An MST

• Minimum Spanning Tree
  – Input: a weight function $w:E \rightarrow R$
  – Output: a spanning tree $T$ of $G$, such that for every spanning tree $S$ of $G$, $w(T) \leq w(S)$
    • $w(S) = \sum_{e \in S} w(e)$

• Distributed setting:
  – Input to node $v$: the weight $w(e)$ of each $e$ touching $v$
  – Output: which edges touching $v$ are in $T$
    • No knowledge of the entire tree is required
Model

• **Synchronous** system. **Complexity**: number of rounds
• In each round:

  node $v$ can send messages to every node in $N(v)$

  node $v$ can send messages of $O(\log n)$ bits to every node in $N(v)$

  – Motivation: sending an ID in a single message
MST in CONGEST

• In general, we can replace $O(\log n)$ for the number of bits by any parameter $B$.

• We need to make sure that edge weights also fit in a message, so we assume $w : E \rightarrow W$, where $W = \{0,1,...,\text{poly}(n)\}$.

• **Assumption:** $w$ is 1:1 (there is a unique MST)
Sequential MST

- **Kruskal**: removing heavy edges from cycles by going over edges in increasing order of weights, adding an edge to T if it does not create a cycle *(red rule)*

- **Prim**: maintain a connected component by adding the lightest edges leaving it *(blue rule)*

- **Burovka**: initially each node is a connected component. Go over connected components in arbitrary order and added lightest edge leaving the component *(blue rule)*
MST in CONGEST

• We can simulate each of the above algorithms
  – In a naïve manner: going over all $m$ edges in each iteration. **Time: $O(nm)$** rounds

• In $O(m)$ rounds we can learn the graph

• **We saw:** An $O(n)$-round BFS-based algorithm that simulates **Kruskal’s** algorithm
The GHS Algorithm for MST

• Gallager-Humblet-Spira

• **Combinatorial claim:** For every subset \( S \) of \( V \), the lightest edge from \( S \) to \( V \setminus S \) belongs to the MST.

• **High-level description:** Simulate the Burovka/Prim algorithms by maintaining a connected component that grows by adding the lightest edge leaving it.
The GHS Template

Variables:

$T$, initially empty

1 repeat
2 \( F \leftarrow \) set of connected components of $T$
3 \textbf{For} each $C \in F$
4 \hspace{1cm} add to $T$ the lightest edge leaving $C$
5 \textbf{until} $T$ is a spanning subgraph
6 \hspace{1cm} (no outgoing edges, single component)
6 return $T$
Correctness

• Still without implementation details

• A phase: one iteration of the loop

• **Claim 1**: The returned set of edges $T$ is an MST.

• **Proof**: An edge $e$ is added to $T$ only if it is the lightest leaving $C$. By the *combinatorial claim*, $e$ belongs to the MST. We return only when we have a spanning subgraph.
Complexity

• **Claim 2**: The algorithm completes after $O(\log n)$ phases.

• **Proof**: We prove by induction, that at the end of phase $i$, the size of each connected component is at least $2^i$.

• **Base case**: For $i=0$, at the end of the initialization, each singleton is a connected component of size 1.
Complexity

• **Induction hypothesis:** At the end of phase $i-1$, each connected component is of size at least $2^{i-1}$.

• **Induction step:** In phase $i$, each connected component adds an edge to another connected component. By the induction hypothesis, the size of each new component is at least $2^{i-1} + 2^{i-1} = 2^i$.

• Hence, after $O(\log n)$ iterations there is a single connected component.
Implementation

• What does it mean for a connected component to choose an edge?

• For each component $C$, we assign a root node $r_C$, which is the node with the smallest ID.

• The ID of $r_C$ is the ID of the component $C$. 

Implementation

• In each phase, every node $v$ sends to all of its neighbors a triplet $(u, w(e), C')$
  – $e=\{v,u\}$ is the lightest edge that leaves $v$ to a different component
  – $w(e)$ is the weight of $e$
  – $C'$ is the ID of the component of $u$
Implementation

• Each node forwards the triplet of the lightest edge it received so far, towards $r_c$ using the edges of $T$.

• The root $r_c$ picks the triplet of the lightest edge it received and sends it back to all nodes of the component.
Implementation

• The chosen node $v$ sends a message to its neighbor $u$, which forwards it through $C'$. The node with the minimal ID among all roots of the merged components becomes the new root.

— There are missing details in the above description
Complexity – cont.

**Theorem:** The GHS algorithm computes an MST within $O(n\log n)$ rounds.

**Proof:**

- Correctness follows from **Claim 1**.
  - with additional details
- **Claim 2** gives that there are $O(\log n)$ phases.
- The implementation completes a phase in $O(n)$ rounds, since this is the maximum diameter of each component.
GHS – notes

• This was not a formal pseudocode
• The complexity can be reduced to $O(n)$
• The GHS algorithm can be implemented in an asynchronous setting within $O(m \cdot \log n)$ messages
  – This can be improved to $O(m + n \cdot \log n)$
Distributed MST Algorithms

• The **BFS-based** algorithm takes a linear number of rounds because it maintains a single connected component.

