University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs,
Z. Parsaeian, G. Schmid

Theory of Distributed Systems
Sample Solution Exercise Sheet 10

Due: Wednesday, 12th of July 2023, 12:00 noon

Exercise 1: Gather everything in CONGEST *(5 Points)*

We are given a connected graph $G = (V, E)$ and we want each node to know the whole graph. We have seen during the lecture that such problem can be solved in $O(diam(G))$ rounds in the LOCAL model. Show that it is possible to solve this problem in $O(|E|)$ in the CONGEST model.

Sample Solution

Algorithm: We use a similar approach as in the wave algorithm for APSP.

1. First we construct a BFS tree (with the "proposal + accept" procedure seen in the lecture).
2. Then we move a token along the BFS tree.
3. For each node that is visited by the token do:
   - 3.1 The node broadcasts its incident edges (in G!) in a "wave" over a tree, one edge per wave.
   - 3.2 After all edges are sent we wait one more round.
   - 3.3 Move the token to the next node and repeat (until the token visited all nodes once)

Running time: It takes $\mathcal{O}(diam(G))$ to construct the BFS tree. Note that we can encode each edge with $\mathcal{O}(\log n)$ bits, thus it is possible to send edges 'step-by-step' to neighbors in CONGEST (while it is not possible to send all incident edges in one round). As the token that traverses the tree will generate at most $\mathcal{O}(|E|)$ waves (at most 2 waves per edge) and since we create a wave at least every other round, it takes at most $\mathcal{O}(|E|)$ rounds until all waves are started. Finally it takes at most $\mathcal{O}(diam(G))$ until the last wave is received by all nodes. So overall it takes $\mathcal{O}(|E| + diam(G))$ rounds. Since the graph is connected we have $\mathcal{O}(diam(G)) \subseteq \mathcal{O}(|E|)$.

Exercise 2: APSP – Slow token *(5 Points)*

During the lecture we have seen two animations of the APSP algorithm:

- One where the token is moved slowly (every 2 rounds), and there are no waves that collide.
- One where the token is moved fast (every round), and many waves collide.

Prove that it is indeed true that, if we move the token every two rounds, each node at each round has to propagate at most one wave.

Sample Solution

Suppose, for a contradiction, that there exists a node v that in a certain round obtains messages from at least 2 waves, W_1 and W_2. Let w_1 and w_2 be the nodes that started the waves W_1 and W_2, respectively. Let $x := d(v, w_1)$ and $y := d(v, w_2)$. If $x = y$, then w_1, w_2 would be at the same distance from v, thus they have started the wave at the same time. But this is a contradiction since only one node (the one that holds the unique token) can start a wave in a given round.

So consider the case $x > y$ (w.l.o.g.). Since W_1 and W_2 (supposedly) hit v at the same time, the difference between travel time of W_1 and W_2, i.e., the time difference when W_1 and W_2 have been started is $x - y$ rounds. So $x - y$ corresponds to the number of rounds the token takes travelling from w_1 to w_2. Thus the distance between nodes w_1 and w_2 can be at most $(x - y)/2$ since we move the token at most every two rounds. By triangle inequality we obtain the following contradiction

$$x = d(v, w_1) \leq d(v, w_2) + d(w_2, w_1) \leq y + \frac{x - y}{2} = \frac{x + y}{2} < x.$$

Exercise 3: Bipartite Graph Detection *(5 Points)*

In this exercise we say that in order to detect if the graph $G = (V, E)$ has a certain property in the CONGEST model, all nodes have to output 1 if the graph fulfills that property, otherwise all nodes have to output 0. A graph is called bipartite if its nodes can be partitioned into two sets, i.e., $V = A \cup B, A \cap B = \emptyset$, such that for all edges $\{u, v\} \in E$ we have $u \in A$ and $v \in B$ (or vice versa). Show that it is possible to detect if the graph is *bipartite* or not in $\mathcal{O}(\text{diam}(G))$ rounds.

Sample Solution

The idea is to use the fact that any bipartite graph is 2-colorable, and any graph that is 2-colorable is a bipartite graph. We start by electing a leader and constructing a BFS tree rooted at the leader in $O(\text{diam}(G))$ rounds. Now we 2-color the BFS tree in the following way: leader picks color 1 and informs its children. Then each other node, once it receives the color of the parent, they pick the color that is different from the parent's one, and if they have children they forward their color to the children. In $O(\text{diam}(G))$ rounds we get a 2-colored BFS tree—notice that a 2-coloring of G implies a 2-coloring of the BFS tree, and 2-coloring of the BFS tree implies a 2-coloring of G if G is bipartite.

So now what is left is to check whether the two coloring is actually a 2-coloring of the graph. For this, nodes spend 1 round and exchange their color with the neighbors. If a node notices a conflict on the coloring, it will set a flag variable $b = 0$, otherwise $b = 1$. Now, as in the previous exercises, we aggregate these bit flags from leaves towards the root, where each non-leaf node will compute the logical "and" from its flag and the ones received from its children. In $O(\text{diam}(G))$ rounds the root will know the result of the "and" aggregation of all bits. If and only if the result of the bit aggregation at the root is 1 the graph has no conflicts and is in fact bipartite.

Exercise 4: Diameter Lower Bound – Refinement *(5 Points)*

In the lecture we have seen that computing the diameter $\text{diam}(G)$ of G in the CONGEST model takes $\Omega(n/\log n)$ rounds in general. Argue that the same lower bound holds even for computing an approximate value $\tilde{D}$ with $\frac{4}{5} \cdot \text{diam}(G) < \tilde{D} \leq \text{diam}(G)$.

Sample Solution

We have seen in the lecture that there is a worst case graph in which it takes $\Omega(n/\log n)$ rounds to decide whether it has diameter 4 or 5. For a contradiction, assume there would be an algorithm A that computes an approximate value $\tilde{D}$ with $\frac{4}{5} \cdot \text{diam}(G) < \tilde{D} \leq \text{diam}(G)$ for any graph G in time $o(n/\log n)$.

We apply this algorithm A on the worst case graph given in the lecture (we assume that this is G now) to compute $\tilde{D}$ with $\frac{4}{5} \cdot \text{diam}(G) < \tilde{D} \leq \text{diam}(G)$. If $\text{diam}(G) = 4$ then $\tilde{D} \leq 4$. If $\text{diam}(G) = 5$ then $\tilde{D} > \frac{4}{5} \cdot \text{diam}(G) = 4$.
So by running algorithm A on G and afterwards outputting "$\text{diam}(G) = 4$" if $\tilde{D} \leq 4$, and "$\text{diam}(G) = 5$" if $\tilde{D} > 4$, in fact solves the original problem of distinguishing if G has diameter 4 or 5 in time $o(n/\log n)$, a contradiction.