ASSIGNMENT 3 SOLUTIONS - DISTRIBUTED ALGORITHMS

(1) Consider a graph $G = (V, E)$, in which $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{D-1}, v_D)\} \bigcup \{(v_D, v_i) | i = D+1, D+2, \ldots, n\}$.

In other words, the graph is a sequence of D vertices, which on of its end vertices is connected to $n - D$ vertices. (I would have placed a figure have I figured out how to do it in LaTeX ;))

This graph holds: $|V| = n$, $|E| = \Theta(n)$ (we'll solve sections (a) and (b) together) and $Diam(G) = D$. Let's analyze the message complexity when invoking *Dijkstra*'s algorithm from v_D:

- The first phase sends $deg(v_D) = n - D + 1$ DISCOVER messages, and receives the same amount of replys. The spanning tree $\Gamma_1(G)$ contains $n - D + 2$ vertices and $n - D + 1$ edges.
- Each additional phase will discover only one new vertex. So $|V(\Gamma_i(G))| = n - D + i$ edges for $i \in \{2, \ldots, D\}$. The phase i uses $n - D + i$ broadcast and convergecast messages, one DISCOVER message and one reply for it.

Since this algorithm runs for exactly D phases, the total message complexity is $\sum_{i=1}^{D}(n - D + i + \Theta(1)) = \Theta(n \cdot D + |E|)$.

Note that this is not the case for every graph and each execution of *Dijkstra*! For example, take the graph described above, with $D = \Theta(\sqrt{n})$, and invoke *Dijkstra*'s algorithm from v_1. The resulting message complexity will be only $\Theta(n + |E|)$.

(2) Consider the graph $G = K_n$: a clique of n vertices $v_1, v_2, \ldots v_n$. Invoke *Bellman-Ford* from v_1 and consider the following scenrio:

- v_1 sends a message "distance=1" to each $v \in V$ except itself.
- v_2 receives "distance=1" from v_1, updates its distance variable to 1 and sends "distance=2" to each $v \in V$ (except itself and maybe v_1).
- v_3 receives "distance=2" from v_2, updates its distance variable to 2 and sends "distance=3" to each $v \in V$ (except itself and maybe v_2).
- $\cdots$
- v_n receives "distance=$n-1$" from v_{n-1}, updates its distance variable to $n-1$ and sends "distance=n" to each $v \in V$ (except itself and maybe v_{n-1}).
- v_3 now receives the "distance=1" message from v_1. It updates its distance variable to 1 and sends "distance=2" to each $v \in V$ (except itself and maybe v_1).
- v_4 receives "distance=2" from v_3, updates its distance variable to 2 and sends "distance=3" to each $v \in V$ (except itself and maybe v_3).
- $\cdots$
- and so on, until for each $v \in V$ the distance variable equals 1

Each vertex sends $n-2$ messages on each step; a vertex v_i starts a sequence of $n-i$ such steps. So the total amount of messages is: $\sum_{i=1}^{n}((n-2)\cdot(n-i)) = \Theta(n^3)$.

Note that this is not the case for every graph and each execution of *Bellman-Ford*; for example, if the "distance=1" messages would have been received by all the vertices approximatly at the same time, the message complexity of this execution would have been only $\Theta(n^2)$.

(3) We'll describe a β-like synchronizer; this synchronizer will use the existing trees $T \in \mathcal{T}$ instead of creating a spanning tree for the graph G. Since the synchronization problem can be reduced to the neighbours updating problem, we'll describe a solution to the latter.

We will perform a convergecast of the b_v bits of each $v \in V$ using all of the trees $T \in \mathcal{T}$, then broadcast all_v messages on them. Each vertex $v \in V$ will toggle its all_v bit only when it gets $Overlap_v(\mathcal{T})$ all_v messages (i.e. it gets a message from *all* of the trees it belongs to).

(Note that otherwise a vertex v might update its all_v bit before all of his neighbours w have updated their b_w bits!)

