The Predicted-Updates Dynamic Model

Quanquan C. Liu
quanquan.liu@yale.edu

Vaidehi Srinivas
vaidehi@u.northwestern.edu

Yale

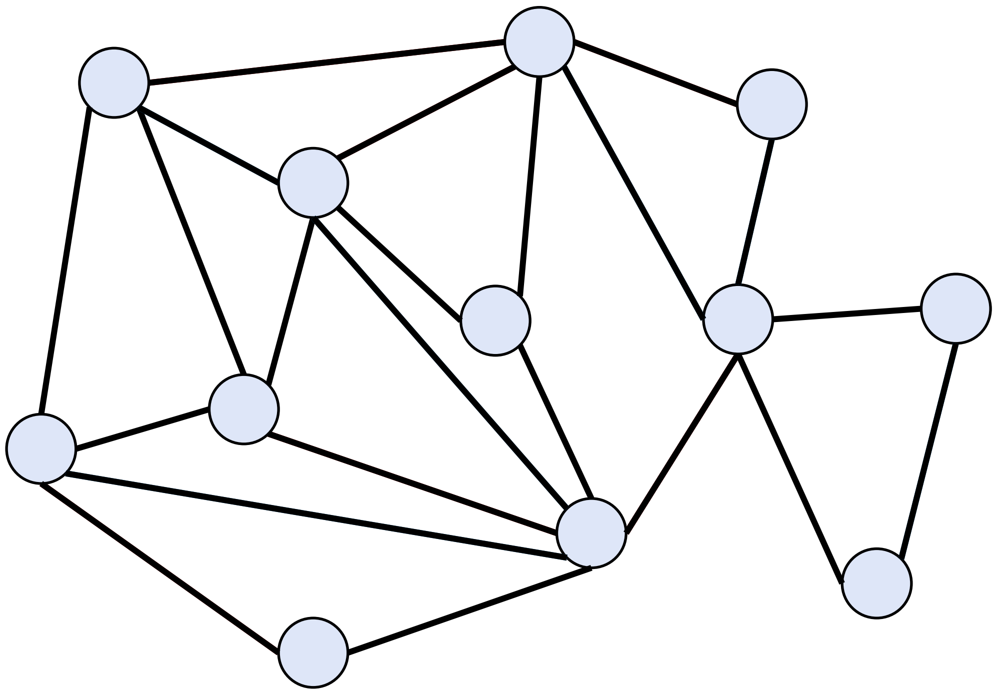
Northwestern University

Dynamic Algorithms

- Updates to the dataset (e.g. graph) occurs where elements are added and deleted from the dataset

Edge **insertions/deletions** arrive **sequentially**

Maintain **graph property** after each update



Minimize Update Time

- **Want**: minimize the update time between updates
    - Amortized or worst-case (often a gap)

Sublinear Runtime:
strive for $\mathrm{poly}(\log n)$

Minimize Update Time

- **Want**: minimize the update time between updates
  - Amortized or worst-case (often a gap)
- Sometimes need to do **preprocessing**
  - Small polynomial in the input graph

Sublinear Runtime:
strive for $\mathrm{poly}(\log n)$

Minimize Update Time

- **Want**: minimize the update time between updates
  - Amortized or worst-case (often a gap)
- Sometimes need to do **preprocessing**
  - Small polynomial in the input graph
- Sometimes have **queries** (e.g. connectivity queries)

Sublinear Runtime:
strive for $\text{poly}(\log n)$

Many Recent Results in Dynamic Graph Algorithms in $\mathrm{poly}(\log n)$ Runtime

- Dynamic maximum matching (find a matching of maximum size):
  - **Best known:** $(1.973 + \varepsilon)$-approximation in $\mathrm{poly}(\log n)$ update time [BKSW SODA '23]

Many Recent Results in Dynamic Graph Algorithms in $\mathrm{poly}(\log n)$ Runtime

- Dynamic maximum matching (find a matching of maximum size):
  - **Best known:** $(1.973 + \varepsilon)$-approximation in $\mathrm{poly}(\log n)$ update time [BKSW SODA ‘23]
- Dynamic $(\Delta + 1)$-coloring (find a valid coloring with $\Delta + 1$ colors):
  - **Best known:** $O(1)$ update time [HP, BGK**L**S TALG ‘22]

Many Recent Results in Dynamic Graph Algorithms in $\mathrm{poly}(\log n)$ Runtime

- Dynamic maximum matching (find a matching of maximum size):
  - **Best known:** $(1.973 + \varepsilon)$-approximation in $\mathrm{poly}(\log n)$ update time [BKSW SODA '23]
- Dynamic $(\Delta + 1)$-coloring (find a valid coloring with $\Delta + 1$ colors):
  - **Best known:** $O(1)$ update time [HP, BGK**L**S TALG '22]
- Approximate Densest Subgraph:
  - **Best known:** $\mathrm{poly}(\log n)$ update time [SW STOC '20, CCHHQRS SODA '24]

Many Recent Results in Dynamic Graph Algorithms in $\text{poly}(\log n)$ Runtime

Other algorithms have significantly larger runtimes!

- Approximate Densest Subgraph:
  - **Best known:** $\text{poly}(\log n)$ update time [SW STOC '20, CCHHQRS SODA '24]

Types of Dynamic Algorithms

- Gaps in runtime between **different types of dynamic algorithms**

Types of Dynamic Algorithms

- Gaps in runtime between **different types of dynamic algorithms**
- **Offline** vs. Fully Dynamic
  - Receive all of the updates at once, process them together

Types of Dynamic Algorithms

- Gaps in runtime between **different types of dynamic algorithms**
- **Offline** vs. Fully Dynamic
  - Receive all of the updates at once, process them together
- **Incremental/Decremental** vs. **Fully Dynamic**
  - Incremental/decremental algorithms:
    - Only edge insertions/deletions, respectively

Types of Dynamic Algorithms

- Gaps in runtime between **different types of dynamic algorithms**
- **Offline** vs. Fully Dynamic
  - Receive all of the updates at once, process them together
- **Incremental/Decremental** vs. **Fully Dynamic**
  - Incremental/decremental algorithms:
    - Only edge insertions/deletions, respectively
- Sometimes **large gap in runtimes**
  - **Polynomial** or **exponential gaps** in runtimes

[**L**, Srinivas COLT '24]

Types of Dynamic Algorithms

	Best Fully Dynamic		Best Offline/Partially Dynamic	
Planar Digraph APSP	$\widetilde{O}\left(n^{2/3}\right)$	[FR06, Kle05]	$\widetilde{O}(\sqrt{n})$	[DGWN22]
Triconnectivity	$O(n^{2/3})$	[GIS99]	$\widetilde{O}(1)$	[HR20, PSS17]
k-Edge Connectivity	$n^{o(1)}$	[JS22]	$\widetilde{O}(1)$	[CDK+21]
APSP	$\left(\frac{256}{k^2}\right)^{4/k}$-Approx $\widetilde{O}\left(n^k\right)$ update $\widetilde{O}(n^{k/8})$ query	[FGNS23]	$(2r-1)^k$-Approx $\widetilde{O}\left(m^{1/(k+1)}n^{k/r}\right)$	[CGH+20]
AP Maxflow/Mincut	$O(\log(n)\log\log n)$-Approx $\widetilde{O}\left(n^{2/3+o(1)}\right)$	[CGH+20]	$O\left(\log^{8k}(n)\right)$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$	[Gor19, GHS19]
MCF	$(1+\varepsilon)$-Approx $\widetilde{O}(1)$ update $\widetilde{O}(n)$ query	[CGH+20]	$O(\log^{8k}(n))$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$ update $\widetilde{O}(P^2)$ query	[Gor19, GHS19]
Uniform Sparsest Cut	$2^{O(\log^{5/6}(n))}$-Approx $2^{O(\log^{5/6}(n))}$ update $O(\log^{1/6}(n))$ query	[GRST21]	$O\left(\log^{8k}(n)\right)$-Approx $\widetilde{O}\left(n^{2/(k+1)}\right)$ $O(1)$ query	[Gor19, GHS19]
Submodular Max	$1/4$-Approx $\widetilde{O}(k^2)$	[DFL+23]	0.3178-Approx $\widetilde{O}(\text{poly}(k))$	[FLN+22]

[**L**, Srinivas COLT '24]

Types of Dynamic Algorithms

	Best Fully Dynamic		Best Offline/Partially Dynamic	
Planar Digraph APSP	$\widetilde{O}\left(n^{2/3}\right)$	[FR06, Kle05]	$\widetilde{O}(\sqrt{n})$	[DGWN22]
Triconnectivity	$O\left(n^{2/3}\right)$	GIS99	$\tilde{O}(1)$	HR20, PSS17
k-Edge Connectivity	$n^{O(1)}$	JS22	$\tilde{O}(1)$	CDKLLPSV21
APSP	$\widetilde{O}\left(n^{k}\right)$ update $\widetilde{O}(n^{k/8})$ query	[FGNS23]	$(2r-1)^k$-Approx $\widetilde{O}\left(m^{1/(k+1)}n^{k/r}\right)$	[CGH+20]
AP Maxflow/Mincut	$O(\log(n)\log\log n)$-Approx $\widetilde{O}\left(n^{2/3+o(1)}\right)$	[CGH+20]	$O\left(\log^{8k}(n)\right)$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$	[Gor19, GHS19]
MCF	$(1+\varepsilon)$-Approx $\widetilde{O}(1)$ update $\widetilde{O}(n)$ query	[CGH+20]	$O(\log^{8k}(n))$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$ update $\widetilde{O}(P^2)$ query	[Gor19, GHS19]
Uniform Sparsest Cut	$2^{O(\log^{5/6}(n))}$-Approx $2^{O(\log^{5/6}(n))}$ update $O(\log^{1/6}(n))$ query	[GRST21]	$O\left(\log^{8k}(n)\right)$-Approx $\widetilde{O}\left(n^{2/(k+1)}\right)$ $O(1)$ query	[Gor19, GHS19]
Submodular Max	$1/4$-Approx $\widetilde{O}(k^2)$	[DFL+23]	0.3178-Approx $\widetilde{O}(\text{poly}(k))$	[FLN+22]

[L, Srinivas COLT '24]

Types of Dynamic Algorithms

Problem	Best Fully Dynamic		Best Offline/Partially Dynamic	
Planar Digraph APSP	$\tilde{O}(n^{2/3})$	[FR06, Kle05]	$\tilde{O}(\sqrt{n})$	[DGWN22]
Triconnectivity	$O(n^{2/3})$	GIS99	$\tilde{O}(1)$	HR20, PSS17
k-Edge Connectivity	$n^{O(1)}$	JS22	$\tilde{O}(1)$	CDKLLPSV21
APSP	$\tilde{O}(n^k)$ update $\tilde{O}(n^{k/8})$ query	[FGNS23]	$(2r-1)^k$-Approx $\tilde{O}(m^{1/(k+1)}n^{k/r})$	[CGH+20]
AP Maxflow/Mincut	$O(\log(n)\log\log n)$-Approx $\tilde{O}(n^{2/3+o(1)})$	[CGH+20]	$O(\log^{8k}(n))$-Approx. $\tilde{O}(n^{2/(k+1)})$	[Gor19, GHS19]
				HS19]
Uniform Sparsest Cut	$O(\log^{1/6}(n))$ query	[GRST21]	$O(1)$ query	[Gor19, GHS19]
Submodular Max	1/4-Approx $\tilde{O}(k^2)$	[DFL+23]	0.3178-Approx $\tilde{O}(\text{poly}(k))$	[FLN+22]

How do we close the gap?

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?
  - Yes!

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?
    - Yes!
- **Predictions on the edge updates [L-Srinivas COLT '24]**

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?
  - Yes!
- **Predictions on the edge updates [L-Srinivas COLT ‘24]**
  - For each edge update, give **prediction on when update occurs**

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?
  - Yes!
- **Predictions on the edge updates [L-Srinivas COLT '24]**
  - For each edge update, give **prediction on when update occurs**
  - Assume one edge insertion/deletion occurs at a time, give prediction on the time of the edge insertion/deletion, T **total update times**

Learning-Augmented Dynamic Graph Algorithms

- Can we use predictions to bridge the gap?
  - Yes!
- **Predictions on the edge updates [L-Srinivas COLT '24]**
  - For each edge update, give **prediction on when update occurs**
  - Assume one edge insertion/deletion occurs at a time, give prediction on the time of the edge insertion/deletion, T **total update times**
- Runtime in terms of L_1-error between predictions and real times

Learning-Augmented Dynamic Graph Algorithms

- Runtime in terms of L_1-error between predictions and real

Learning-Augmented Dynamic Graph Algorithms

- Runtime in terms of L_1-error between predictions and real
  - $\mathbf{p}$ vector of **predicted** timestamps
  - $\mathbf{r}$ vector of **real** timestamps
  - L_1 error: $||\mathbf{p} - \mathbf{r}||_1$

Learning-Augmented Dynamic Graph Algorithms

- Runtime in terms of L_1-error between predictions and real
  - $\mathbf{p}$ vector of **predicted** timestamps
  - $\mathbf{r}$ vector of **real** timestamps
  - L_1 error: $||\mathbf{p} - \mathbf{r}||_1$

Runtime same as partially dynamic with total

$$\tilde{O}\left(\left(||\mathbf{p} - \mathbf{r}||_1 + T\right) \cdot update\right)$$

time for T updates

[L-Srinivas COLT '24]

Learning-Augmented Dynamic Graph Algorithms

- Runtime in terms of L_1-error between predictions and real
  - $\mathbf{p}$ vector of **predicted** timestamps
  - $\mathbf{r}$ vector of **real** timestamps
  - L_1 error: $||\mathbf{p} - \mathbf{r}||_1$

$update$ is **worst-case update time** of **incremental/decremental** algorithm

Runtime same as partially dynamic with total

$$\tilde{O}\left(\left(||\mathbf{p} - \mathbf{r}||_1 + T\right) \cdot update\right)$$

time for T updates

[L-Srinivas COLT '24]

Related Work

- [McCauley-Moseley-Niaparast-Singh NeurIPS ‘23, ICML ‘24]:
    - Incremental/online list labeling, topological ordering, and cycle detection

Related Work

- [McCauley-Moseley-Niaparast-Singh NeurIPS ‘23, ICML ‘24]:
  - Incremental/online list labeling, topological ordering, and cycle detection
- [Peng-Rubinstein SOSA ‘23]:
  - Incremental to fully dynamic transformation given deletion order

Related Work

- [McCauley-Moseley-Niaparast-Singh NeurIPS ‘23, ICML ‘24]:
  - Incremental/online list labeling, topological ordering, and cycle detection
- [Peng-Rubinstein SOSA ‘23]:
  - Incremental to fully dynamic transformation given deletion order
- [van den Brand-Forster-Nazari-Polak SODA ‘24]:
  - Incremental to fully dynamic transformation when given predictions of deletion order and other models

Related Work

- [McCauley-Moseley-Niaparast-Singh NeurIPS '23, ICML '24]:
  - Incremental/online list labeling, topological ordering, and cycle detection
- [Peng-Rubinstein SOSA '23]:
  - Incremental to fully dynamic transformation given deletion order
- [van den Brand-Forster-Nazari-Polak SODA '24]:
  - Incremental to fully dynamic transformation when given predictions of deletion order and other models
- [Henzinger-Saha-Seybold-Ye ITCS '24]:
  - Various lower bounds for different models of learning-augmented dynamic algorithms

What Guarantees Do We Want?

