TD 12 : Path and cycle 3-coloring, spanning tree, graph size

1 Path 3-coloring

We first recall the definition of an oriented path.

Definition (Oriented path)**.** In an oriented path, each process has at most one predecessor and at most one successor. Messages can be sent by processes in both directions.

Problem 1.1. In the $O(\log^* n)$ algorithm (which we shall refer to as the Cole-Vishkin algorithm) done during the class, it was shown how the number of colors reduce from m to $2\lfloor \log_2 m \rfloor + 1$ for $m \geq 5$, thus eventually reducing the number of colors to 6 (assuming that the final colors are $0, \ldots, 5$). Show how to further reduce this number to 3. Do the processes need to know the number of nodes in the graph beforehand ?

Solution. The processes have to know the number of nodes n beforehand. They can carry out the m to $2\log_2 m + 1$ range reduction for $\mathcal{O}(\log^* n)$ rounds, knowing that in the end, the range of colors is in $\{0, \ldots, 5\}$. They can then run the simple 3-coloring algorithm, where in every round, processes whose values are the local maxima adopt a value in $\{0, 1, 2\}$. It takes at most 3 rounds for all processes to have colors in $\{0, 1, 2\}$. □

† **Problem 1.2.** The Cole-Vishkin algorithm done during the class works with oriented paths. Give an algorithm that is almost as fast and works even on non-oriented paths. You can assume that everyone has unique identifiers.

Solution. The idea is to break up the path into oriented paths. In the first round, all processes exchange their values with their neighbors and determine whether their value is a local maximum, local minimum, or neither. Processes with local maxima orient their edges outwards and those with local minima orient their edges inwards. For the rest, they orient one edge towards the neighbor having a lower value and the other edge away from the neighbor having a higher value.

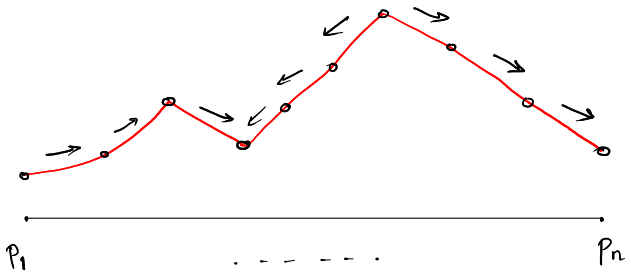

FIGURE 1 – Breaking path into oriented paths

Having done this in the first round, it is easy to see that processes can follow the Cole-Vishkin algorithm for paths, except for those who have a local minima as they have values coming in from both neighbors. But this is easy to resolve : instead of following the Cole-Vishkin algorithm, they choose the lowest available value in $\{0, 1, 2\}$ after its neighbors have decided. Correctness at all other points follows from the correctness of the Cole-Vishkin algorithm. □

† **Problem 1.3** (Randomized and fast)**.** The simple randomized 3-coloring algorithm done during the class finds a 3-coloring in time $\mathcal{O}(\log n)$ with high probability, and it does *not* need any unique identifiers. Give a fast randomized algorithm that does this in $o(\log n)$ time. You can assume that the processes know the number of nodes n.

Solution. The idea is to first randomly choose an identifier from a large set, hope that it is locally unique (else try try try again) and then run the Cole-Vishkin algorithm. In every round, every process chooses a value uniformly at random from $\{0, \ldots, 2n-1\}$, then shares it with its neighbors, adopts it if it is different from their values, and lets them know in the next round if the value was adopted. Once a process knows that the neighbor to its right (we are assuming that the path is oriented from right to left) has also adopted a value, it runs the Cole-Vishkin algorithm. If the neighbor to the right is not able to compute a value for some round of the Cole-Vishkin algorithm, the process waits until the neighbor computes a value and is then able to continue with the Cole-Vishkin algorithm. Correctness then follows from the correctness of the Cole-Vishkin algorithm. We claim that all processes finish within constant time with high probability. If a process has not yet decided at the start of some round, the probability that its chosen value is different from that of its neighbors is at least $\frac{2n-2}{2n}$; indeed, for any choice values by the other $n-1$ processes, there exist $n-2$ available values that work. So the probability that the process has not yet decided by round α is $1/n^{\alpha}$. Thus

$$\mathbb{P}\left(\text{some process decides after round } \alpha\right) \leq \frac{1}{n^{\alpha-1}},$$

showing that all processes decide in $\mathcal{O}(1)$ number of rounds with high probability. □

†† **Problem 1.4** (Oblivious algorithm)**.** Show that the Cole-Vishkin algorithm can be adapted to work even if the processes do not know the number of nodes in the path beforehand. You may assume that the path is oriented.

Solution. Processes fix an initial constant value d.

- In the first phase, only processes with UID at most d run the Cole-Vishkin algorithm. It is known by all processes that they will decide within $\mathcal{O}(\log^* d) + \mathcal{O}(1)$ rounds. The processes then halt at the end of the phase.
- In the second phase, only processes with UID between d and 2^d run the Cole-Vishkin algorithm and halt at the end of the phase.
- For $i > 2$, in the i-th phase, only processes with UID between $2^{\cdot^{\cdot^{\cdot^{2^d}}}}$ with tower height (i.e. number of 2s in the tower) 2^{i-2} and $2^{2^{\cdot^{\cdot^{\cdot^{2^d}}}}}$ with tower height 2^{i-1} run Cole-Vishkin.
- This keeps going on till some phase k at the end of which all processes have decided.

