ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2012

Prof. R. Wattenhofer
Stephan Holzer

Principles of Distributed Computing
Exercise 11: Sample Solution

1 Communication Complexity of Set Disjointness

a) We obtain

$$M^{DISJ} = \left(\begin{array}{c|cccccccc|l} \text{DISJ} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 & \leftarrow x \\ \hline 000 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ 001 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \\ 010 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \\ 011 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\ 100 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 101 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \\ 110 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 111 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \uparrow y & & & & & & & & & \end{array}\right).$$

b) When $k = 3$, a fooling set of size 4 for *DISJ* is e.g.

$$S_1 := \{(000, 111), (001, 110), (010, 101), (100, 001)\}.$$

Entries in M^{DISJ} corresponding to elements of S_1 are marked dark gray. However, a fooling set does not always need to be on a diagonal of the matrix. An example for such a set is

$$S_2 := \{(001, 110), (010, 001), (011, 100), (100, 010)\},$$

and marked light gray in M^{DISJ}.

c) In general, $S := \{(x, \overline{x}) \mid x \in \{0,1\}^k\}$ is a fooling set for *DISJ*. To prove this, we note: If $y > \overline{x}$ then there is always an index i such that $x_i = y_i = 1$ and we conclude $DISJ(x, y) = 0$. Second, we note for any elements $(x_1, y_1), (x_2, y_2)$ of any fooling set that $x_1 \neq x_2$. Otherwise it was $(x_1, y_j) = (x_2, y_j)$ for $j \in \{1, 2\}$ and thus $f(x_2, y_1) = f(x_1, y_2) = f(x_1, y_1) = f(x_2, y_2) =: z$ which contradicts the definition of a fooling set. Similarly it is $y_1 \neq y_2$.

- For any $(x, y) \in S$ it is $DISJ(x, y) = 1$.
- Now consider any $(x_1, y_1) \neq (x_2, y_2) \in S$. Since $x_1 \neq x_2$ and $y_1 \neq y_2$, we conclude that either $y_2 > \overline{x_1}$, in which case $DISJ(x_1, y_2) = 0$, or $y_1 > \overline{x_2}$ causing $DISJ(x_2, y_1) = 0$.

2 Distinguishing Diameter 2 from 4

a)
- Choosing $v \in L$ takes $O(D)$: Use any leader election protocol from the lecture. E.g. the node with smallest ID in L can be elected as a leader. Then this node will be v.
- Computing a BFS tree from a vertex usually takes $O(D)$. Since in our setting all graphs are guaranteed to have constant diameter, the time required for this is $O(1)$. As node v is in L, at most $N_1(v) < s$ executions of BFS are performed. These can be started one after each other and yield a complexity of $O(s)$.
- The comment states: Computing an H-dominating set $\mathcal{DOM}$ takes time $O(D) = O(1)$.
- Since $|\mathcal{DOM}| \leq \frac{n \log n}{s}$, the time complexity of computing all BFS trees from each vertex in $\mathcal{DOM}$ (one after each other) is $O(\frac{n \log n}{s})$.
- Checking whether all trees have depth of at most 2 can be done in $O(D) = O(1)$ as well: Each node knows its depth in any of the computed trees. If its depth is 3 or 4, it floods "diameter is 4" to the graph. If a node gets such a message from several neighbors, it only forwards it to those from which it did not receive it yet. If any node did not receive message "diameter is 4" after 4 rounds, it decides that the diameter is 2. Otherwise it decides that the diameter is 4. This decision will be consistent among all nodes.
- By adding all these runtimes, we conclude that the total time complexity of Algorithm 2-vs-4 is $O\left(s + \frac{n \log n}{s}\right)$.

b) By deriving $O\left(s + \frac{n \log n}{s}\right)$ as a function of s we can argue that $O\left(s + \frac{n \log n}{s}\right)$ is minimal for $s = \sqrt{n \log n}$. Thus the runtime of the Algorithm is $O(\sqrt{n \log n})$.