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm

What Guarantees Do We Want?

Match the offline, incremental, decremental runtimes up to $\text{poly}(\log T)$ factors

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm

What Guarantees Do We Want?

Match **the offline, incremental, decremental runtimes** **up to** $\mathrm{poly}(\log T)$ **factors**

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm
- **Competitiveness**
  - If the predictions are **low quality**, then **algorithm does not perform any worse than worst-case** algorithm

What Guarantees Do We Want?

Match the offline, incremental, decremental runtimes up to $\mathrm{poly}(\log T)$ factors

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm
- **Competitiveness**
  - If the predictions are **low quality**, then **algorithm does not perform any worse than worst-case** algorithm

Match the best-known fully dynamic

What Guarantees Do We Want?

Match **the offline, incremental, decremental runtimes** **up to** $\mathbf{poly(\log T)}$ **factors**

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm
- **Competitiveness**
  - If the predictions are **low quality**, then **algorithm does not perform any worse than worst-case** algorithm
- **Robustness**
  - Performance of algorithm **degrades gracefully** as function of prediction error

Match **the** **best-known fully dynamic**

What Guarantees Do We Want?

Match **the offline, incremental, decremental runtimes** **up to** $\mathrm{poly}(\log T)$ **factors**

- **Consistency**
  - If the predictions are **high quality**, then algorithm performs **much better than worst-case** algorithm
- **Competitiveness**
  - If the predictions are **low quality**, then **algorithm does not perform any worse than worst-case** algorithm
- **Robustness**
  - Performance of algorithm **degrades gracefully** as function of prediction error

Match **the** **best-known fully dynamic**

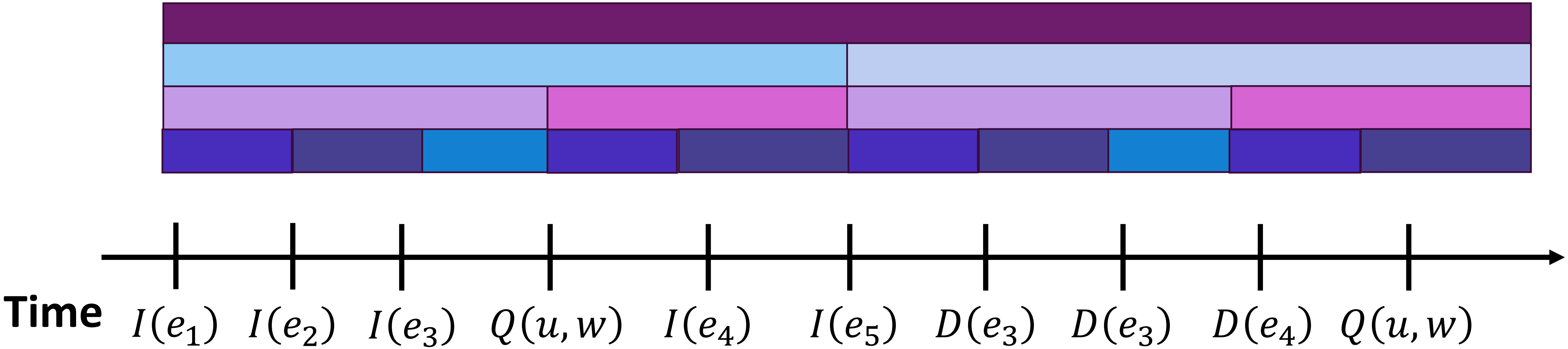
Degrades as a **linear function of the** L_1 **error**

Offline to Fully Dynamic Transformation

- Transforms an **offline dynamic divide-and-conquer algorithm to fully dynamic algorithm**

Offline to Fully Dynamic Transformation

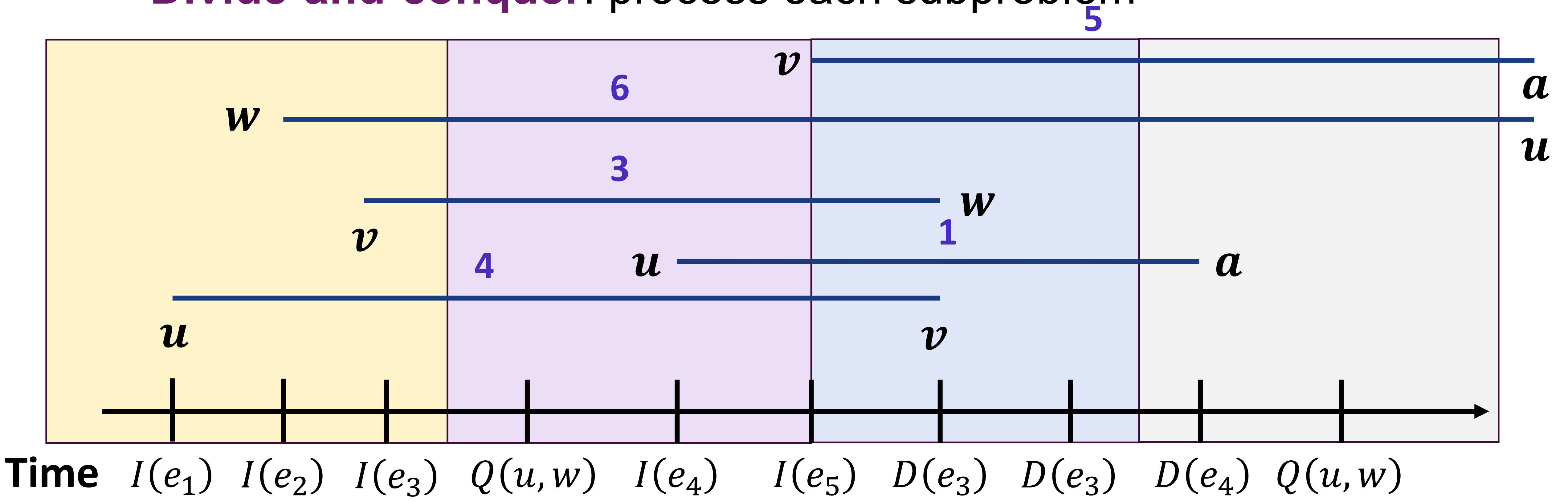
- Transforms an **offline dynamic divide-and-conquer algorithm to fully dynamic algorithm**



Solution for each update obtained from divide-and-conquer over update timestamps

Offline-Dynamic Minimum Spanning Tree

- Geometric representation of the problem
- **Divide-and-conquer**: process each subproblem



Offline-Dynamic Minimum Spanning Tree

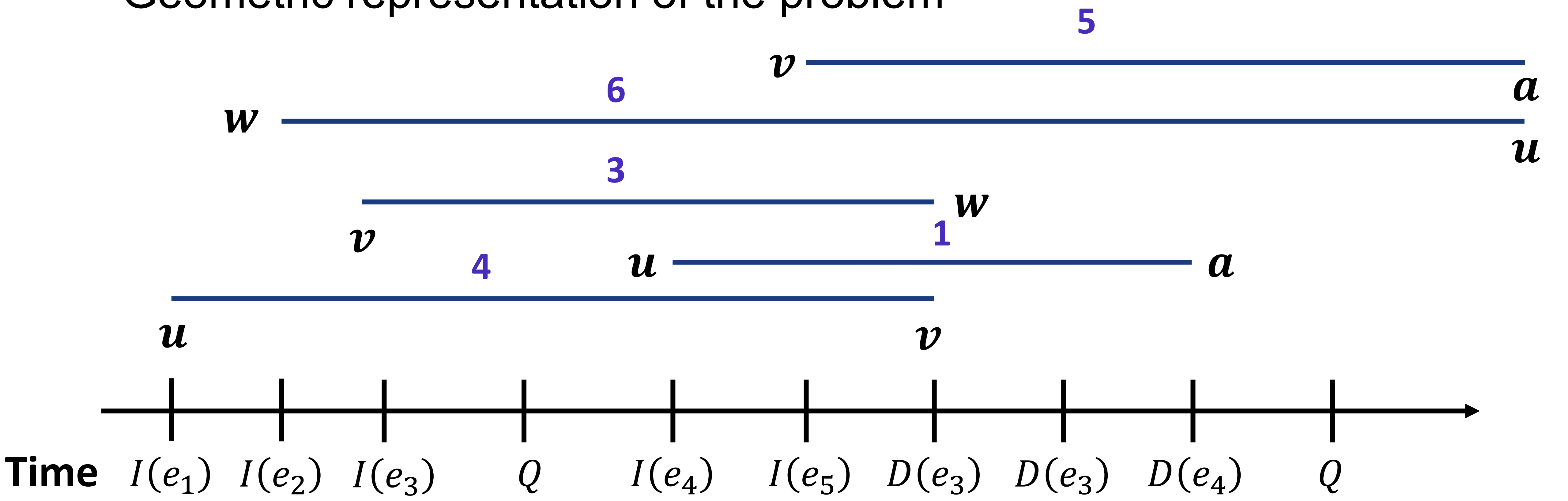
- We will use the offline dynamic minimum spanning tree algorithm of Eppstein '92

Offline-Dynamic Minimum Spanning Tree

- We will use the offline dynamic minimum spanning tree algorithm of Eppstein ‘92
- Geometric representation of the problem

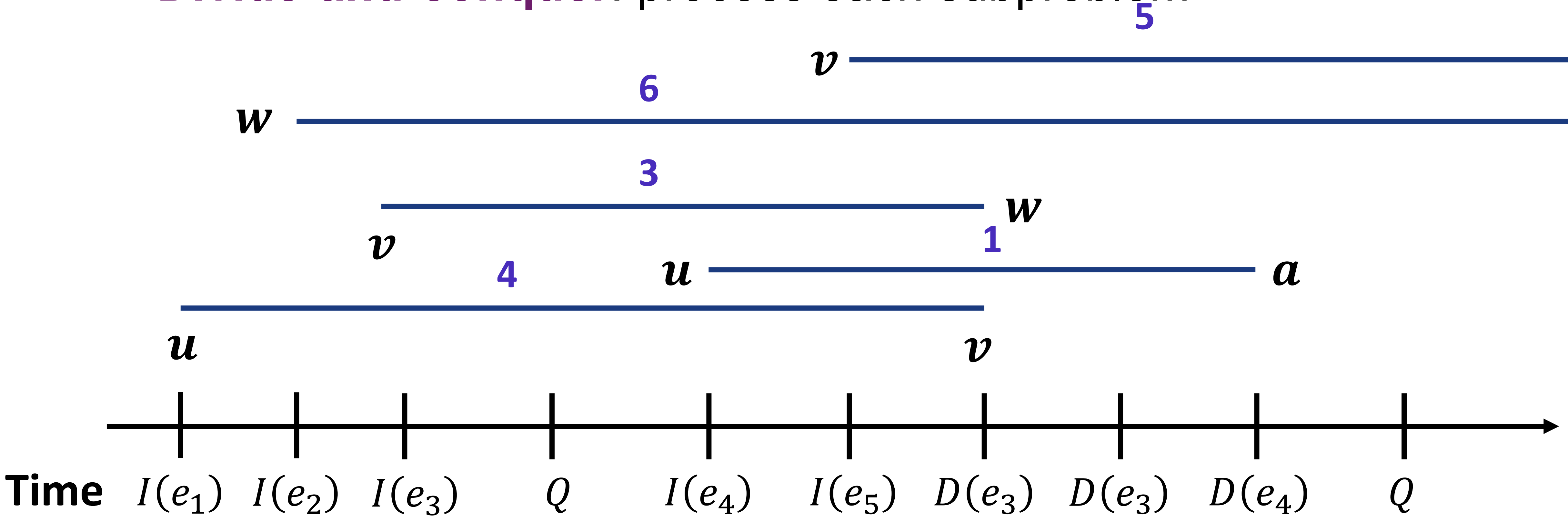
Offline-Dynamic Minimum Spanning Tree

- We will use the offline dynamic minimum spanning tree algorithm of Eppstein '92
- Geometric representation of the problem



