Chapter 11
Massively Parallel Computations

Part I

Theory of Distributed Systems

Fabian Kuhn

Massively Parallel Computations

Challenges

- Moore's law does not hold for ever
- We can only increase computational power by increasing the parallelism
- We need algorithmic techniques to deal with immense amounts of data

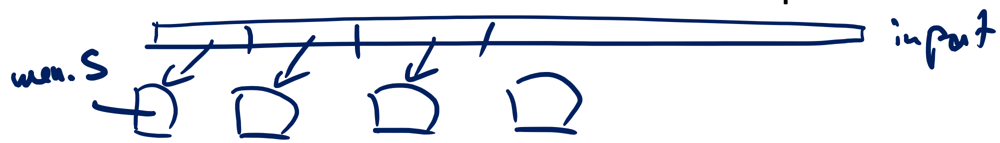
Massively Parallel Graph Computations

- Many important applications require solving standard graph problems in very large graphs (e.g., search engines, shortest path computations, etc.)
- We need ways to perform graph computations in highly parallel settings:
  - Graph data is shared among many servers / machines
  - Each machine can only store a small part of the graph
  - Need techniques to split and parallelize computations among machines
  - Use communication to coordinate between the machines
- Related to (standard) distributed graph computations

Massively Parallel Computation (MPC) Model

MPC Model

- An abstract formal model to study large-scale parallel computations
  - Aims to study parallelism at a more coarse-grained level than classic fine-grained parallel models like PRAM
    (models settings where communication is much more expensive than computation)



Formal Model

- Input of size N words (1 word $= O(\log N)$ bits, for graphs, $N = O(|E|)$)
- There are $M \ll N$ machines

$S = \frac{N}{M} \cdot \text{polylog}(N)$

- Each machine has a memory of S words, i.e., we need $S \geq N/M$
  - We typically assume that $S = N^c$ for a constant $c < 1$

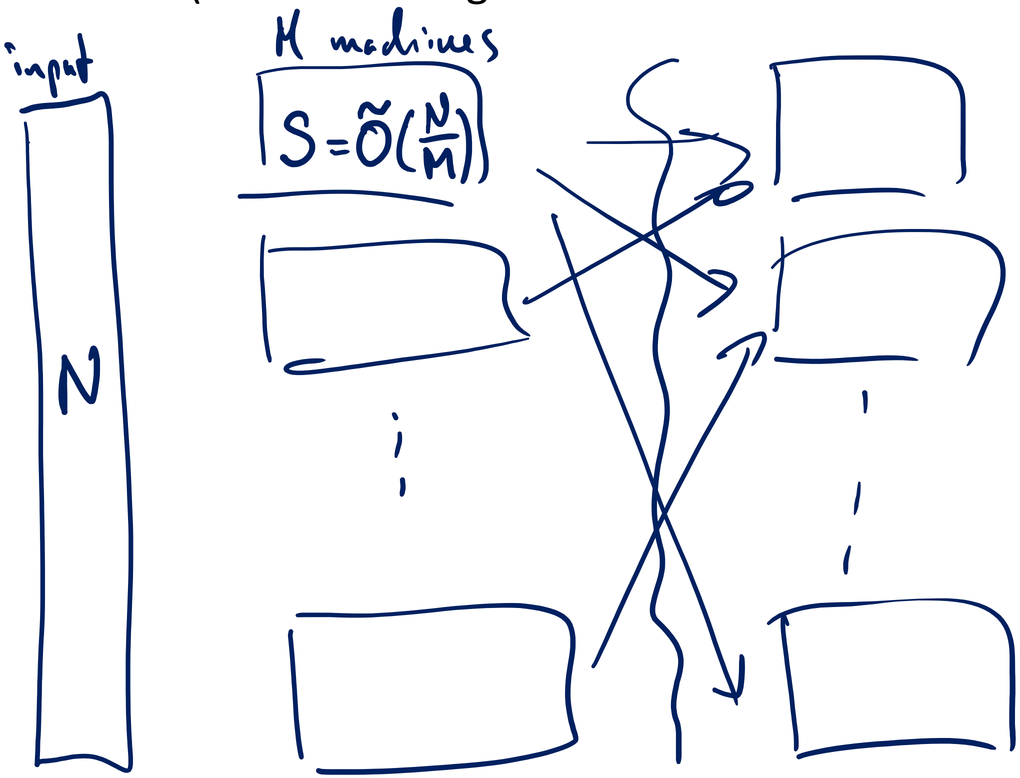
$M \geq \frac{N}{S} \cdot \text{polylog}(N)$

