Principles of Distributed Computing	03/27, 2019
Exercise 6	
Lecturer: Mohsen Ghaffari	

Network Decompositions

Exercise 1: Explain how given a $(\mathcal{C}, \mathcal{D})$ network decomposition of graph G, we can deterministically compute a $(\Delta + 1)$-coloring of the graph in $O(\mathcal{CD})$ rounds. Here, Δ denotes an upper bound on the maximum degree of the graph, and is given to the algorithm as an input.

Solution: We will color graphs G_1, G_2, ..., $G_{\mathcal{C}}$ one by one, each time considering the coloring assigned to the previous subgraphs. Suppose that vertices of graphs G_1, G_2, ..., G_i are already colored using colors in $\{1, 2, \ldots, \Delta + 1\}$. We explain how to color G_{i+1} in $O(\mathcal{D})$ rounds. Consider the clusters X_1, X_2, ..., X_ℓ of G_{i+1} and notice their two properties: (1) they are mutually non-adjacent, (2) for each cluster X_j, its vertices are within distance $\mathcal{D}$ of each other (where distances are according to the base graph G). For each cluster X_j, let node $v_j \in X_j$ who has the maximum identifier among nodes of X_j be the leader of X_j. Then, let v_j aggregate the topology of the subgraph induced by X_j as well as the colors assigned to nodes adjacent to X_j in the previous graphs G_1, G_2, ..., G_i. This again can be done in $O(\mathcal{D})$ rounds, thanks to the fact that all the relevant information is within distance $\mathcal{D} + 1$ of v_j. Once this information is gathered, node v_j can compute a $(\Delta + 1)$-coloring for vertices of X_j, while taking into account the colors of neighboring nodes of previous graphs, using a simple greedy procedure. Then, node v_j can report back these colors to nodes of X_j. This will happen for all the clusters X_1, X_2, ..., X_ℓ in parallel, thanks to the fact that they are non-adjacent and thus, their coloring choices does not interfere with each other.

Exercise 2: In this exercise, we prove that every n-node graph G has an $(\mathcal{C}, \mathcal{D})$ (strong-diameter) network decomposition for $\mathcal{C} = O(\log n)$ and $\mathcal{D} = O(\log n)$. The process that we see that be viewed as a simple and efficient sequential algorithm for computing such a network decomposition.

We determine the blocks G_1, G_2, ..., $G_{\mathcal{C}}$ of network decomposition one by one, in C phases. Consider phase i and the graph $G \setminus (\cup_{j=1}^{i-1} G_j)$ remaining after the first $i-1$ phases which defined the first i blocks G_1, ..., G_{i-1}. To define the next block, we repeatedly perform a ball carving starting from arbitrary nodes, until all nodes of $G \setminus (\cup_{j=1}^{i-1} G_j)$ are removed. This ball carving process works as follows: consider an arbitrary node $v \in G \setminus (\cup_{j=1}^{i} G_j)$ and consider gradually growing a ball around v, hop by hop. In the k^{th} step, the ball $B_k(v)$ is simply the set all nodes within distance k of v in the remaining graph. In the very first step that the ball does not grow by more than a 2 factor — i.e., smallest value of k for which $|B_{k+1}(v)|/|B_k(v)| \leq 2$ — we stop the ball growing. Then, we carve out the inside of this ball — i.e., all nodes in $B_k(v)$ — and define them to be a cluster of G_i. Hence, these nodes are added to G_i. Moreover, we remove all boundary nodes of this ball —i.e., those of $B_{k+1}(v) \setminus B_k(v)$—and from the graph considered for the rest of this phase. These nodes will never be put in G_i. We will bring them back in the next phases, so that they get clustered in the future phases. Then, we repeat a similar ball carving starting at an arbitrary other node v' in the remaining graph. We continue a similar ball carving until all nodes are removed. This finishes the description of phase i. Once no node remains in this graph, we move to the next phase. The algorithm terminates once all nodes have been clustered.

Prove the following properties:

1. Each cluster defined in the above process has diameter at most $O(\log n)$. In particular, for each ball that we carve, the related radius k is at most $O(\log n)$.

   Solution: We show that the ball carving finishes in $\lceil \log n \rceil$ steps, which implies that the radius is at most $\lceil \log n \rceil$ as well. First, note that whenever we do not stop the ball growing, the size of a ball doubles, as $|B_{k+1}(v)] \geq 2 \cdot |B_k(v)|$. Thus, if we did not stop the ball growing within k steps, the ball $B_k(v)$ has size $|B_k(v)| \geq 2^k$. After $k \geq \lceil \log n \rceil + 1$ steps, this would mean that $B_k(v)$ contained at least $2^k = 2^{\lceil \log n \rceil + 1} > n$ nodes, a contradiction.

2. In each phase i, the number of nodes that we cluster —and thus put in G_i — is at least $1/2$ of the nodes of $G \setminus (\cup_{j=1}^{i} G_j)$.

   Solution: Note that every node is either clustered or not, thus we show that the number of nodes included in G_i is at least as large as the number of nodes that are not clustered. Let us focus on a cluster created by a vertex v, which has radius k. By the stopping condition, $|B_{k+1}(v)|/|B_k(v)| \leq 2$ must hold. This implies $|B_k(v)| \geq 1/2|B_{k+1}(v)|$ or that at least $1/2$ of the nodes removed by this step are included in G_i. As is this is true for any ball, it proves the desired statement.

3. Conclude that the process terminates in at most $O(\log n)$ phases, which means that the network decomposition has at most $O(\log n)$ blocks.

   Solution: In every step we remove at least $1/2$ of the remaining nodes. Thus after building G_i at most $n/2^i$ vertices remain. After $\lceil \log n \rceil$ phases this means that at most $n/2^{\lceil \log n \rceil} = 1$ vertex remains which will trivially form the last cluster.

Exercise 3 (*optional*): Develop a deterministic distributed algorithm with round complexity $2^{O(\sqrt{\log n \cdot \log \log n})}$ for computing an $(\mathcal{C}, \mathcal{D})$ (strong-diameter) network decomposition in any n-node network, such that $\mathcal{C} = O(\log n)$ and $\mathcal{D} = O(\log n)$.