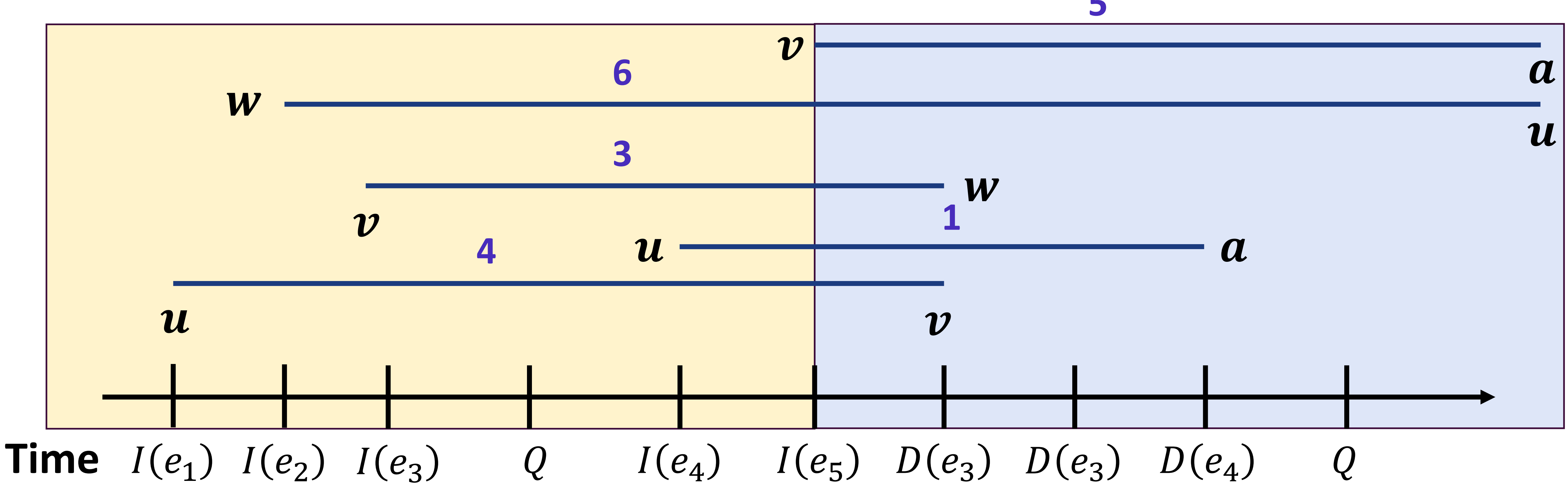
Offline-Dynamic Minimum Spanning Tree

- Geometric representation of the problem
- **Divide-and-conquer**: process each subproblem



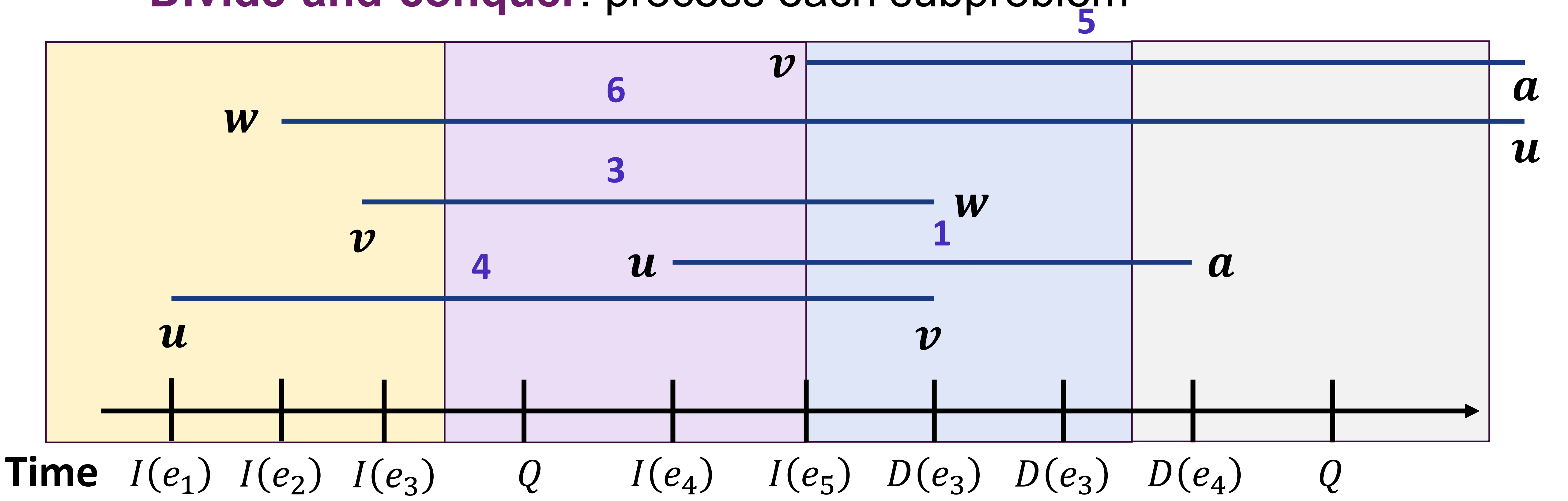
Offline-Dynamic Minimum Spanning Tree

- Geometric representation of the problem
- **Divide-and-conquer**: process each subproblem



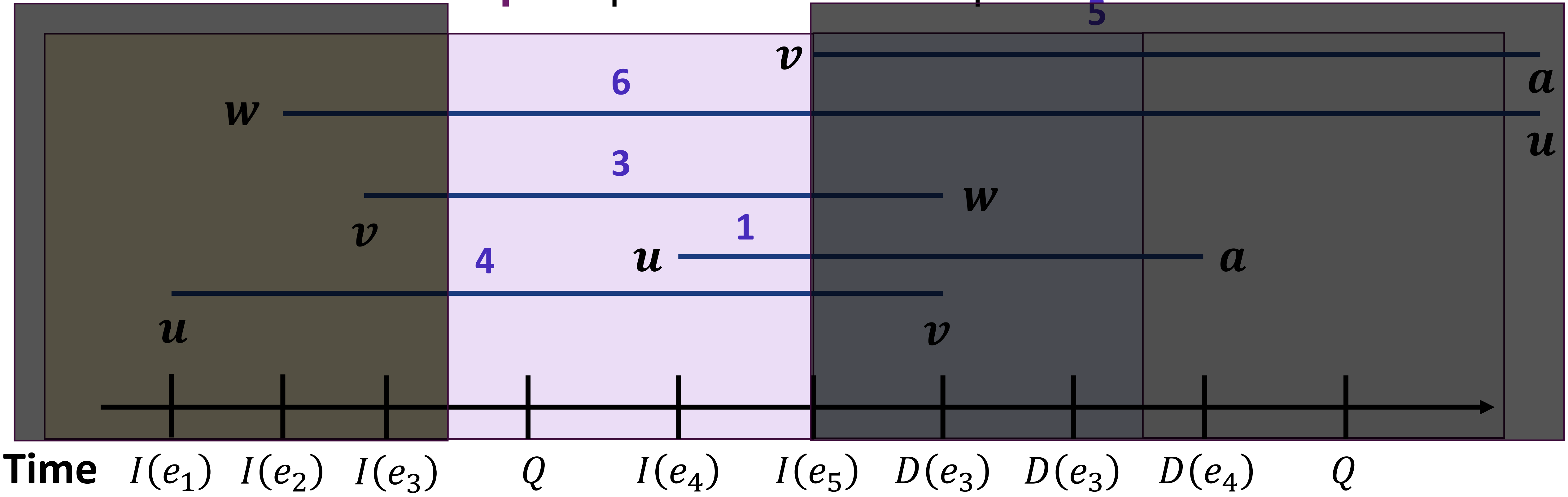
Offline-Dynamic Minimum Spanning Tree

- Geometric representation of the problem
- **Divide-and-conquer**: process each subproblem



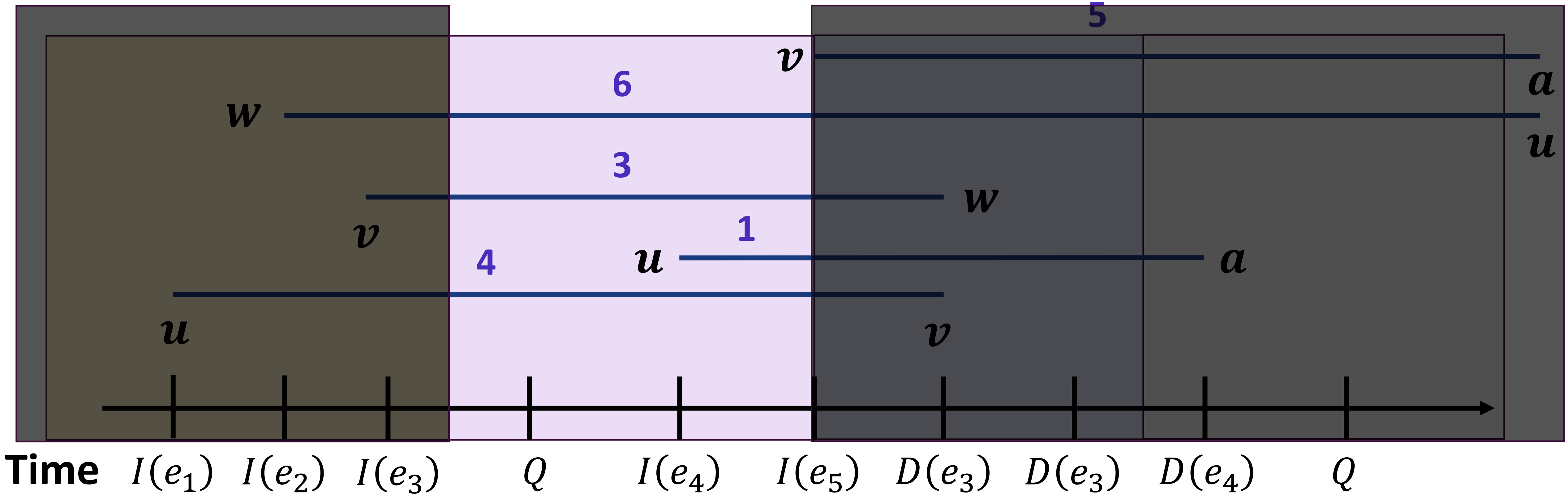
Offline-Dynamic Minimum Spanning Tree

- Geometric representation of the problem
- **Divide-and-conquer**: process each subproblem



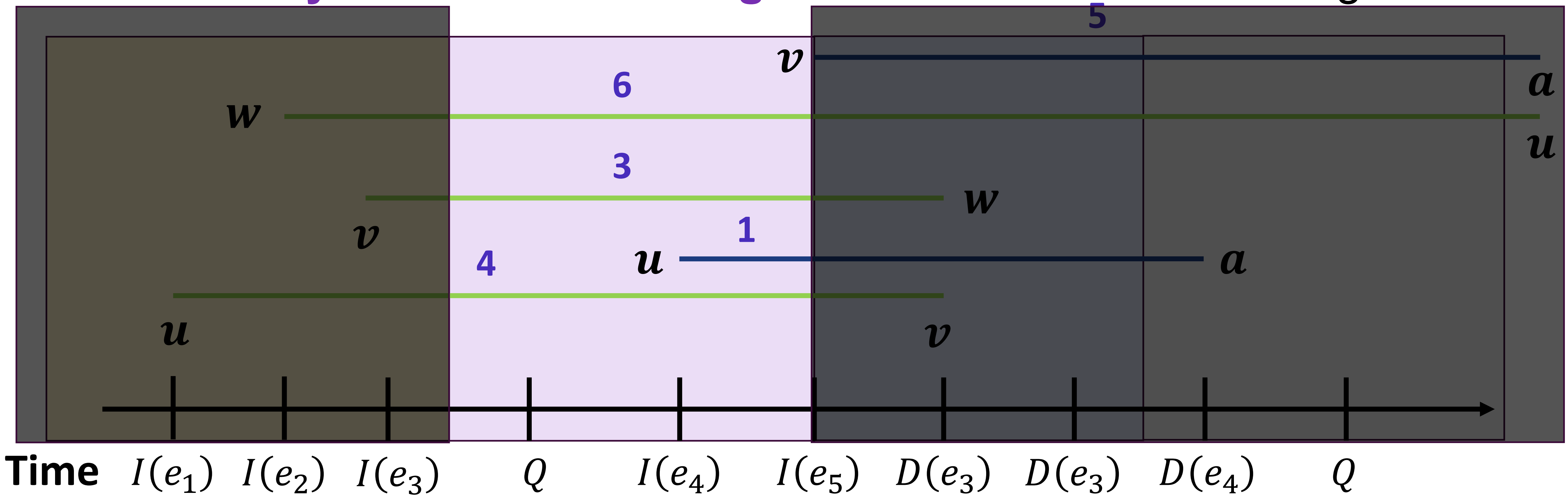
Offline-Dynamic Minimum Spanning Tree

- First, consider all **permanent edges** (edges that go **across the subproblem**)



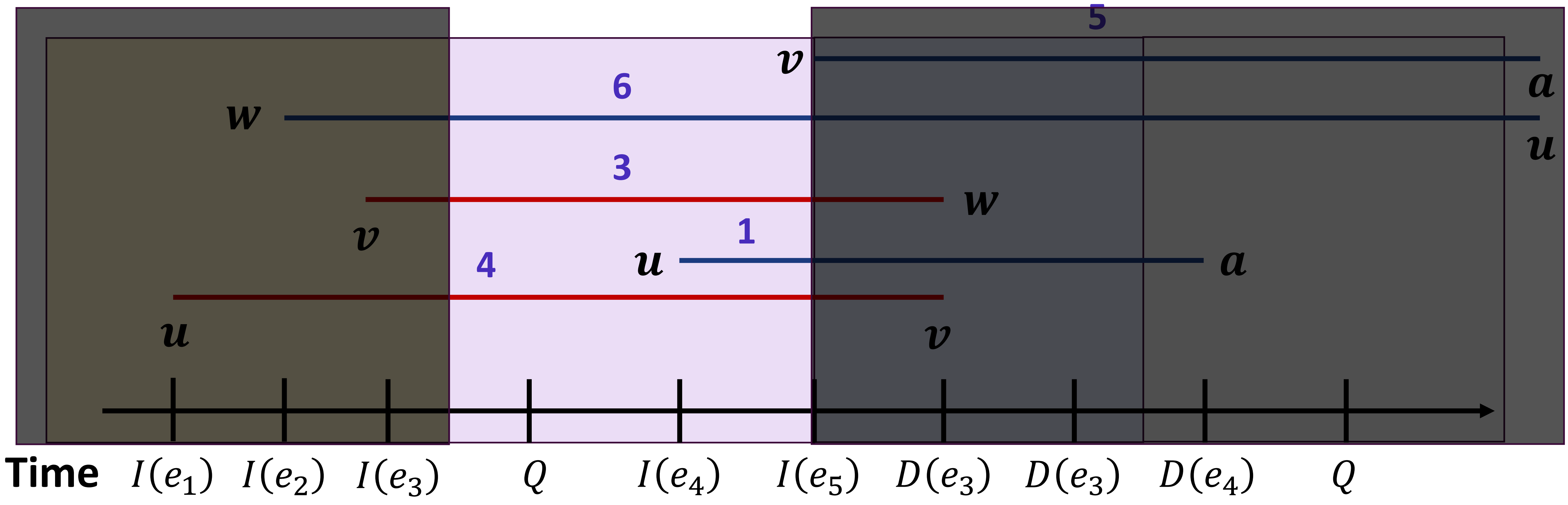
Offline-Dynamic Minimum Spanning Tree

- First, consider all **permanent edges** (edges that **go across the subproblem**)
- Run **any linear time MST algorithm** on all considered edges



