Exam:
Modern Parallel Algorithms

Wydział Mathematyki, Informatyki i Mechaniki, Universytet Warszawski

Due date: January 8, 2023

1. As discussed in the lectures, one can reduce the *writing phase* of CRCW PRAM *simulation* (of a P-processor CRCW PRAM that uses C shared memory cells) on an MPC with local space $\mathfrak{s} = O(N^\delta)$ and global space $\mathfrak{s}_{\text{tot}} = O(N)$, where $N = O(P + C)$, to the following task:

   - The content of each shared memory cell is stored on one of the $O(\frac{C}{\mathfrak{s}})$ MPC machines. Without loss of generality, we can assume that (the value of) cell j is stored on machine $M_{\lceil \frac{j}{\mathfrak{s}} \rceil}$.
   - The input to the simulation is a sequence of P triplets $\langle i, v_i, c_i \rangle$, where $1 \le i \le P$, v_i is a single word, and $1 \le c_i \le C$. (Meaning: the i-th PRAM processor attempts to write value v_i in cell c_i.) Any triplet $\langle i, v_i, c_i \rangle$ is given on machine $M_{\lceil \frac{i}{\mathfrak{s}} \rceil}$.
   - The process should update accordingly all memory cells which the processors want to write into. That is, for every $1 \le j \le C$,
     - ◇ if there is some $1 \le i \le P$ with $c_i = j$, then the simulated PRAM memory cell j stored on the MPC is modified in this round to take value $v_{i'}$, where $1 \le i' \le P$ is an arbitrary index of the PRAM processor with $c_{i'} = j$;
     - ◇ if there is no $1 \le i \le P$ with $c_i = j$, then the simulated PRAM memory cell j stored on the MPC is not modified in this round.

   Let $0 < \delta < 1$ be an arbitrary constant. Design a deterministic algorithm that performs this task in a constant number of rounds on an MPC with local space $\mathfrak{s} = O(N^\delta)$ and global space $\mathfrak{s}_{\text{tot}} = O(N)$.

   (In your solution you can use some other basic primitives discussed in the lectures, like sorting, all-prefix-sum, predecessor, colored summation, etc., all having constant-rounds MPC algorithms.)

   Bonus question: The PRAM simulation above runs for the so-called ARBITRARY CRCW PRAM model (with the conflict resolution protocol for writing to the same shared memory cell: write any of the values attempted to be written). Can you extend it to the $\mathfrak{f}$-CRCW PRAM model, arguably the strongest variant of the PRAM model, where concurrent writes to the same memory location are resolved by applying a *commutative semigroup operator* $\mathfrak{f}$ on all values being written to the same memory address, such as SUM, MIN, MAX, etc? In this case, when we want to update a PRAM memory cell to which one of the processors writes, then the updated value is equal to applying the operation $\mathfrak{f}$ to all values of the processors which want to write to that cell. That is, if there is some $1 \le i \le P$ with $c_i = j$, then the simulated PRAM memory cell j stored on the MPC is modified in this round to take value $\mathfrak{f}(V_j)$, where $V_j = \{v_{i'} : c_{i'} = j\}$. And so, for example, when $\mathfrak{f} \equiv$ SUM, then $\mathfrak{f}(V_j) = \sum_{i':c_{i'}=j} v_{i'}$.

2. In Lecture 3 (slide 3), we sketched a deterministic algorithm that for any constant $0 \le \delta < 1$, sorts N numbers in a constant number of rounds on an MPC with local space $\mathfrak{s} = O(N^\delta)$ and global space $\mathfrak{s}_{\text{tot}} = O(N)$. Your task is to determine the number of MPC rounds of this sorting algorithm as a function of δ. (In particular, the number of MPC rounds *is not* $O(\log_{\mathfrak{s}} N)$.)

3. Consider the **1-bit broadcasting** problem: machine M_1 has a single bit x stored in its local memory, and the goal is to inform all $\mathfrak{m}$ machines in the system about the value of x. Design an MPC algorithm

that in a single round solves the 1-bit broadcasting problem for $\mathfrak{m} = 1 + 2\mathfrak{s}$. (That is, the task is, for a given bit $x \in \{0,1\}$ provided as the input to M_1, in a single MPC round to inform machines $M_2, \ldots, M_{1+2\mathfrak{s}}$ about x.)

(In this question, one should assume that in a single MPC round each machine can send and receive at most $\mathfrak{s}$ *bits* (and not $\mathfrak{s}$ words or $O(\mathfrak{s})$ words, as used frequently throughout the course).)

4. Design an MPC algorithm that takes as its input a simple undirected graph G on n vertices, m edges, and the diameter of each connected component at most D, and determines all connected component of G in $O(D)$ rounds on an MPC with local space $\mathfrak{s} = O(n^\delta)$ and global space $\mathfrak{s}_{\text{tot}} = O(n+m)$.

5. The **1-vs-2-cycle problem** is for a given graph G on n vertices, to determine if G is a single cycle, or consists of two cycles of length $\frac{n}{2}$ (more precisely, of length $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, respectively). This problem is conjectured to be hard for low-space MPC, in that it is conjectured to require $\Omega(\log n)$ rounds.