- Time progresses in synchronous rounds, in each round,
  every machine can send & receive S words to & from other machines
- Initially, the data is partitioned in an arbitrary way among the M machines
  - Such that every machine has a roughly equal part of the data
  - W.l.o.g., data is partitioned in a random way among the machines

Massively Parallel Computation (MPC) Model

MPC Model

- An abstract formal model to study large-scale parallel computations
  - Aims to study parallelism at a more coarse-grained level than classic fine-grained parallel models like PRAM
    (models settings where communication is much more expensive than computation)



time = #rounds

total comm. (send/rcv.) per machine is $\leq S$

MPC Model for Graph Computations

$S = m^c$

Assumption: Input is a graph $G = (V, E)$

- Number of nodes $n = |V|$, number of edges $m = |E|$, nodes have IDs
- Input can be specified by the set E of edges
  - each edge might have some other information, e.g., a weight
  - for simplicity, assume that every node has degree ≥ 1
- Initially, each edge is given to a uniformly random machine
- We typically assume that $S = \tilde{O}(N/M) = \tilde{O}(m/M)$

$\sim$ means that up to $(\log n)^{O(1)}$ factors

Strongly superlinear memory regime

$$S = n^{1+\varepsilon} \text{ for a constant } \varepsilon > 0$$

Strongly sublinear memory regime

$$S = n^{\alpha} \text{ for a constant } 0 < \alpha < 1$$

Near-linear memory regime

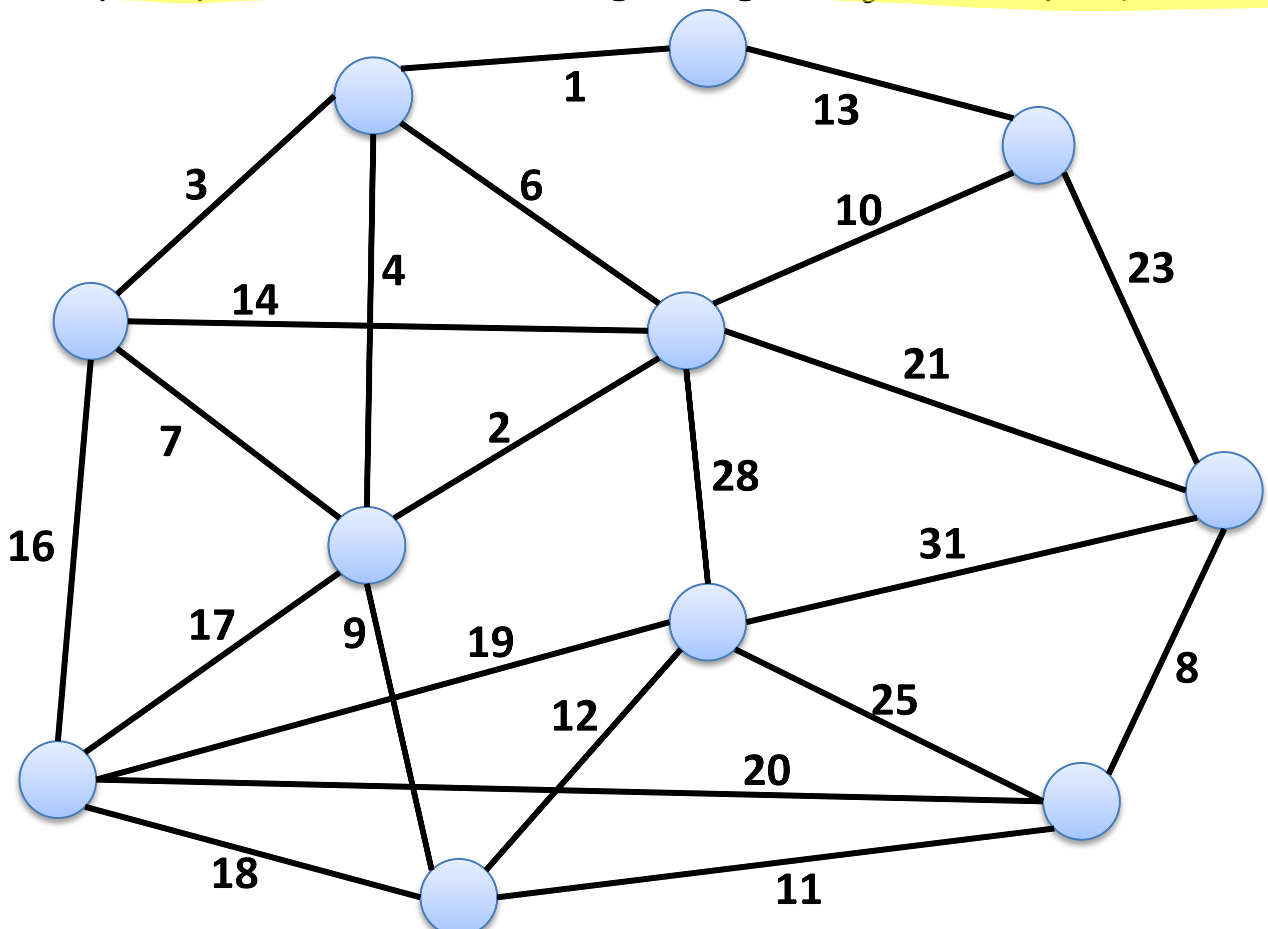
$$S = n \cdot \text{poly}\log n$$

Minimum Spanning Tree (MST) Problem

Given: connected graph $G = (V, E)$ with edge weights w_e

Goal: find a spanning tree $T = (V, E_T)$ of minimum total weight

- For simplicity, assume that the edge weights w_e are unique (makes MST unique)



Properties of the MST

$V' \subseteq V, \; E' \subseteq E$

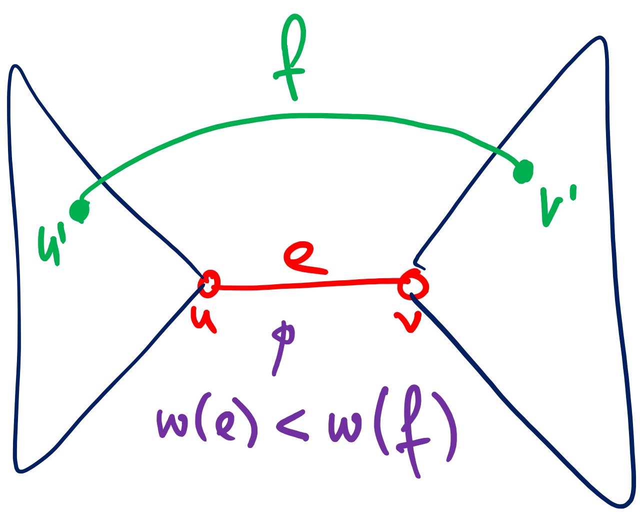
Minimum Spanning Forest (MSF) of G:

$e \in E' \cap E = E'$

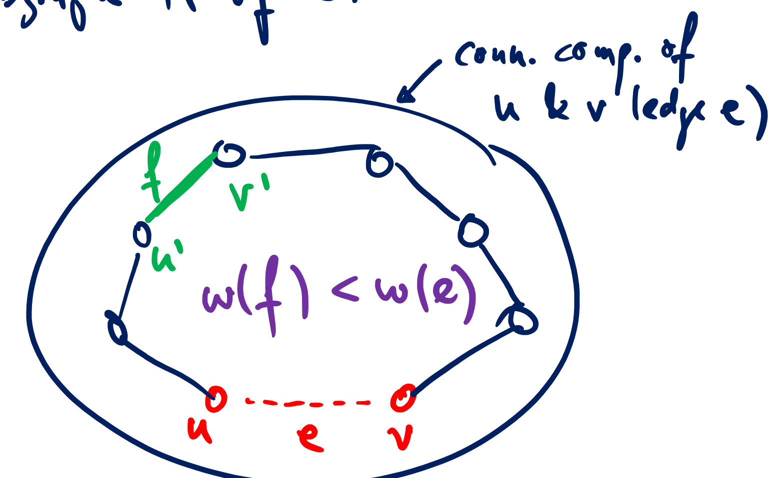
- A forest consisting of the MST of each of the connected components of G
  - Maximal forest of minimum total weight

Claim: Let $G = (V, E, w)$ be a weighted graph and let $H = (V', E', w)$ be a subgraph of G. If $e \in E'$ is an edge of the MST (or MSF) of G, then e is also an edge of the minimum spanning forest (MSF) of H

MST of G



subgraph H of G



assume for contradiction that e is not part of MSF of H

MST With Strongly Superlinear Memory

Initially:

- Each machine has $O(n^{1+\varepsilon})$ edges
  - There are $M = O(m/n^{1+\varepsilon})$ machines
- Let H_M be the subgraph induced by the edges of machine M

MPC Algorithm:

1. Each machine M computes minimum spanning forest F_M of H_M
2. Discard all edges that are not part of some MSF F_M
3. Remaining number of edges: M·(n−1)

$$m' \leq M \cdot n = O(m/n^{\varepsilon})$$

4. Redistribute remaining edges to $M' = O(m'/n^{1+\varepsilon})$ machines
   - Randomly reassign each edge

$m \to \frac{m}{n^{\varepsilon}} \to \frac{m}{n^{2\varepsilon}} \to \frac{m}{n^{3\varepsilon}}$

$m \leq n^2$

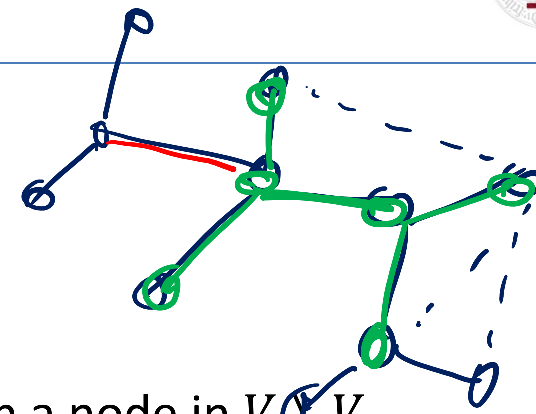
$n^{2-\varepsilon}$

$n^{2-2\varepsilon}$

- Algorithm reduces number of edges by factor $\Theta(n^{\varepsilon})$ in 1 round.
- $O(1/\varepsilon)$ repetitions suffice to solve the problem

$\leftarrow O(1)$

Borůvka's MST Algorithm



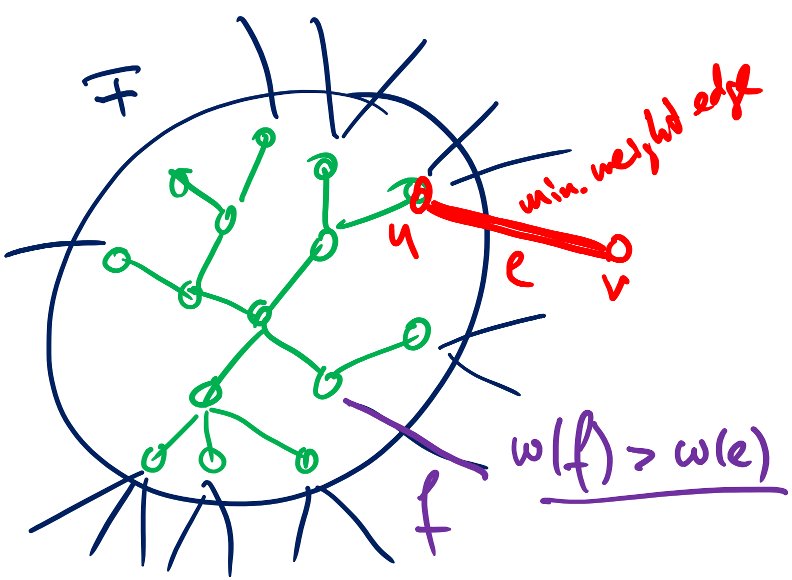
MST Fragment:

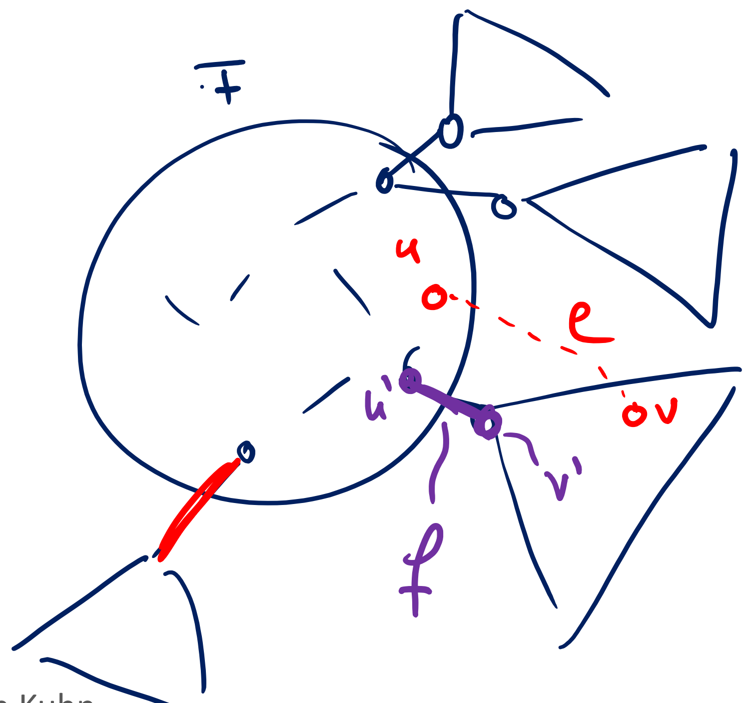
- A connected subtree $F = (V_F, E_F)$ of the MST