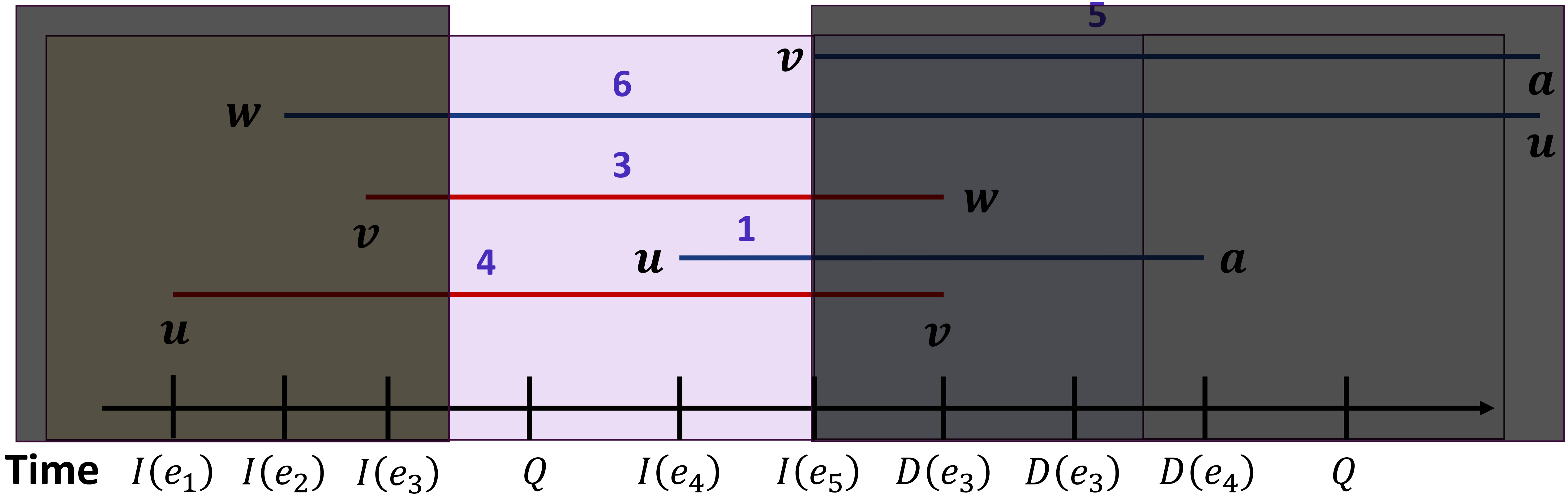
Offline-Dynamic Minimum Spanning Tree

- Run **any linear time MST algorithm** on all considered edges



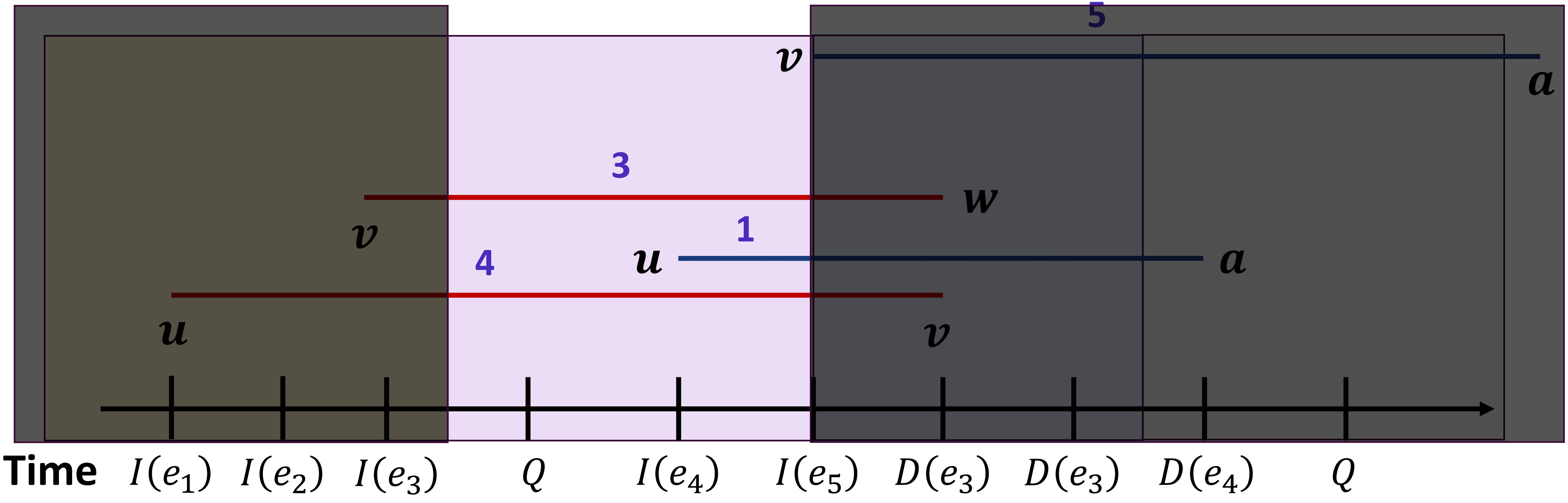
Offline-Dynamic Minimum Spanning Tree

- Run **any linear time MST algorithm** on all considered edges
- **Red edges** are in the MST



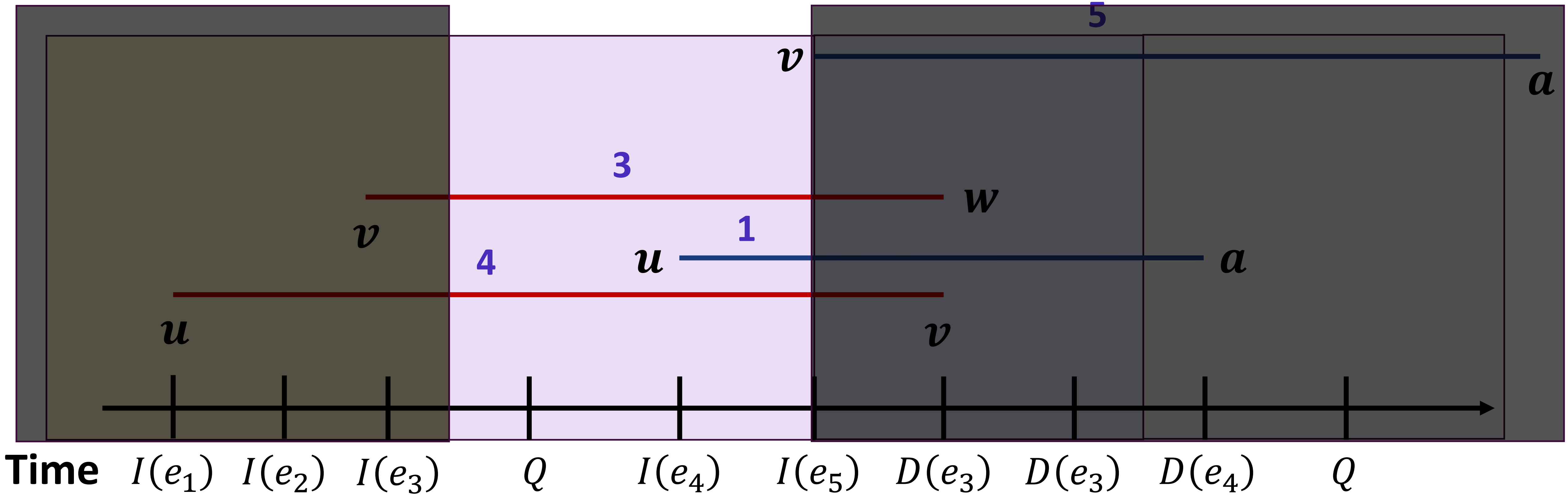
Offline-Dynamic Minimum Spanning Tree

- Run **any linear time MST algorithm** on all considered edges
- **Red edges** are in the MST; **Delete permanent edges not red**



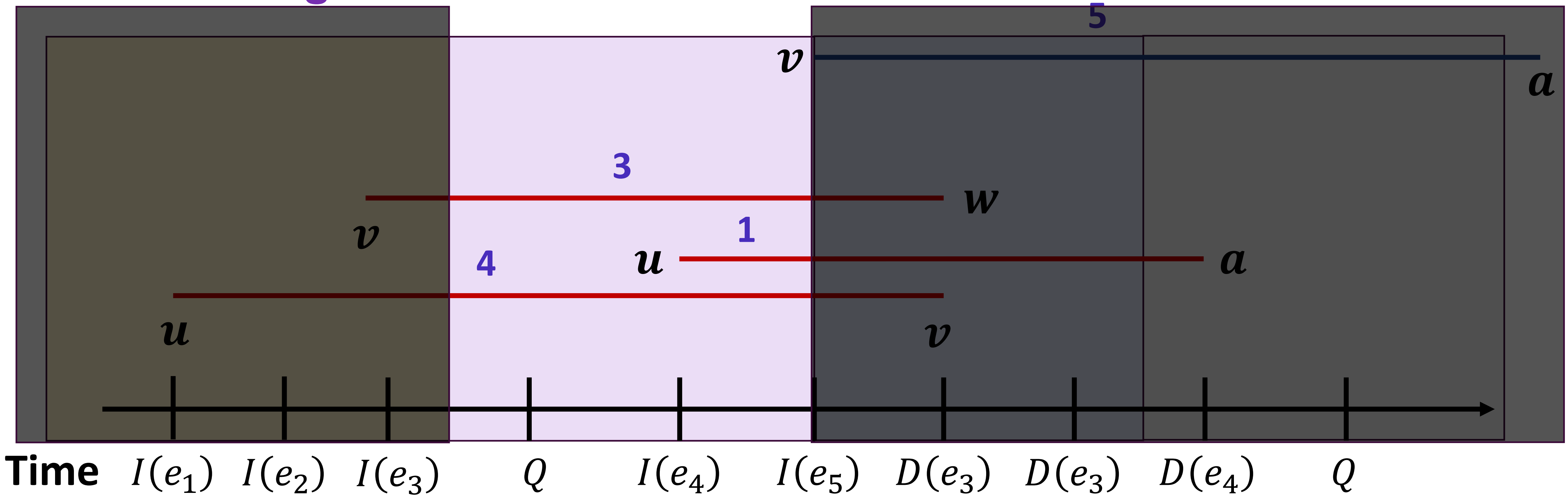
Offline-Dynamic Minimum Spanning Tree

- **Red edges** are in the MST; **Delete permanent edges not red**
- **Now consider all edges in subproblem**



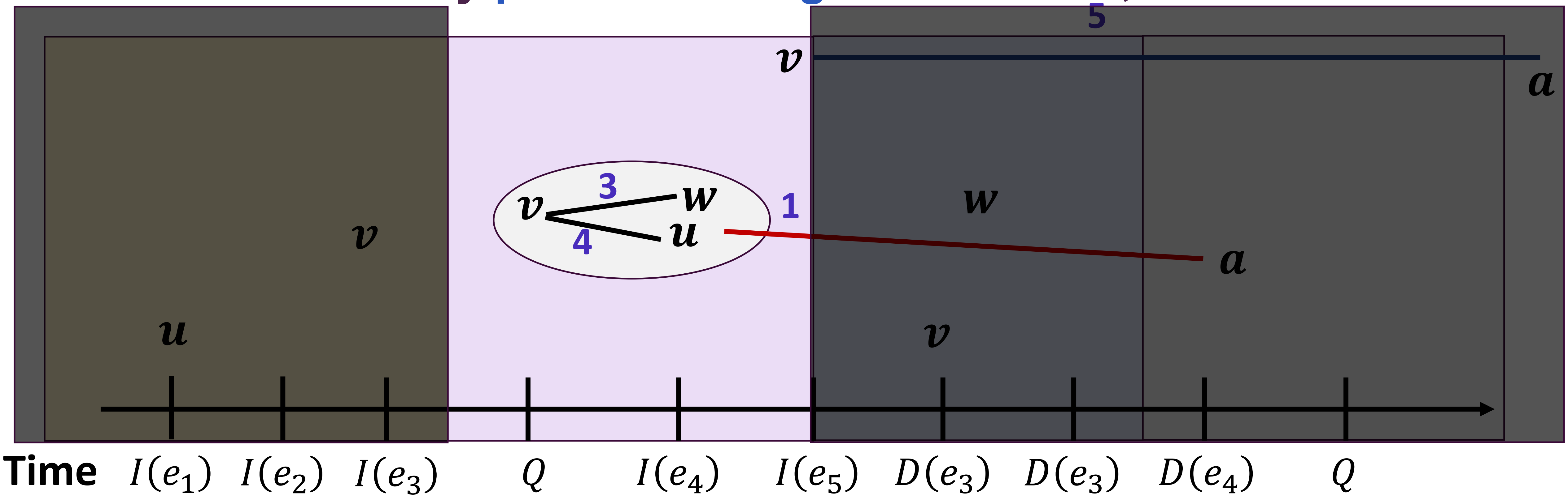
Offline-Dynamic Minimum Spanning Tree

- **Red edges** are in the MST; **Delete permanent edges not red**
- **Now consider all edges in subproblem**. Run **any linear time MST algorithm**