c) Since in this case no BFS tree can have depth larger than 2 the algorithm returns "diameter is 2".

d) Using the triangle inequality we obtain that $d(w, v) \geq d(u, v) - d(w, v) = 3$ thus the BFS tree of w has at least depth 3. Therefore Algorithm 2-vs-4 decides "diameter is 4".

e) If there is a $w \in L$ such that a BFS started in w has depth at least 3, we are done. In the other case it is $d(u, w) \leq 2$. Using d) we conclude that $d(u, w) = 2$. Let w' be a node that connects u to w. Since $w' \in N_1(w)$ and $w \in L$, Algorithm 2-vs-4 executes a BFS from w'. Then we apply d) using that $w' \in N_1(u)$.

f) Since $\mathcal{DOM}$ is a dominating set for $H = V \setminus L = V$, it follows immediately that the algorithm executes a BFS from a node $w \in D \cap N(u) \neq \emptyset$. Now apply d).

g) A careful look into the construction of family $\mathcal{G}$ reveals that we essentially showed an $\Omega(n/\log n)$ lower bound to distinguish diameter 2 from 3. Since the graphs considered here cannot have diameter 3, the studied algorithm does not contradict this lower bound.

3 Fast Computation of an H-Dominating Set $\mathcal{DOM}$

Note that the simplified Chernoff Bound suffices for the problems below, but in many other settings the general version is favorable. Also note that most of the statements below could be proven in a not extremely difficult way without using a Chernoff Bound. But using it makes life easier!

a) To take away the fear of Chernoff, we started with a nice warm up. Since all coin tosses are independent, we can use linearity of expectation and obtain $\mathbb{E}[X] = \frac{N}{2}$. Applying the simplified Chernoff Bound ($Pr\left[X \leq \frac{1}{2}\mathbb{E}[X]\right] \leq e^{-\mathbb{E}[X]/8}$) yields $Pr\left[X \leq \frac{N}{4}\right] \leq e^{-N/16}$.

b) Denote by $X^v := \sum_{u \in N(v)} X_u$ the number of marked nodes in the neighborhood of v. Since each node marks itself independently and v has $|N(v)|$ neighbors, we can use linearity of expectation to obtain

$$\mathbb{E}\left[X^v\right] = \sum_{u \in N(v)} \mathbb{E}\left[X_u\right] = \frac{8(c+1) \cdot |N(v)| \cdot \ln n}{s}.$$

Since v is in H, we know that $N(v) \geq s$ and conclude that $\mathbb{E}\left[X^v\right] \geq 8(c+1) \cdot \ln n$.

c)

$$Pr\left[X^v \geq 1\right] \geq 1 - Pr\left[X^v \leq 4(c+1) \cdot \ln n\right] \geq 1 - e^{-\mathbb{E}[X^v]/8} \geq 1 - e^{-(c+1)\cdot \ln n} = 1 - n^{-(c+1)}.$$

d) The probability that each out of $|H| \leq n$ nodes has a neighbor that is in H is at least $\left(1 - \frac{1}{n^{c+1}}\right)^n$. Using the fact that $(1 + x/n)^n \geq e^x$ we obtain that this is less than $e^{-n^{-c}}$. Using the fact that $e^x \geq 1 + x$ we conclude that this probability is higher than $1 - n^{-c}$. This is w.h.p..

e) Denote by $X := \sum_{u \in V} X_u$ the number of nodes in S. Then the expected number is $\mathbb{E}[X] = 8(c+1)\sqrt{n \ln n}$. Applying the simplified Chernoff Bound yields

$$Pr\left[|S| \geq 4(c+1)\frac{n \ln n}{s}\right] \geq 1 - e^{-(c+1)\cdot\sqrt{n \ln n}} = 1 - 2^{-\Omega(\sqrt{n \ln n})}.$$

f) No round of communication is necessary! Each node can internally decide whether to join S or not. This is in some sense better than what we assumed during the analysis of Algorithm 2-vs-4. However, it is not always realistic that all nodes know s and n and start at the same time – this needs some preprocessing!