Analysis : If we assume that the largest UID value is bounded by n^C for some constant C, then $2^{2^{\cdot^{\cdot^{\cdot^{2^d}}}}} < n^C$ where 2^{k-2} is the height of the tower. Thus $2^{k-2} = \mathcal{O}(\log^* n)$. Since the i-th phase takes $2^i + \log^* d$ rounds, adding up from $i = 1$ to $i = k$ gives that the round complexity is $(1 + 2 + \cdots + 2^{k-2}) + k \log^* d = \mathcal{O}(\log^* n)$. □

2 Cycle 3-coloring

Assume that there are n processes that form an oriented cycle C_n (as for paths, every process has at most one predecessor and at most one successor).

Problem 2.1. Show that the randomized and Cole-Vishkin algorithms used for 3-coloring paths can also be used for 3-coloring cycles.

Solution. This should be straightforward to check. □

† **Problem 2.2.** You may assume that the Cole-Vishkin algorithm actually has $\log^* n + \mathcal{O}(1)$ round complexity. Show that there is in fact an algorithm with $\frac{1}{2}\log^* n + \mathcal{O}(1)$ round complexity.

Solution. The processes can simulate 2 rounds of the Cole-Vishkin algorithm (except the part where 6 colors are reduced to 3) in one round as follows. In a round r of the Cole-Vishkin algorithm, let $c_0(u), c_{-1}(u), c_1(u)$ be the values of process u and its left and right neighbors respectively. Let $c'_0(u), c'_{-1}(u), c'_1(u)$ be their new values. Value $c'_0(u)$ is calculated using $c_0(u)$ and $c_1(u)$, and value $c'_{-1}(u)$ is calculated using $c_{-1}(u)$ and $c_0(u)$. Since u receives its left neighbors message, it knows $c_{-1}(u)$ and thus the value $c'_{-1}(u)$. Then u can also calculate the value its left neighbor will calculate in round $r+1$ by using $c'_{-1}(u)$ and $c'_0(u)$. Say this value is $c''_{-1}(u)$. Then u adopts the value $c''_{-1}(u)$ in round r. Thus a value that should have been adopted at the end of round $r+1$ has already been adopted at the end of round r. □

††† **Problem 2.3** (Nonexistence of constant-time algorithm)**.** For all $t \geq 0$, every t-round algorithm fails to 3-color some cycle. *Hint :* One proof of this relies on Ramsey's theorem on hypergraphs given below. Determine an appropriate choice of $k, c, n_1, \ldots, n_c$ in order to prove the lower bound.

Theorem (Ramsey)**.** *For any integers k and c and any integers $n_1, \ldots, n_c$, there is an integer $R = R(n_1, \ldots, n_c; k)$ such that if every size-k subset of a set X where $|X| = R$ is colored with c different colors, then there exists $i \in [c]$ and a subset $X' \subseteq X$ where $|X'| = n_i$ such that all size-k subsets of X' are colored i.*

Solution. Suppose for the sake of contradiction that there is a constant time algorithm $\mathcal{A}$ running in at most t rounds on any cycle C_n. We can then look at $\mathcal{A}$ as a function from the set of $(2t+1)$-tuples $(x_{-t}, \ldots, x_0, \ldots, x_t)$ where all x_i are distinct to the set $\{1, 2, 3\}$. We choose $c = 3, k = 2t+1$ and $n_1 = n_2 = n_3 = 2t+2$.

Consider the set $X = \{0, \ldots, R-1\}$ where $R = R(n_1, n_2, n_3; 2t+1)$ and a coloring of the $(2t+1)$-subsets such that the subset $\{a_0, \ldots, a_{2t+1}\}$ is colored with $\mathcal{A}(a'_0, \ldots, a'_{2t+1})$ where $a'_0 < \cdots < a'_{2t+1}$ and $\{a_0, \ldots, a_{2t+1}\} = \{a'_0, \ldots, a'_{2t+1}\}$; we sort the a_i in order to have a consistent way to color all the subsets.

By Ramsey's theorem, there exists a subset $X' \subseteq X$ where $|X| = 2t+2$ and a color $i \in \{1, 2, 3\}$ such that any $(2t+1)$-size subset of X' is colored i. Let $X' = \{x_{-t}, \ldots, x_0, \ldots, x_t, x_{t+1}\}$ where $x_{-t} < \cdots < x_0 < \cdots < x_t < x_{t+1}$. Consider a cycle C where $x_{-t}, \ldots, x_0, \ldots, x_t, x_{t+1}$ are consecutive values. But in that case, the processes with UIDs x_0 and x_1, which output $\mathcal{A}(x_{-t}, \ldots, x_0, \ldots x_t)$ and $\mathcal{A}(x_{-t+1}, \ldots, x_1, \ldots x_{t+1})$ must actually have the same output i, contradiction. □

3 Graph size and Minimum Spanning Tree

† **Problem 3.1.** Give an algorithm that allows the processes to calculate a spanning tree (i) when there is a designated leader whose ID all processes know, and (ii) when there is no designated leader.

Problem 3.2. Give an algorithm for the processes to calculate the total number of vertices in the graph (i) when there is a designated leader whose ID all processes know, and (ii) when there is no designated leader.