\log_\Delta n)$ rounds must use at least $\Omega(\Delta/\log\Delta)$ colors. Moreover, we complement this by showing that given $O(\log n)$ rounds, we can compute a 3-coloring of any n-node tree.

2.1 The Lower Bound

Theorem 7. *Any (deterministic) distributed algorithm $\mathcal{A}$ that colors n-node trees with maximum degree Δ using less than $o(\Delta/\log\Delta)$ colors has round complexity at least $\Omega(\log_\Delta n)$.*

To prove the claimed lower bound, we will use a graph-theoretic result about the existence of certain graphs. The proof of this lemma is based on a *probabilistic method* argument, but we do not cover it in this lecture. Before stating the properties of this graph, we recall that the *girth* of a graph is the length of the shortest cycle.

Fact 8 (Bollobas [Bol78])**.** *There exists an n-node graph H^* where all nodes have degree Δ, with girth $g(H^*) = \Omega(\log_\Delta n)$ and chromatic number $\chi(H^*) = \Omega(\Delta/\log\Delta)$.*

Remark We note that this lower bound on the chromatic number is asymptotically tight, because high-girth graphs, and more generally triangle-free graphs, with maximum degree Δ have chromatic number $O(\Delta/\log\Delta)$ [Kim95, Jam11, PS15].

Proof of Theorem 7. For the sake of contradiction, suppose that there exists a deterministic distributed algorithm $\mathcal{A}$ that computes a $o(\Delta/\log\Delta)$-coloring of any n-node tree with maximum degree Δ, in $o(\log_\Delta n)$ rounds.

We run $\mathcal{A}$ on the graph H^*, stated in Fact 8. Notice that H^* is not a tree. However, since $g(H^*) = \Omega(\log_\Delta n)$, within the $o(\log_\Delta n)$ rounds of the algorithm, no one will notice! In particular, since $g(H^*) = \Omega(\log_\Delta n)$, for any node v, the subgraph T_v of H^* induced by nodes within distance $o(\log_\Delta n)$ of v is a tree (why?). Thus, within the $o(\log_\Delta n)$ rounds of the algorithm, node v will think that the algorithm $\mathcal{A}$ is being run on T_v and will not realize that the algorithm is being run on a non-tree graph H^*. Similarly, none of the nodes will recognize that we are not on a tree. Hence, each node v will compute an output as if the algorithm $\mathcal{A}$ was being run on its local tree T_v. This must produce a valid coloring of H^* with $o(\Delta/\log\Delta)$ colors. That is because if the algorithm creates two neighbors v and u with the same color, then running the algorithm on the tree T_v would also produce a non-valid color. However, the fact that $\mathcal{A}$ is able to compute a $o(\Delta/\log\Delta)$-coloring of H^* is in contradiction with the fact that $\chi(H^*) = \Omega(\Delta/\log\Delta)$. □

2.2 The Upper Bound

Theorem 9. *There is a deterministic distributed algorithm that computes a 3-coloring of any n-node tree in $O(\log n)$ rounds.*

The Algorithm for Coloring Unrooted Trees, Step 1 We first perform an iterated peeling process, on the given tree $T = (V, E)$. Let $T_1 = T$ and let layer L_1 be the set of all vertices of T_1 whose degree in T_1 is at most 2. Then, let $T_2 = T_1 \setminus L_1$ be the forest obtained by removing from T_1 all the L_1 vertices. Then, define layer L_2 be the set of all vertices of T_2 whose degree in T_2 is at most 2. Then, define $T_3 = T_2 \setminus L_2$ similar to before. More generally, each T_{i+1} is defined as $T_{i+1} = T_i \setminus L_i$, which is the forest obtained by removing from T_i all the layer L_i vertices, and then layer L_{i+1} is defined to be the vertices that have degree at most 2 in T_{i+1}.

Lemma 10. *The process terminates in $O(\log n)$ iterations, meaning that V gets decomposed into disjoint sets L_1, L_2, ..., L_ℓ for some $\ell = O(\log n)$.*

Proof. T_i has at most $|T_i| - 1$ edges, since it is a forest. Hence, the number of vertices of T_i that have degree at least 3 is at most $(2|T_i| - 1)/3$. Hence, at least $1/3$ of the nodes of T_i are put in L_i. This means in each iteration the number of nodes reduces by a $2/3$-factor, which implies that we are done within $\ell = \log_{3/2} n$ iterations. □

The Algorithm for Coloring Unrooted Trees, Step 2 Now, we color each of the subgraphs $T[L_i]$ independently using 3 colors, in $O(\log^* n)$ rounds. Notice that this is doable for instance using the algorithm that we saw in the 5^{th} lecture, because $T[L_i]$ has maximum degree at most 2. We use these colors of the graphs $T[L_i]$ mainly as a *schedule-color*, for computing the final output coloring of the vertices.

The Algorithm for Coloring Unrooted Trees, Step 3 We process the graph by going through the layers L_ℓ to L_1, spending 3 rounds on each. Each time, we make sure that we have a valid *final-coloring* of the graph $T[\cup_{j=i}^{\ell} L_i]$ with 3 colors, for decreasing value of i.

Suppose we have an arbitrary final-coloring of $T[\cup_{j=i+1}^{\ell} L_j]$ already, with 3 colors. How do we compute a 3-coloring for vertices of L_i in a manner that does not create a violation with the colors of vertices of $T[\cup_{j=i+1}^{\ell} L_j]$?

This can be done easily using our usual trick of applying one coloring as a schedule for computing another coloring. In particular, we will solve this part of the problem in 3 rounds. We go through the 3 schedule-colors $q \in \{1, 2, 3\}$ of $T[L_i]$, one by one, each time picking a *final-color* in $\{1, 2, 3\}$ for all the vertices in L_i with schedule color q.

References

[Bol78] Béla Bollobás. Chromatic number, girth and maximal degree. *Discrete Mathematics*, 24(3):311–314, 1978.

[CFS10] David Conlon, Jacob Fox, and Benny Sudakov. Hypergraph ramsey numbers. *Journal of the American Mathematical Society*, 23(1):247–266, 2010.

[Jam11] Mohammad Shoaib Jamall. A brooks' theorem for triangle-free graphs. *arXiv preprint arXiv:1106.1958*, 2011.

[Kim95] Jeong Han Kim. On brooks' theorem for sparse graphs. *Combinatorics, Probability and Computing*, 4(02):97–132, 1995.

[Lin87] Nathan Linial. Distributive graph algorithms global solutions from local data. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 331–335. IEEE, 1987.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[LS14] Juhana Laurinharju and Jukka Suomela. Brief announcement: Linial's lower bound made easy. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 377–378. ACM, 2014.

[Nao91] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.

[PS15] Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015.