Minimum edge of MST fragment $\boldsymbol{F = (V_F, E_F)}$:

- Minimum weight edge connecting a node in V_F with a node in $V \setminus V_F$

Lemma: For every MST fragment F, the minimum edge of F is in the MST



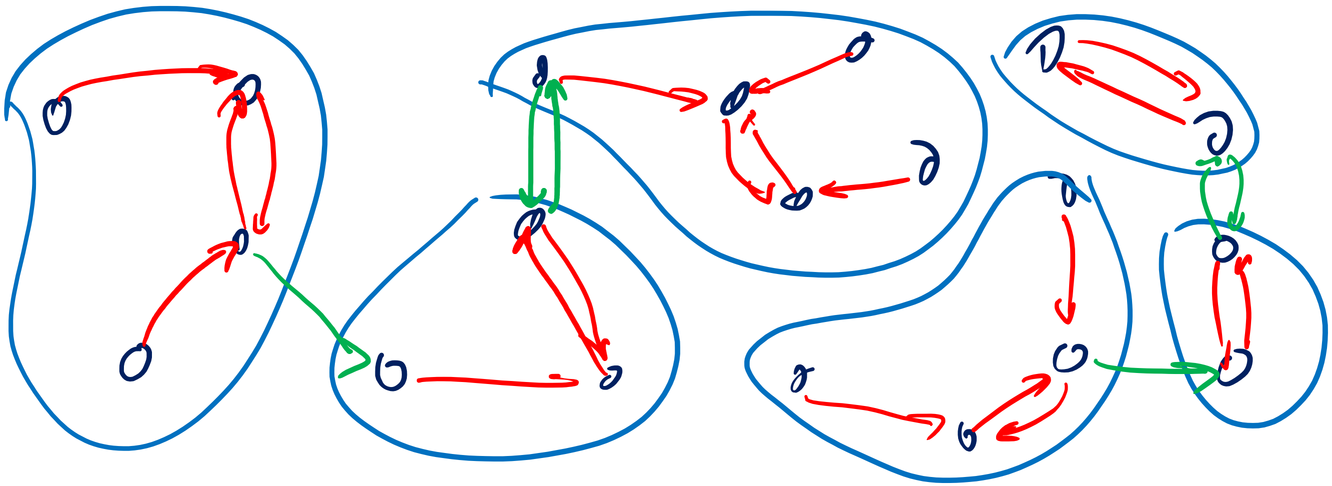


Borůvka's MST Algorithm

Algorithm description:

- Develops the MST in parallel phases
- Initially, each node is an MST fragment of size 1 (and with no edges)
- **In each phase:** *add the minimum edge of each fragment* to the MST
- Terminate when there is only one fragment
  - or when there are no edges between different fragments

Theorem: The above alg. computes the MST in $O(\log n)$ phases.



MST With Strongly Sublinear Memory: Ideas

Assume: $G = (V, E)$ with n nodes, m edges, memory $S = n^{\alpha}$ for const. $\alpha > 0$

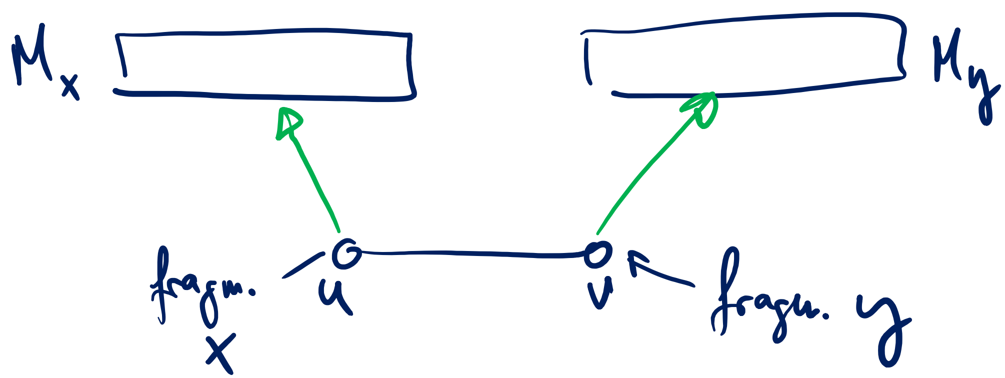
- Also assume that we have $M \geq m/S \cdot c \log n$ machines for suff. large $c \geq 1$