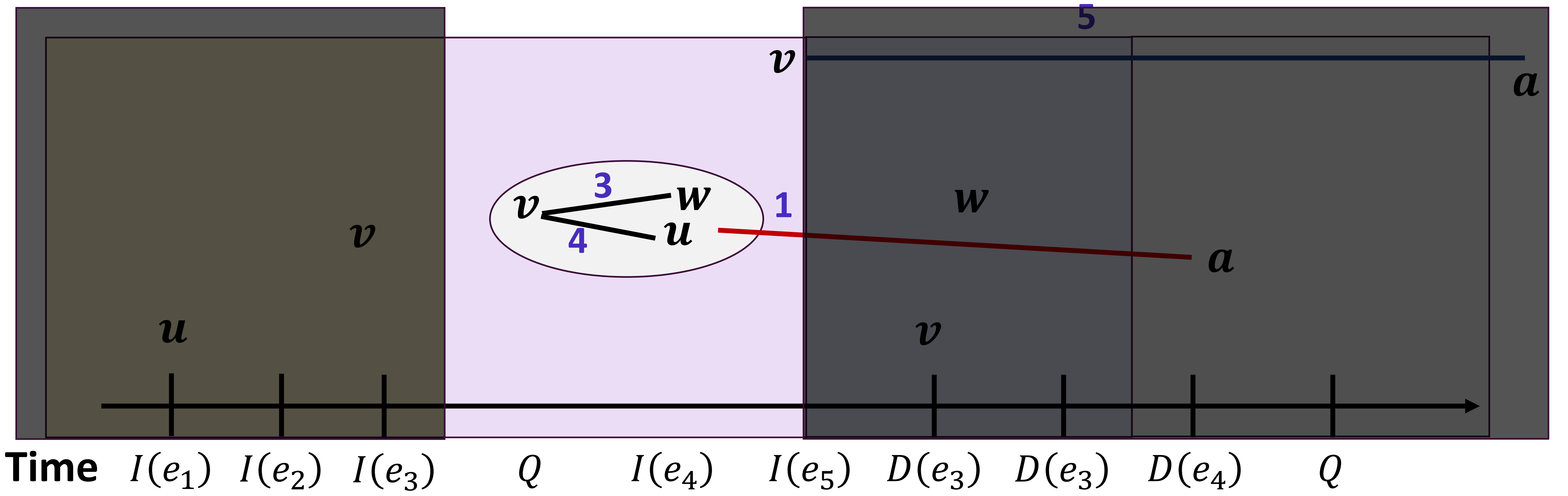
Offline-Dynamic Minimum Spanning Tree

- **Now consider all edges in subproblem**. Run **any linear time MST algorithm**
- **"Contract" any permanent edges in the MST;** link-cut tree



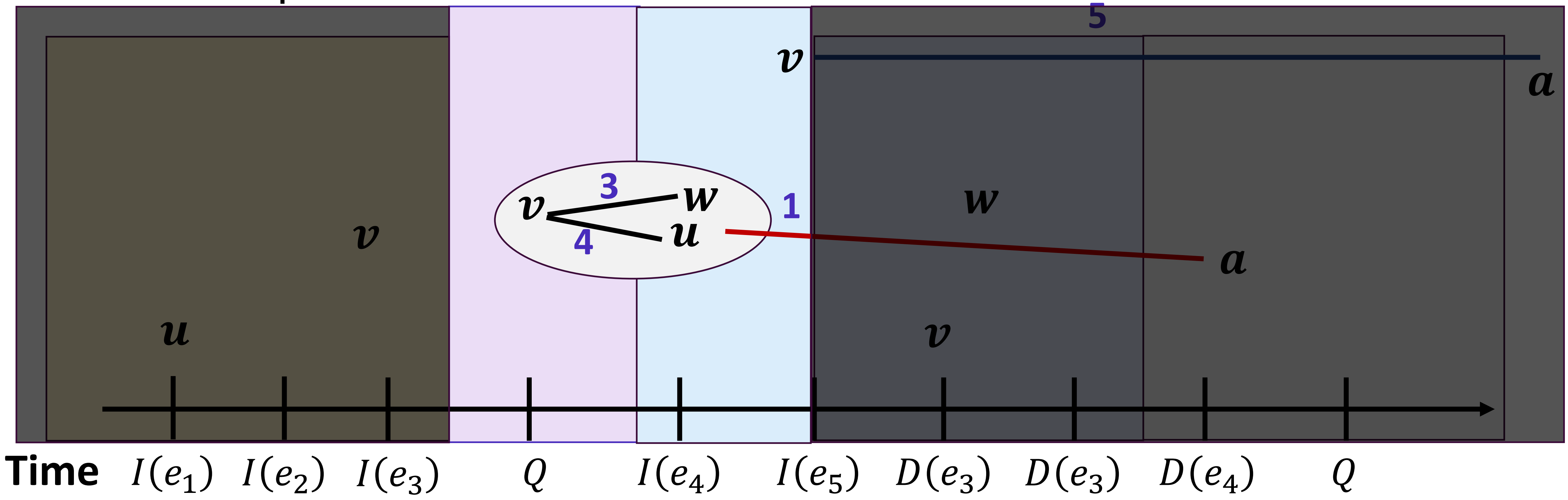
Offline-Dynamic Minimum Spanning Tree

- **Pass data structure to next smaller subproblem (persistence)**



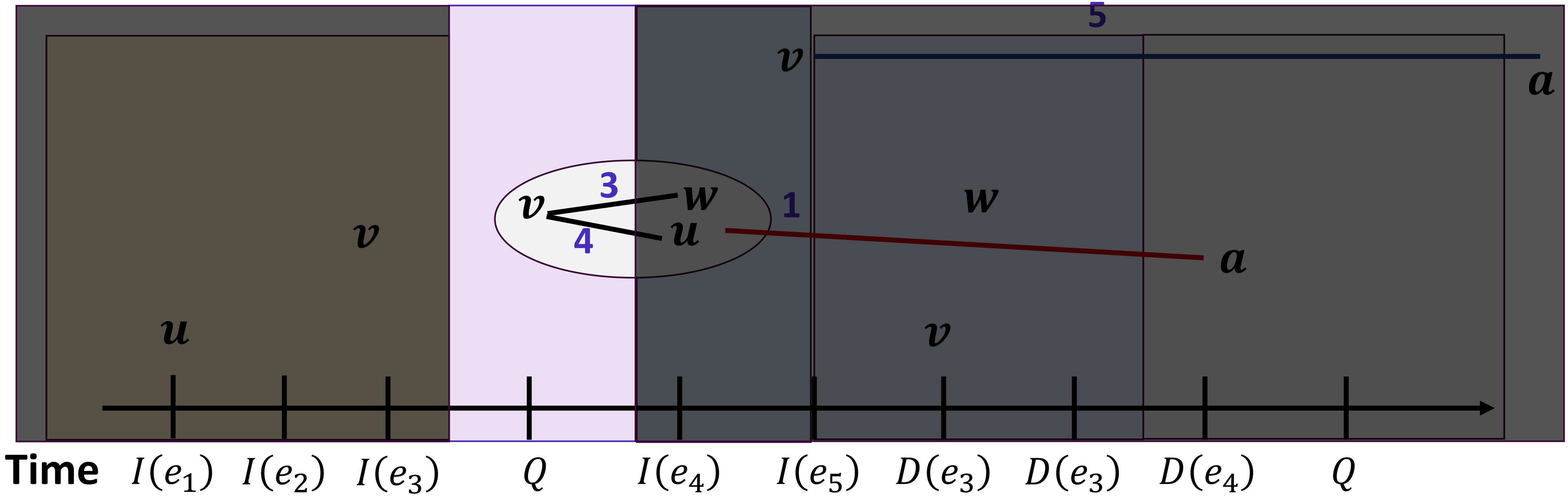
Offline-Dynamic Minimum Spanning Tree

- Pass data structure to next smaller subproblem (persistence)
- Consider **non-contracted** and **not deleted edges permanent** in subproblem



Offline-Dynamic Minimum Spanning Tree

- **Queries**: consider tree at the **smallest subproblem containing the query Q**



Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,** or **neither**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,** or **neither**
  - **Deleted and contracted edges charged to previous level**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with T total permanent and non-permanent edges takes $\tilde{O}(T)$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,** or **neither**
  - **Deleted and contracted edges charged to previous level**
  - Each permanent edge not deleted **charged to unique current level non-permanent edge**

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes $\tilde{O}(\boldsymbol{T})$ **time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,** or **neither**
  - **Deleted and contracted edges charged to previous level**
  - Each permanent edge not deleted **charged to unique current level non-permanent edge**
- Thus, $3T$ charges in each subproblem with T updates

Offline-Dynamic Minimum Spanning Tree

- Assume link-cut tree and persistence such that each subproblem with $\boldsymbol{T}$ total permanent and non-permanent edges takes **$\tilde{O}(T)$ time**
- First, consider permanent edges, what happens to them?
  - They are either **deleted, contracted,** or **neither**
  - **Deleted and contracted edges charged to previous level**
  - Each permanent edge not deleted **charged to unique current level non-permanent edge**
- Thus, $3T$ charges in each subproblem with T updates

Total Runtime: $\tilde{O}(T \log(T))$ by Master Theorem

Offline-to-Fully Dynamic

- Learning-augmented minimum spanning tree:

Offline-to-Fully Dynamic

- Learning-augmented minimum spanning tree:
  - **Predicted sequence**:
    - All updates and query **times** are given as **predictions**

Offline-to-Fully Dynamic

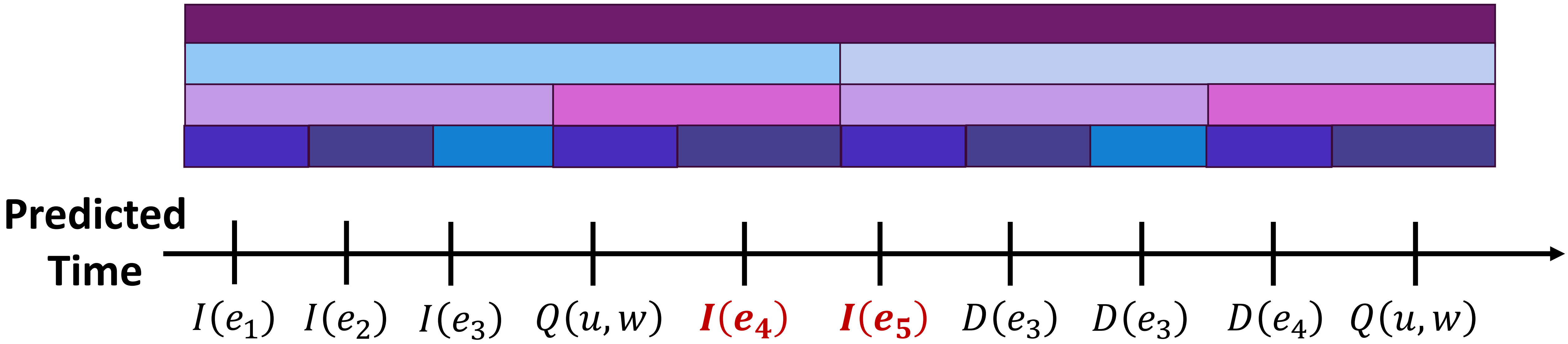
- Learning-augmented minimum spanning tree:
  - **Predicted sequence**:
    - All updates and query **times** are given as **predictions**
    - Can give the predictions in batches
      - Will not discuss today

Offline-to-Fully Dynamic

- Learning-augmented minimum spanning tree:
  - **Predicted sequence**:
    - All updates and query **times** are given as **predictions**
    - Can give the predictions in batches
      - Will not discuss today
  - **Key Problem:** deterministic division of divide-and-conquer tree can lead to **arbitrarily bad runtime**