• The **GHS** algorithm is slow because despite merging components fast, their diameter may become linear very early.
The GKP algorithm for MST

• The Garay-Kutten-Peleg (GKP) algorithm which we will see next, merges components in a more careful manner that restricts their diameter
The GKP Template

Variables:
$T$, initially empty

1 for $i=1,\ldots,\log vn$ do
2 $F \leftarrow$ set of connected components of $T$,
3 $S \leftarrow$ empty
4 For each $C \in F$ of $\text{diam}(C) \leq 2^{i-1}$
5 add lightest outgoing edge to $S$
6 add a maximal matching $S_M$ in $S$ to $T$
7 If $C$ is not matched
8 add its edge to $T$
9 contract edges in $T$, run the **BFS-based algorithm** and add edges to $T$
10 return $T$
Maximal Matching

• A set of edges $M$ in $E$
  – No two edges in $M$ share an endpoint
  – Every $e \in E$ has an $e' \in M$ with the same endpoint

• Illustration

• Claim 1: In a graph with outdegree 1, a maximal matching can be found in $O(\log^* n)$ rounds

• $\log^* n = \text{the minimal } k \text{ such that } \log(\log\ldots(n)) \leq 1$

  \[ k \text{ times} \]
Correctness

• **Claim 2**: The algorithm returns an MST

• **Proof**:  
  • Edges added to $T$ are only lightest edges between components.  
  • Edges are added until there is a single component.
Analysis

- **Claim 3**: At the end of iteration $i$, the diameter of each component is at most $6 \cdot 2^i$.

- **Claim 4**: At the end of the last iteration, there are at most $\sqrt{n}$ components.
Analysis

• **Claim 5**: At the end of the last iteration, if there are $k$ components with maximal diameter $D_{\text{max}}$, then line 9 completes in $O(D+D_{\text{max}}+k)$ rounds.
GKP - Complexity

• **Theorem**: The GKP algorithm finds an MST in $O(\sqrt{n}\log^* n + D)$ rounds.

• **Proof**: By Claim 2, the algorithm returns an MST. By Claims 1 and 3, each iteration completes in $O(2^i \log^* n)$ rounds, which in total gives $\Sigma_{i=1}^{\log\sqrt{n}} O(2^i \log^* n) = O(2^{\log\sqrt{n}} \log^* n) = O(\sqrt{n}\log^* n)$ rounds.

• By Claims 4 and 5, the last step takes $O(D + \sqrt{n} + 2^{\log\sqrt{n}}) = O(D + \sqrt{n})$ rounds.
Analysis – Claim 3

• **Claim 3**: At the end of iteration $i$, the diameter of each component is at most $6 \cdot 2^i$.

• **Proof**: By induction on $i$.

• **Base case**: For $i=0$, the diameter is 0.

• **Induction hypothesis**: At the end of iteration $i$, the diameter of each component is at most $6 \cdot 2^i$. 
Analysis – Claim 3

• **Induction step**: At the end of iteration $i+1$, we claim that the longest distance in each *new* component is created by traveling through at most 3 *old* components of diameter at most $2^i$ and 1 *old* component of diameter at most $6 \cdot 2^i$.

• **Illustration**
Analysis – Claim 3

• **Because:** if $C$ adds an edge to $C'$ then $C'$ is matched (since the matching is maximal) to some $C''$. Only one of $C'$ or $C''$ can have diameter larger than $2^i$, and by the induction hypothesis its diameter is at most $6 \cdot 2^i$. (For $i=0$ all **old** diameters are 1) For $i \geq 1$:

$$6 \cdot 2^i + 3 \cdot 2^i + 3 \leq 9 \cdot 2^i + 3$$

$$\leq 5 \cdot 2^{i+1} + 3$$

$$\leq 5 \cdot 2^{i+1} + 2^{i+1}$$

$$\leq 6 \cdot 2^{i+1}$$
Analysis – Claim 4

• **Claim 4:** At the end of the last iteration, there are at most $\sqrt{n}$ components.

• **Proof:** By induction on $i$: at the end of iteration $i$, all components are of size at least $2^i$.

• **Base case:** for $i=0$ all components are of size $1$. 
Analysis – Claim 4

- **Claim 4**: At the end of the last iteration, there are at most $\sqrt{n}$ components.

- **Induction hypothesis**: at the end of iteration $i-1$, all components are of size at least $2^{i-1}$.

- **Induction step**: in iteration $i$, every *old* component of size at most $2^{i-1}$ connects to at least one other *old* component, creating a *new* component of size at least $2 \cdot 2^{i-1} = 2^i$. 
Analysis – Claim 5

• **Claim 5**: At the end of the last iteration, if there are $k$ components with maximal diameter $D_{\text{max}}$, then line 9 completes in $O(D + D_{\text{max}} + k)$ rounds.

• **Proof**: We need $D_{\text{max}}$ rounds to have an ID for each component, and then we can implement the BFS-based $O(D + k)$-round algorithm.
MST in CONGEST

• **BFS-based** algorithm in $O(n)$ rounds

• **GHS algorithm** in $O(n \log n)$ rounds

• **GKP algorithm** in $O(Vn \log^* n + D)$ rounds

• **Question**: $D$ is necessary. What about $\sqrt{n}$?