Representation of algorithm state:

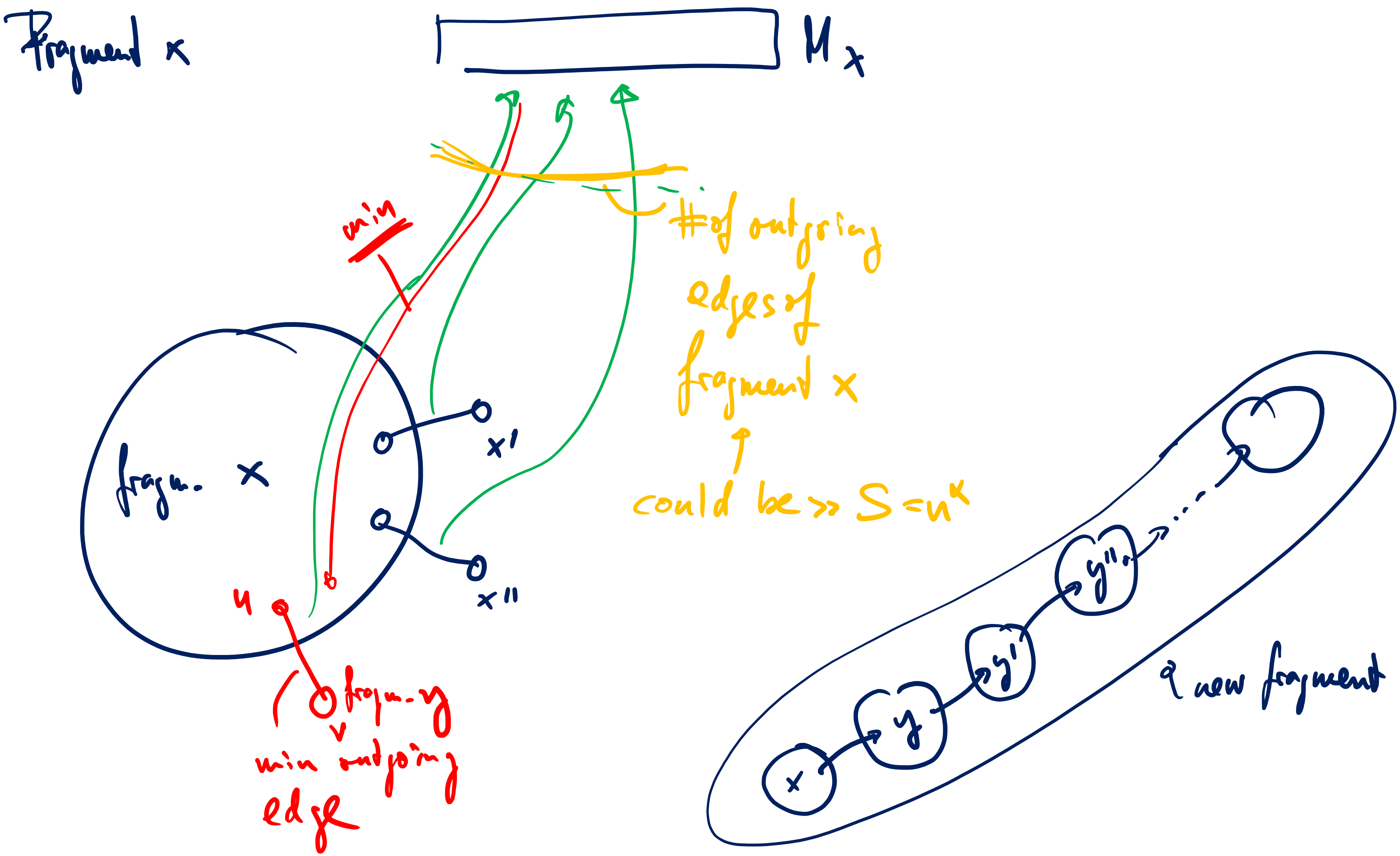
- Each fragment has a unique ID, fragment ID of node u: $\mathrm{FID}(u)$
- The machine storing an edge $\{u, v\}$ knows the fragment IDs of u and v