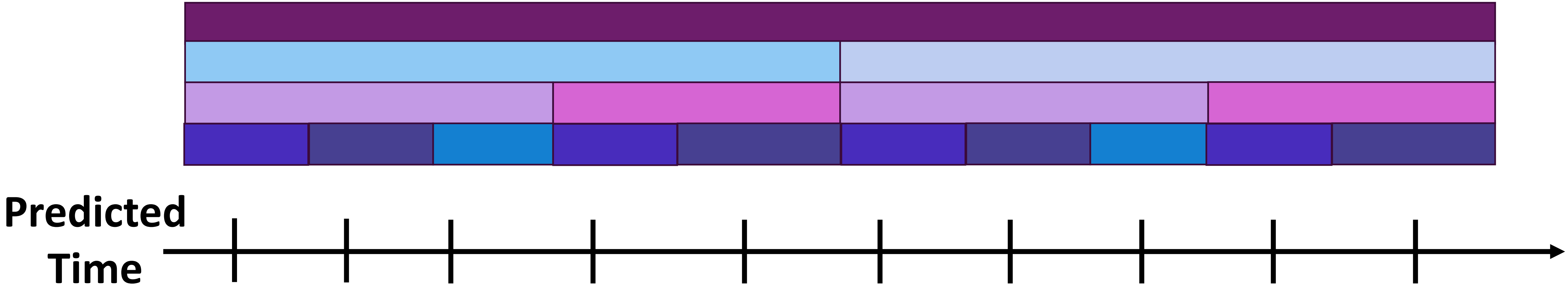
Offline to Online **First Attempt**

Use Offline Divide-and-Conquer Algorithm on Predicted Sequence



Offline to Online **First Attempt**

Use Offline Divide-and-Conquer Algorithm on Predicted Sequence



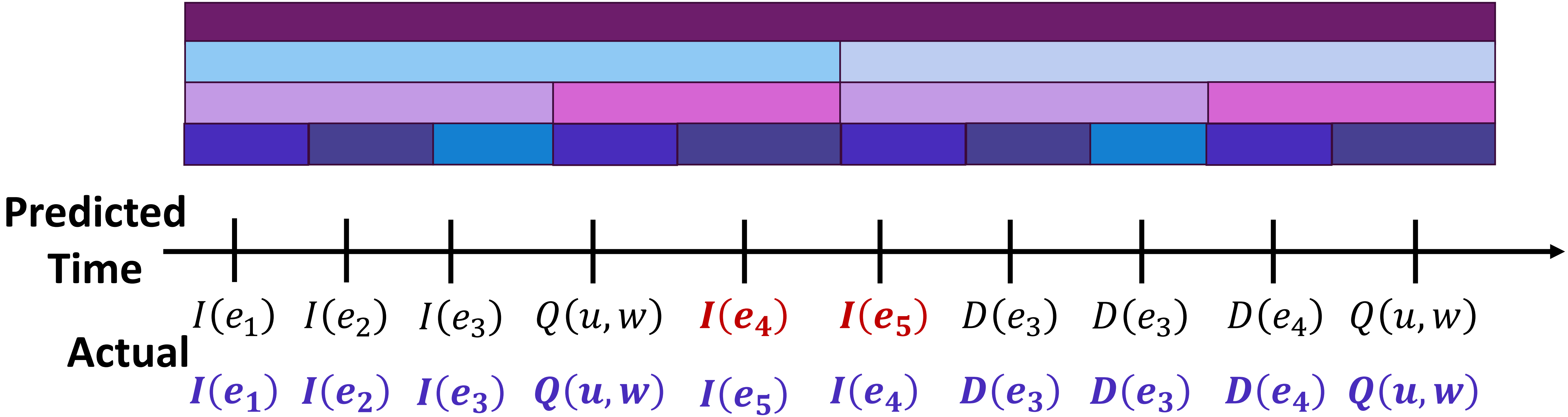
Predicted: $I(e_1)$ $I(e_2)$ $I(e_3)$ $Q(u,w)$ $\boldsymbol{I(e_4)}$ $\boldsymbol{I(e_5)}$ $D(e_3)$ $D(e_3)$ $D(e_4)$ $Q(u,w)$

Actual

$\boldsymbol{I(e_1)}$ $\boldsymbol{I(e_2)}$ $\boldsymbol{I(e_3)}$ $\boldsymbol{Q(u,w)}$ $\boldsymbol{I(e_5)}$ $\boldsymbol{I(e_4)}$ $\boldsymbol{D(e_3)}$ $\boldsymbol{D(e_3)}$ $\boldsymbol{D(e_4)}$ $\boldsymbol{Q(u,w)}$

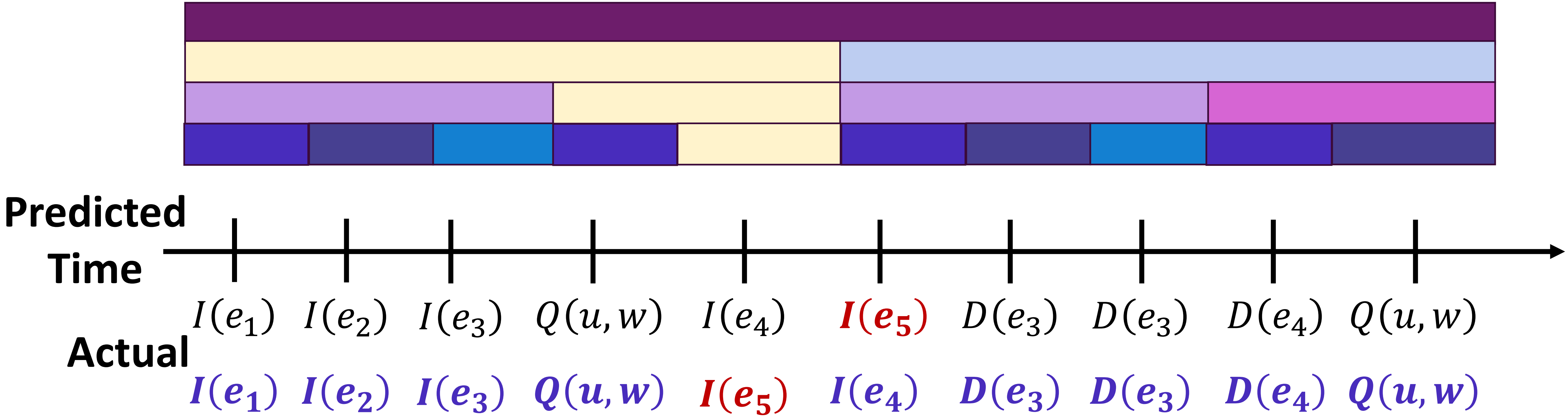
Offline to Online **First Attempt**

Fix all subtrees up to largest subtree containing error as (i.e. redo subtrees containing $\boldsymbol{I(e_5)}$ as permanent edge; it becomes non-permanent)



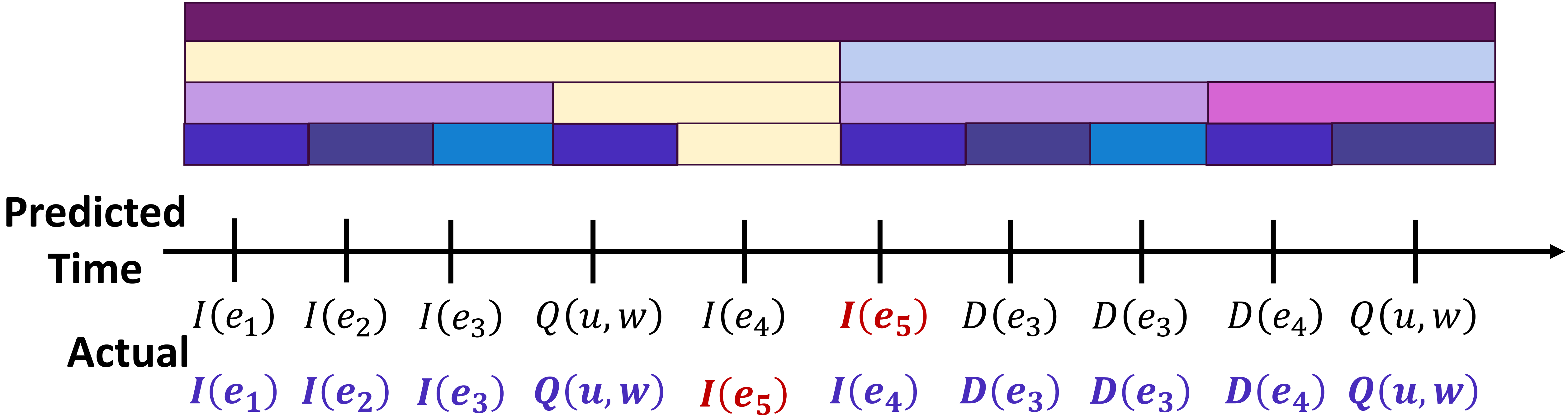
Offline to Online **First Attempt**

Fix all subtrees up to largest subtree containing error as (i.e. redo subtrees containing $\boldsymbol{I(e_5)}$ as permanent edge; it becomes non-permanent)



Offline to Online **First Attempt**

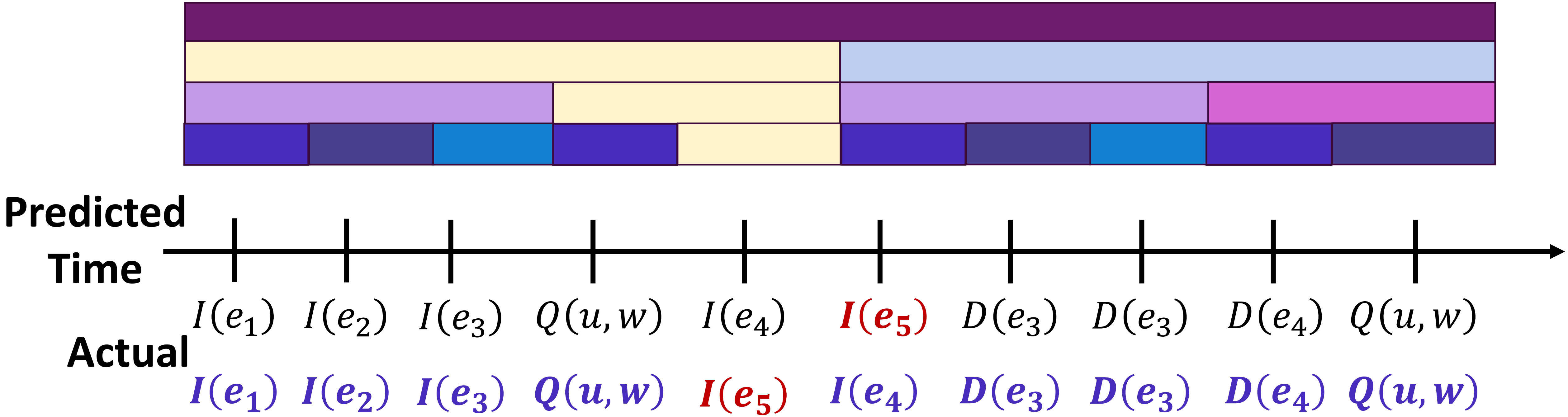
Issue: large subtree divides predicted and real timestamps



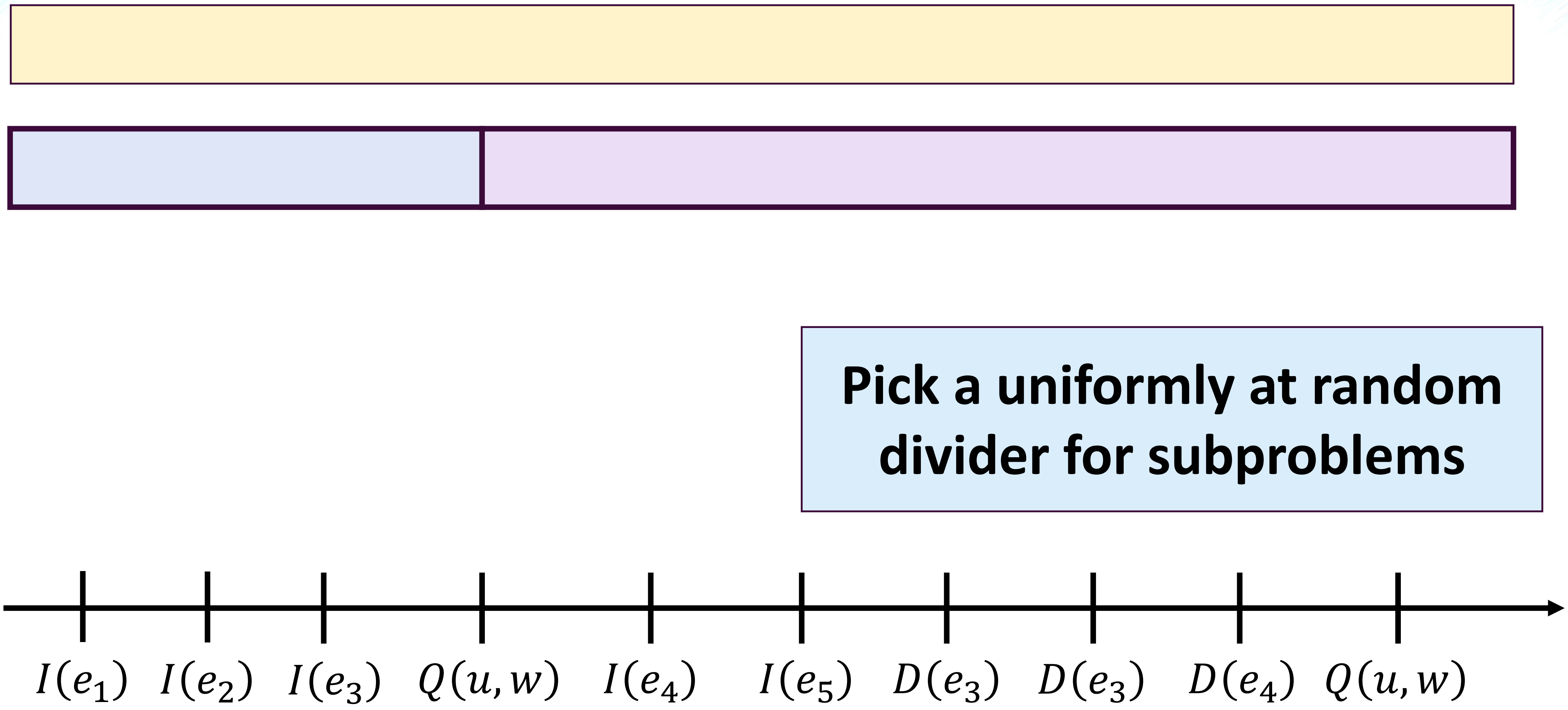
Offline to Online **First Attempt:** **Failed**