- $Time_{init} = Comm_{init} = 0$, since the trees are already given and no further preparations are needed.
- $Time_{pulse} = O(d)$, since this is the maximal diamater of a tree $T \in \mathcal{T}$ - and hence the maximal time needed for a pulse counter to change in the entire network.
- $Comm_{pulse} = O(l \cdot n)$: a vertex $v \in V$ sends $\Theta(Overlap_v(\mathcal{T}))$ messages on each pulse (Note that this number is not bounded by $|E|$ - an edge may be used more than once). Since there are n vertices with a maximal Overlap of l, the total amount of messages sent is $O(l \cdot n)$.
- $Time_{gap} \leq PulseDiff \cdot Time_{pulse} = 1 \cdot O(d) = O(d)$.

(4) There are two common approaches to this problem. We'll describe them both:

(a) Use an α-like synchronizer, using the edges in U to pass messages. The problem here is that $(u,v) \in E$ does not imply that $(u,v) \in U$; say in other words, two neighbouring vertices in G may be up to k edges away in G'. The implication is that when a vertex v sends a message to its neighbours in G', this message is received by vertices in G that are up to k edges away from v's immediate neighbours.
Therefore, if we use the α synchronizer as-is, a vertex v might update its all_v bit before all of his neighbours w have updated their b_w bits.
To solve this, each vertex $v \in V$ will hold a counter c_v which is initialized to 0. When its b_v bit updates, it sends a message to its neighbours in G'; when a vertex $u \in V$ gets such message it increments its counter c_u by one and, if $c_u < k$, sends this message to its neighbours in G'.

When $c_v = k$, the vertex v knows that each of its original neighbours in G have their b bits on.

- $Time_{init} = \Theta(1)$: define the c_v counter for each vertex and initialize it.
- $Comm_{init} = 0$
- $Time_{pulse} = O(k)$, since each vertex sends k messages to its neighbours in G'; each of these messages requires $O(1)$ time steps.
- $Comm_{pulse} = O(h \cdot k)$, since each vertex v sends k messages to its $deg(v)$ neighbours in G' ($\sum_{v \in V} deg(v) = \Theta(|U|)$).
- $Time_{gap} \leq PulseDiff \cdot Time_{pulse} = O(k) \cdot O(k) = O(k^2)$. An explanation is needed here: two neighbouring vertices in G may be up to k edges away in G', therefore thet pulse difference between them can be $O(k)$.

(b) Use a β-like synchronizer using G'. This method requires finding a spanning tree T of G' before the execution.

- $Time_{init}$ = the time complexity of the spanning tree algorithm.
- $Comm_{init}$ = the communication complexity of the spanning tree algorithm.
- $Time_{pulse} = Depth(T)$. If T is a MST, then $Time_{pulse} = O(Diam(G'))$.
- $Comm_{pulse} = O(n)$, since the messages pass on the spanning tree edges.
- $Time_{gap} \leq PulseDiff \cdot Time_{pulse} = O(1) \cdot Depth(T)$. Again, if T is a MST then $Time_{gap} = O(Diam(G'))$.

(5) (a) In the α synchronizer $|P_v - P_u| \leq 1$ for $(u, v) \in E$, hence $|P_v - P_u| \leq dist_G(u, v)|$ for any $u, v \in V$.
In this case, $|P_{v'} - P_v| = |P_{v'} - 27| \leq dist(v_{11}, v_2) = 6$. Hence, $p_{v'} \in \{21, 22, \ldots, 33\}$.

(b) In the β synchronizer $|P_v - P_u| = \Theta(1)$ for any $u, v \in V$, but note that a vertex v will increment its pulse number before the vertices in its rooted subtree.
Therefore, if v is the tree root, v' is a leaf and $p_v = 27$, there are two possibilities:

- $p_v = p_{v'} = 27$.
- $p_v = 27$, and v has issued a broadcast for the new pulse number, yet v' have not received it yet; hence $p_{v'} = 26$.

To sum it up: $p_{v'} \in \{26, 27\}$.

(c) In this case, when both v and v' are leaves, there are three possibilities:

- $p_v = p_{v'} = 27$.
- $p_{rt} = 27$, and rt has issued a broadcast for the new pulse number. v have got this message, so $p_v = 27$, yet v' have not received it yet, hence $p_{v'} = 26$.
- $p_{rt} = 28$, and rt has issued a broadcast for the new pulse number. v haven't received this message yet, so $p_v = 27$, but v' have received it, hence $p_{v'} = 28$.

To sum it up: $p_{v'} \in \{26, 27, 28\}$.