Goal: implement one phase in time $O(1)$:

- Assume that for each fragment ID x, there is some responsible machine M_x
  - Additional empty machines that are randomly assigned (e.g. by a hash function)
- For now, assume that each node u directly interacts with machine $M_{\mathrm{FID}(u)}$

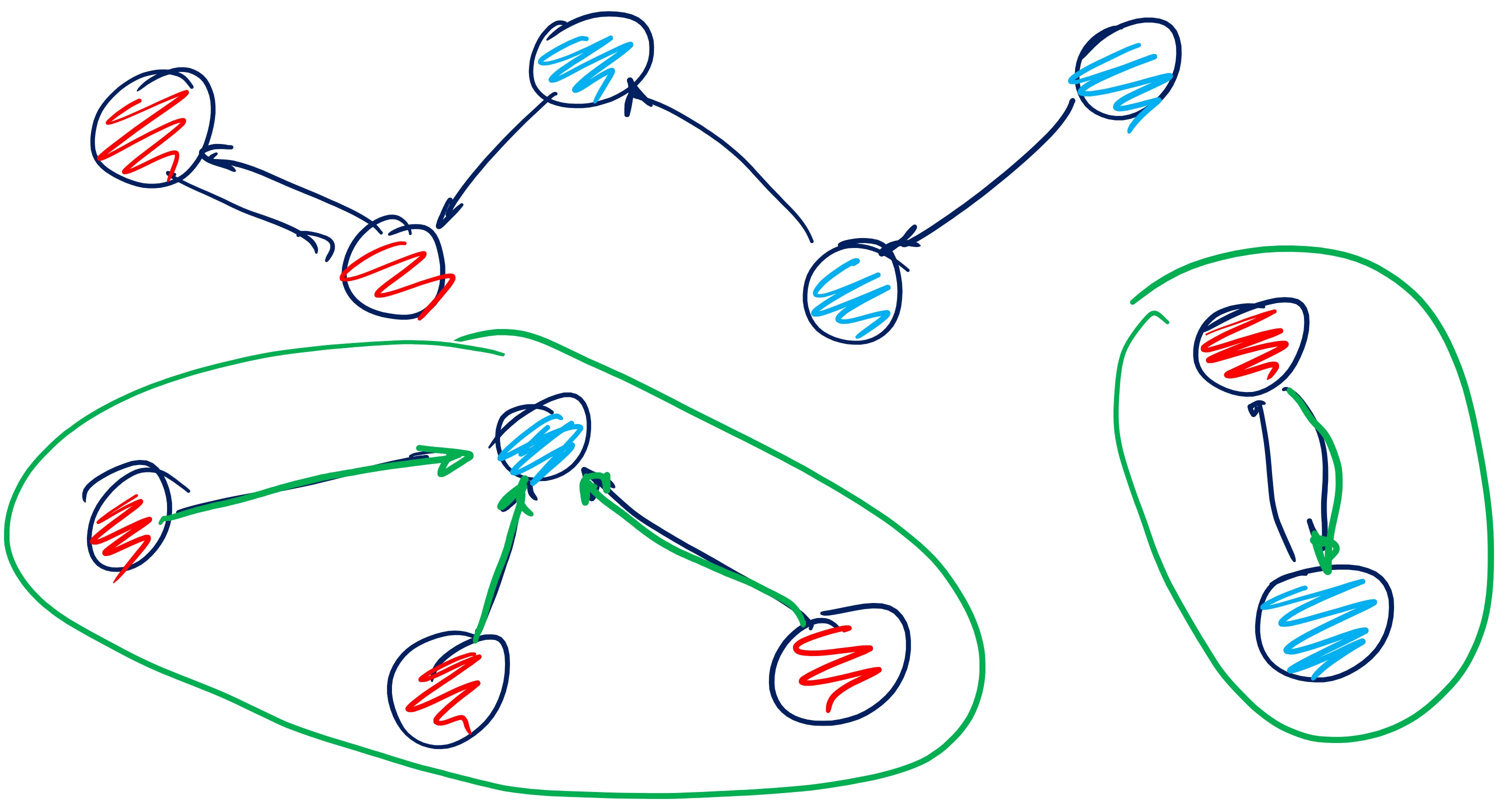


Implementing One Phase (First Attempt)