   Your task is to show that (assuming $D = \omega(\log n)$) the existence of an $o(\log D)$ algorithm Alg for the graph connectivity problem implies an $o(\log n)$-rounds solution to the 1-vs-2-cycle problem. (Notice that this implies that it is *unlikely* to obtain an $o(\log D)$ algorithm Alg for graph connectivity.)

   In what follows, let Alg be a hypothetical algorithm that for a given undirected simple graph G of diameter at most D^*, where $D^* = \omega(\log n)$, determines all connected components of G in $o(\log D^*)$ rounds on an MPC with local space $\mathfrak{s} = O(n^\delta)$ and $\mathfrak{m}$ machines.

   Consider the following algorithm[1]. Given an instance $G = (V, E)$ to the 1-vs-2-cycle problem:

   - if the entire graph G fits a single machine (that is, $|V| + |E| = O(\mathfrak{s})$), then on that single machine determine if G consists of a single cycle or consists of two disjoint cycles;
   - otherwise,
     (i) define graph $H = (V, E')$ obtained by removing each $e \in E$ at random with probability p;
     (ii) find all connected components of H using algorithm Alg;
     (iii) contract each connected component of H to a single vertex in G;
     (iv) recurse on the contracted graph.

   Suppose that we call the recursive algorithm above on a simple undirected graph $G^* := G = (V, E)$ with n vertices and $m = n$ edges, such that the edges of G^* either form a single cycle of length n, or form two disjoint cycles of length $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, respectively. It is easy to see that the algorithm above will maintain that in each recursive call we process a graph G with $N \leq n$ vertices and N edges (unless one of the cycles has been reduced to a single vertex), such that the edges of G either form a single cycle (if G^* was a single cycle), or form two disjoint cycles (if G^* was formed by two cycles). Therefore, at the end of recursion, if the entire graph G fits a single machine then by determining whether G has a single cycle or two cycles allows us to solve the 1-vs-2-cycle problem for the original graph G^*.

   Prove the following claim about a single recursive call to the algorithm above with graph G on N vertices and N edges, $N \leq n$, where each connected component is of size greater than $n^{\delta/2}$:

   assuming n is sufficiently large, if we take (for the use of Alg) $D^* \leq \frac{n^{\delta/2}}{3}$ and $p = \frac{3 \ln n}{D^*}$, then with probability at least $1 - O\left(\frac{1}{n^2}\right)$ the following two conditions hold:

   (i) each connected component in H is of size at most D^* (and thus we can use Alg in step (ii)), and

   (ii) the contracted graph obtained in step (iii) has at most $\frac{100 N \ln n}{D^*}$ edges.

   Hint: for part *(ii)* use Chernoff bound[2].

[1] An appropriate parameter p will be set up later.

[2] For example, the following form is handy: Let $X_1, \cdots, X_n$ be independent 0-1 random variables and let $X := \sum_{i=1}^n X_i$. Then $\mathbf{Pr}[X \geq 2 \cdot \mathbb{E}[X]] \leq e^{-\mathbb{E}[X]/3}$

Then, argue that this claim implies that if there is an algorithm ALG that for any simple undirected graph with n vertices and of maximum diameter at most D (where $D = \omega(\log n)$) determines all connected components in $o(\log D)$-rounds on an MPC with local space $\mathfrak{s} = O(n^\delta)$ using $\mathfrak{m} = \text{poly}(n)$ machines, then the 1-vs-2-cycle conjecture fails (that is, one could solve that 1-vs-2-cycle problem in $o(\log n)$ rounds on an MPC with local space $\mathfrak{s} = O(n^\delta)$ using $\mathfrak{m} = \text{poly}(n)$ machines).

6. In many of the algorithms discussed in the lectures, we have been aiming to design MPC algorithms that solve a problem at hand with input of size N in $O(\log_{\mathfrak{s}} N)$ rounds on an MPC with local space $\mathfrak{s} = O(N^\delta)$ and global space $\mathfrak{s}_{\text{tot}} = O(N)$. In order to argue that the barrier of $O(\log_{\mathfrak{s}} N)$ rounds is often needed, consider the 1-bit broadcasting problem and show that if the number of machines $\mathfrak{m}$ satisfies $\mathfrak{m} \geq (1+2\mathfrak{s})^t$, then any MPC algorithm (with local space $\mathfrak{s}$) for the 1-bit broadcasting problem requires at least t rounds. (In other words, 1-bit broadcasting requires at least $\lceil \log_{1+2\mathfrak{s}} (1+\mathfrak{m}) \rceil$ rounds.)

   (In your analysis do not be concerned about the exact constant 2 in $(1+2\mathfrak{s})$; any positive constant would do. Further, in this question, one should assume that in a single MPC round each machine can send and receive at most $\mathfrak{s}$ *bits* (and not $\mathfrak{s}$ words or $O(\mathfrak{s})$ words, as used frequently throughout the course).)

December 2022

Solutions (and any questions) should be sent before January 8, 2023 to A.Czumaj@warwick.ac.uk.

Any such email should use term Modern Parallel Algorithms in the subject.

Try to solve as many questions as you can. ...