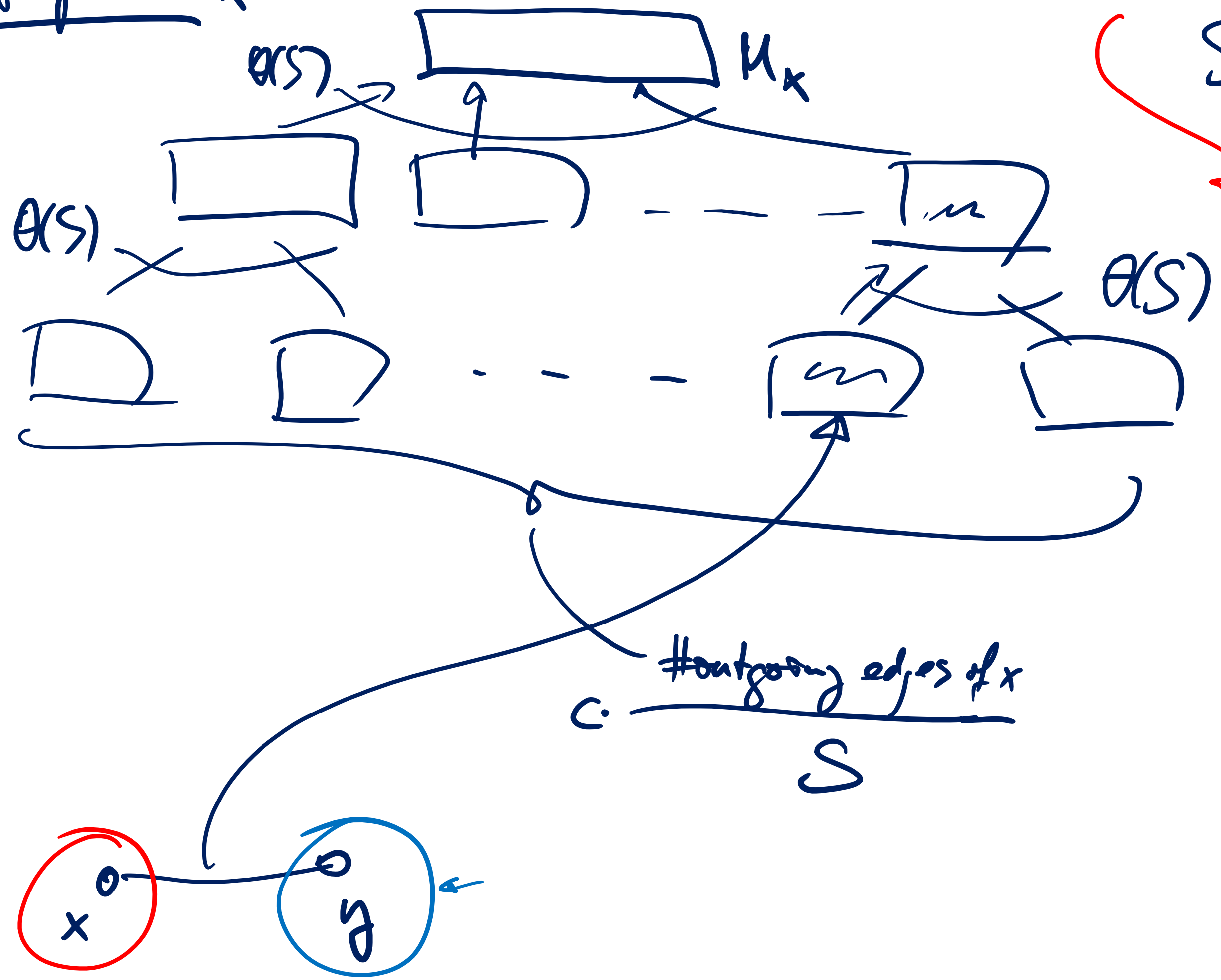
Small Change to the Basic Algorithm

- In each phase, each fragment initially picks a random color in $\{\text{red}, \text{blue}\}$
- Let $\{u, v\}$ be the minimum edge of a fragment F
- Only add $\{u, v\}$ to MST in current phase if F is a red fragment and $\{u, v\}$ connects to a blue fragment.



Implementation with Aggregation Trees

for each fragment x



$S = n^{\alpha}$

height h

$S^h > n^2$

$\hookrightarrow h = O(1)$

MST with Strongly Sublinear Memory

Theorem: In the strongly sublinear memory regime (i.e., when $S = n^{\alpha}$ for a constant $\alpha \in (0,1)$), an MST can be computed in time $\underline{O(\log n)}$.

Borůvka's algorithm: $O(\log n)$ phases

$O(1)$ rounds per phase