Issue: large subtree divides predicted and real timestamps

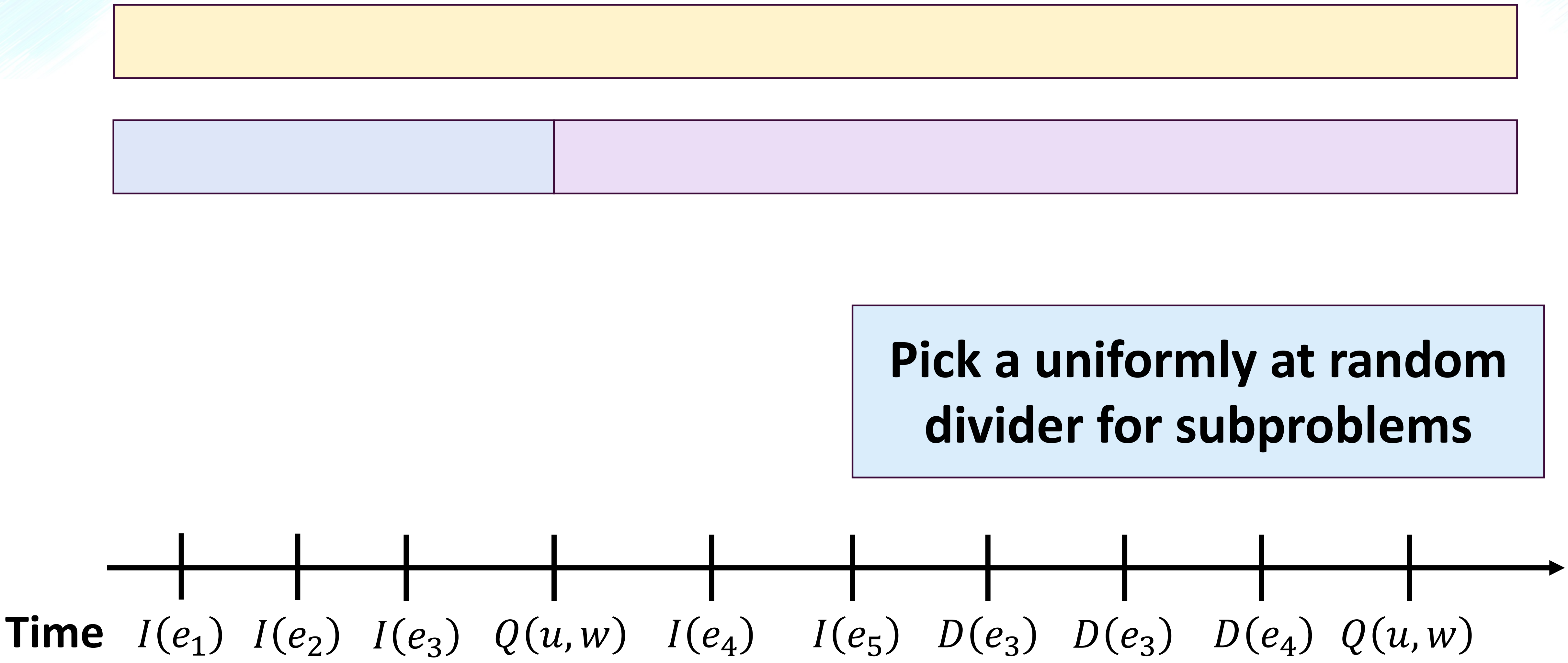
L_1 Error: **1**; Update time: $\boldsymbol{O(n)}$



Random Partition Tree Data Structure

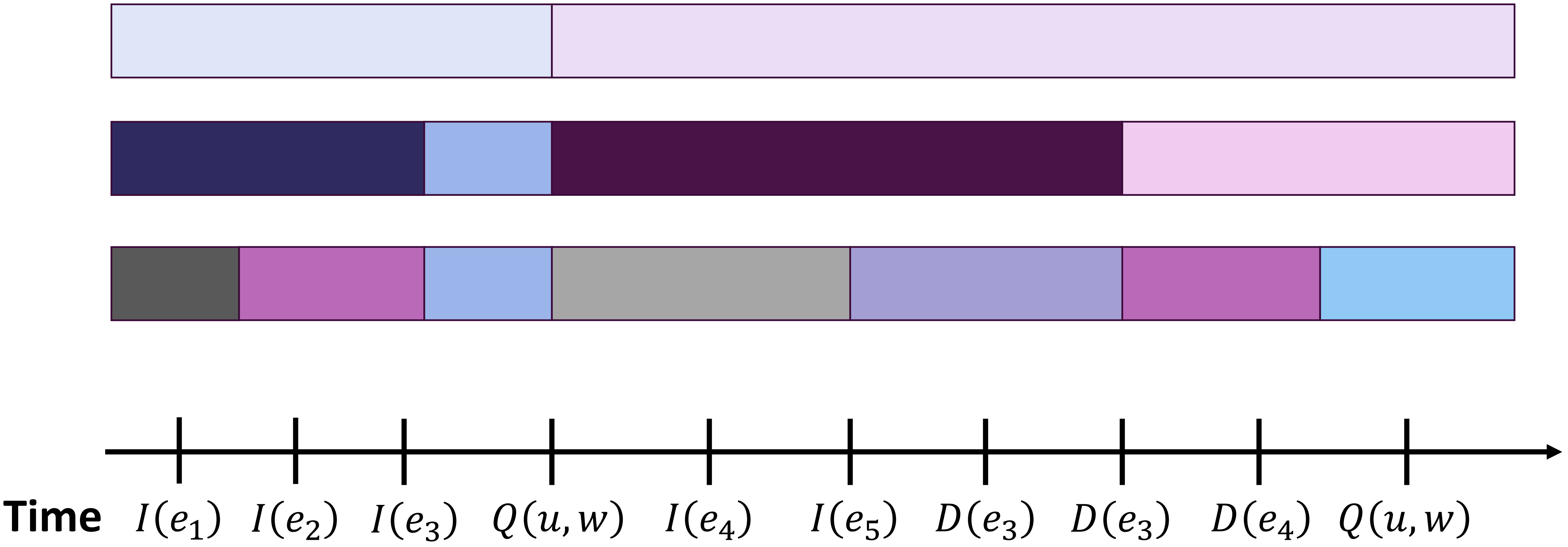


Random Partition Tree Data Structure



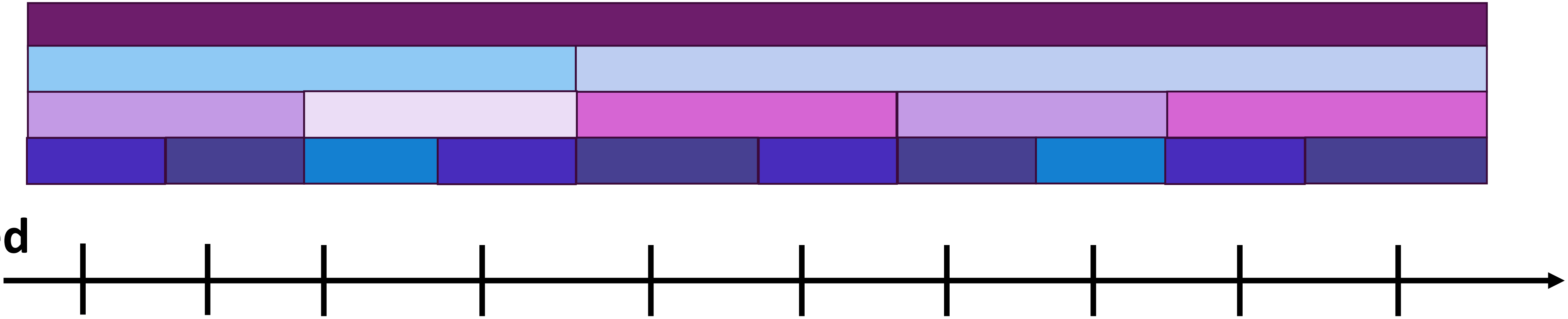
Random Partition Tree Data Structure

Run offline divide-and-conquer algorithm on randomly picked subproblems



Random Partition Tree Data Structure

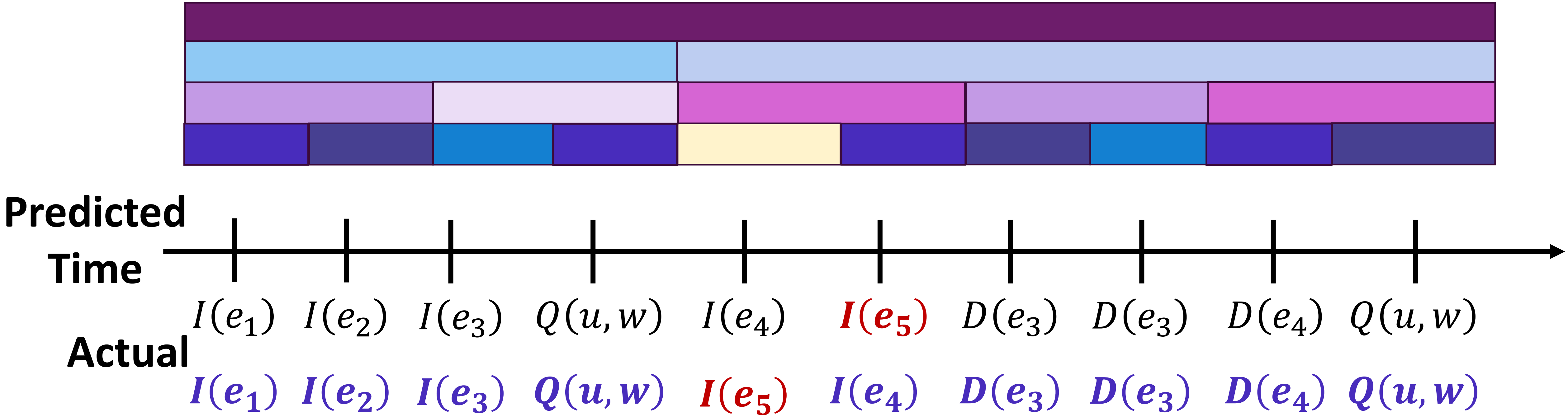
Run offline divide-and-conquer algorithm on randomly picked subproblems



Predicted	$I(e_1)$	$I(e_2)$	$I(e_3)$	$Q(u,w)$	$\boldsymbol{I(e_4)}$	$\boldsymbol{I(e_5)}$	$D(e_3)$	$D(e_3)$	$D(e_4)$	$Q(u,w)$
Actual	$\boldsymbol{I(e_1)}$	$\boldsymbol{I(e_2)}$	$\boldsymbol{I(e_3)}$	$\boldsymbol{Q(u,w)}$	$\boldsymbol{I(e_5)}$	$\boldsymbol{I(e_4)}$	$\boldsymbol{D(e_3)}$	$\boldsymbol{D(e_3)}$	$\boldsymbol{D(e_4)}$	$\boldsymbol{Q(u,w)}$

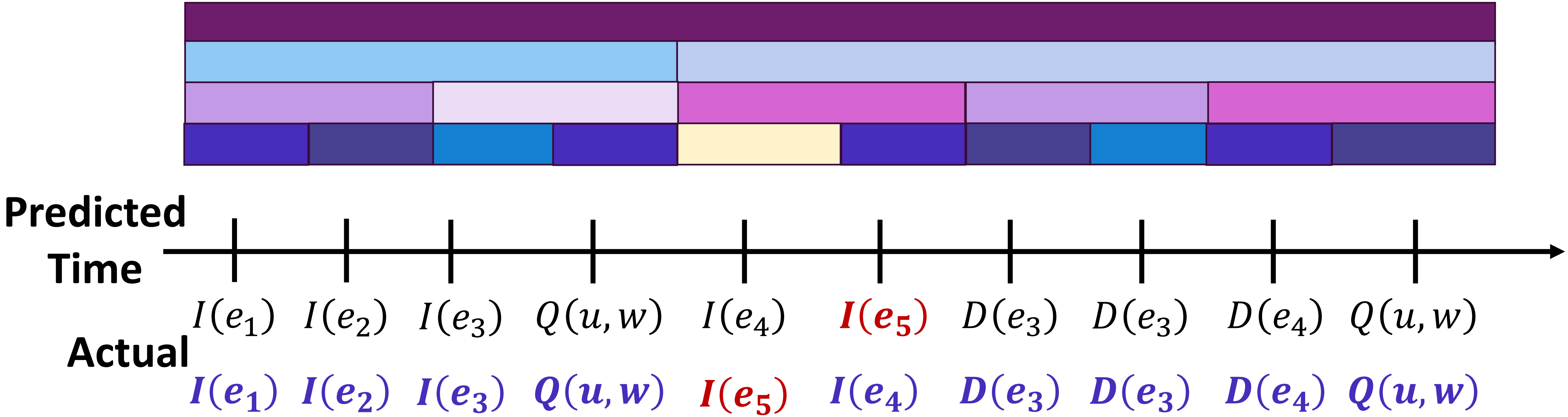
Random Partition Tree Data Structure

Run offline divide-and-conquer algorithm on randomly picked subproblems



Random Partition Tree Data Structure

Purpose of the random partition tree: in expectation size of subproblem (largest subtree going in between) equal to L_1 error



Random Partition Tree Data Structure

Purpose of the random partition tree: in expectation size of subproblem (smallest subtree) equal to L_1 error

Proof: Coupling argument for splitting subproblems to drawing dividers:

1. For each divider between two timestamps in the original sequence, draw value (called rank) in $[0, 1]$

Random Partition Tree Data Structure

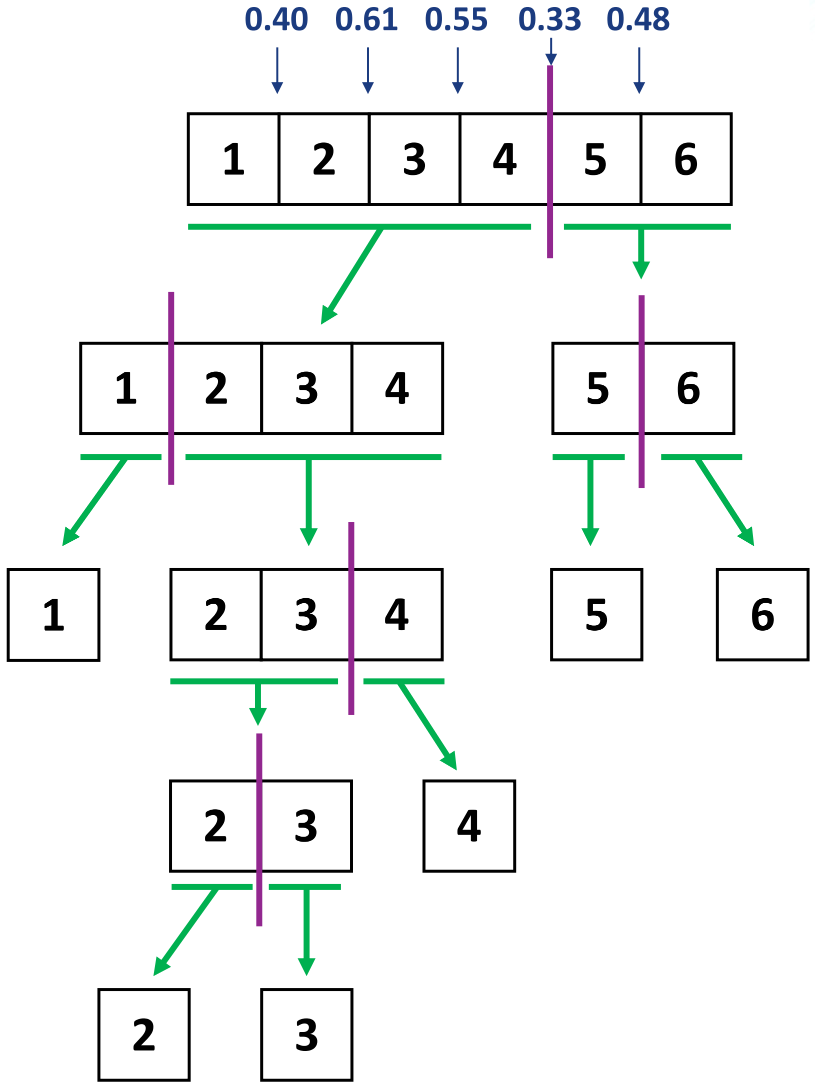
Purpose of the random partition tree: in expectation size of subproblem (smallest subtree) equal to L_1 error

Proof: Coupling argument for splitting subproblems to drawing dividers:

1. For each divider between two timestamps in the original sequence, draw value (called rank) in $[0, 1]$
2. Divide sequence of updates lowest rank first

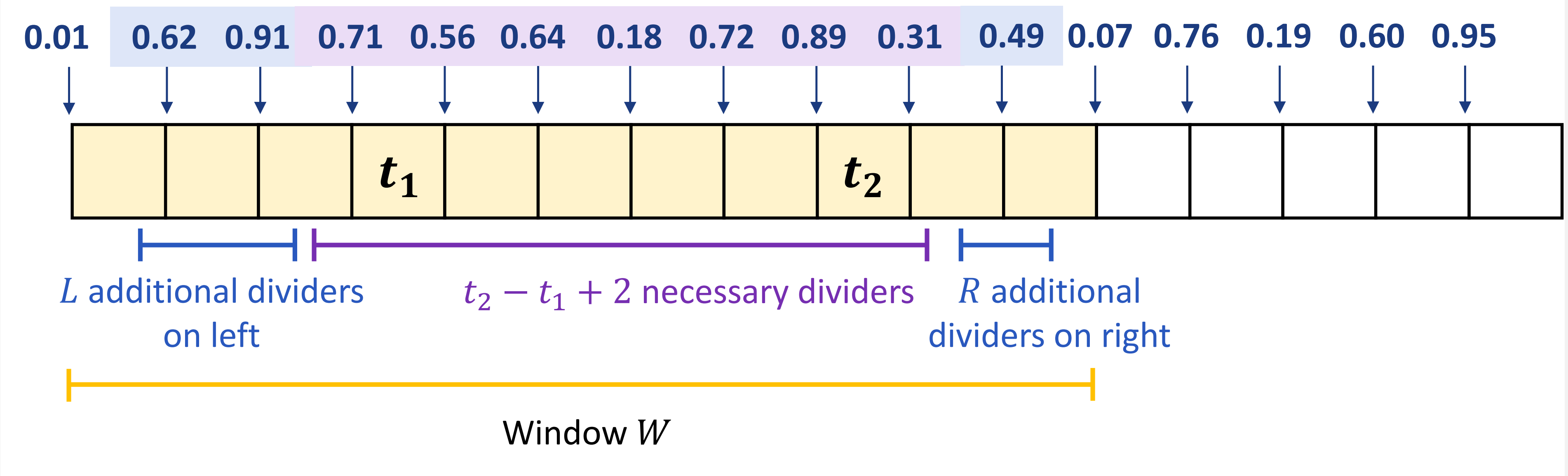
Random Partition Tree Data Structure

Example Random Tree produced via drawing dividers



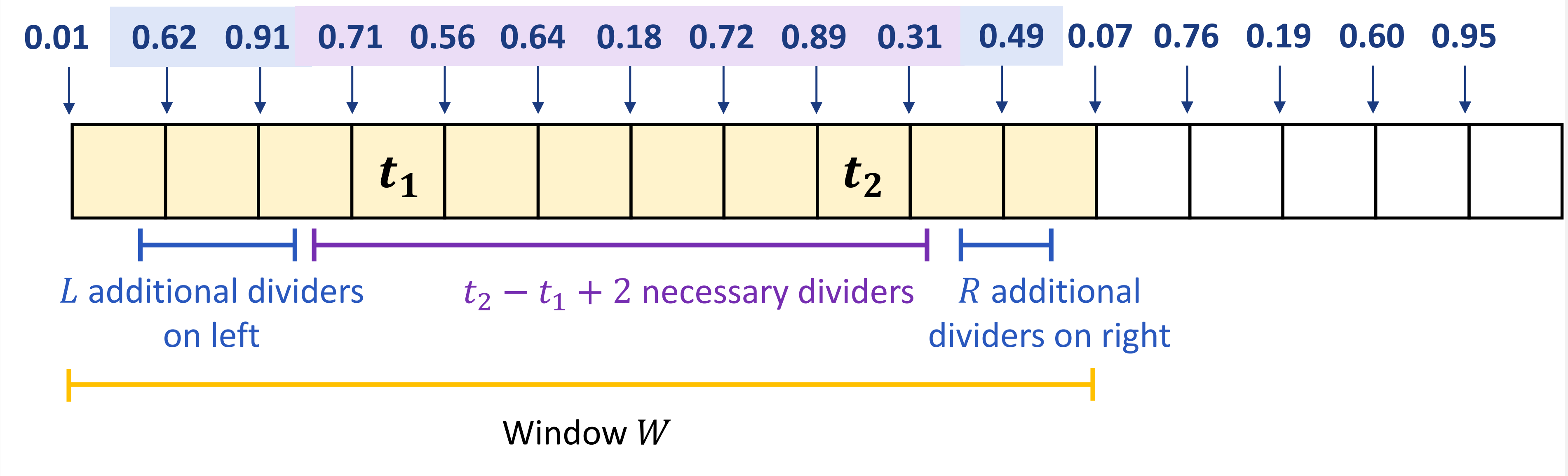
Random Partition Tree Data Structure

Consider an error that occurred at times t_1, t_2



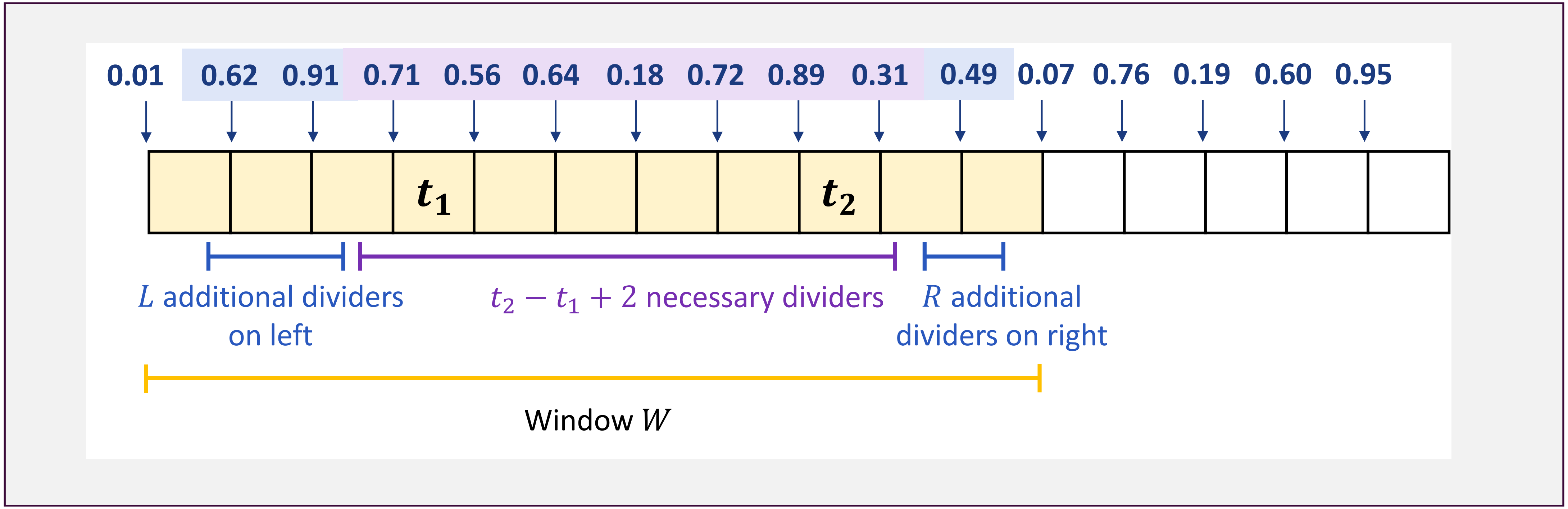
Random Partition Tree Data Structure

Expected window W size is equal to $L + R + (t_2 - t_1 + 1)$



Random Partition Tree Data Structure

$$\mathbf{E}[W] = O(|t_2 - t_1| \cdot \log(T))$$



Predicted-Updates Result

$$\mathbf{E}[W] = O\left(|t_2 - t_1| \cdot \log(T)\right)$$

Expected Work per error t_1, t_2: $\tilde{O}(|t_2 - t_1|)$

for each recomputation over window $[t_1, t_2]$ (leaf dominated divide-and-conquer)

Predicted-Updates Result

$$\mathbf{E}[W] = O\big(|t_2 - t_1| \cdot \log(T)\big)$$

Expected Work per error t_1, t_2: $\tilde{O}(|t_2 - t_1|)$

Total expected work: $\tilde{O}(|\mathbf{p} - \mathbf{r}|_1)$

Boosting to high probability via $O(\log n)$ independent trials

Predicted-Updates Result

- Runtime in terms of L_1-error between predictions and real
  - $\mathbf{p}$ vector of **predicted** timestamps
  - $\mathbf{r}$ vector of **real** timestamps
  - L_1 error: $|\mathbf{p} - \mathbf{r}|_1$

$\boldsymbol{update}$ is **worst-case update time** of **incremental/decremental** algorithm

Runtime same as partially dynamic with total

$$\tilde{O}\left(\left(||\mathbf{p} - \mathbf{r}||_1 + T\right) \cdot \boldsymbol{update}\right)$$

time for T updates

[L-Srinivas COLT '24]

The Predicted-Updates Model

	Best Fully Dynamic		Best Predicted-Updates when $\|\|\mathbf{p}-\mathbf{r}\|\|_1 = \tilde{O}(T)$	
Planar Digraph APSP	$\tilde{O}\left(n^{2/3}\right)$	[FR06, Kle05]	$\tilde{O}(\sqrt{n})$	[DGWN22]
Triconnectivity	$O(n^{2/3})$	[GIS99]	$\tilde{O}(1)$	[HR20, PSS17]
k-Edge Connectivity	$n^{o(1)}$	[JS22]	$\tilde{O}(1)$	[CDK+21]
APSP	$\left(\frac{256}{k^2}\right)^{4/k}$-Approx $\tilde{O}\left(n^k\right)$ update $\tilde{O}(n^{k/8})$ query	[FGNS23]	$(2r-1)^k$-Approx $\tilde{O}\left(m^{1/(k+1)}n^{k/r}\right)$	[CGH+20]
AP Maxflow/Mincut	$O(\log(n)\log\log n)$-Approx $\tilde{O}\left(n^{2/3+o(1)}\right)$	[CGH+20]	$O\left(\log^{8k}(n)\right)$-Approx. $\tilde{O}\left(n^{2/(k+1)}\right)$	[Gor19, GHS19]
MCF	$(1+\varepsilon)$-Approx $\tilde{O}(1)$ update $\tilde{O}(n)$ query	[CGH+20]	$O(\log^{8k}(n))$-Approx. $\tilde{O}\left(n^{2/(k+1)}\right)$ update $\tilde{O}(P^2)$ query	[Gor19, GHS19]
Uniform Sparsest Cut	$2^{O(\log^{5/6}(n))}$-Approx $2^{O(\log^{5/6}(n))}$ update $O(\log^{1/6}(n))$ query	[GRST21]	$O\left(\log^{8k}(n)\right)$-Approx $\tilde{O}\left(n^{2/(k+1)}\right)$ $O(1)$ query	[Gor19, GHS19]
Submodular Max	$1/4$-Approx $\tilde{O}(k^2)$	[DFL+23]	0.3178-Approx $\tilde{O}(\text{poly}(k))$	[FLN+22]

Conclusion

Link Prediction – Predict edges in a network using Networkx

Last Updated : 08 May, 2020

- **Practicality of link-prediction:**
  - Lots of work on insertion link prediction

RESEARCH ARTICLE | COMPUTER SCIENCES |

Link prediction using low-dimensional node embeddings: The measurement problem

Nicolas Menand and C. Seshadhri Authors Info & Affiliations

Towards Better Evaluation for Dynamic Link Prediction

Farimah Poursafaei*, Shenyang Huang*, Kellin Pelrine, Reihaneh Rabbany
McGill University School of Computer Science, Mila – Quebec AI Institute
[farimah.poursafaei,huangshe,kellin.pelrine,reihaneh.rabbany]@mila.quebec

Learning Spectral Graph Transformations for Link Prediction

Jérôme Kunegis KUNEGIS@DAI-LAB.DE
Andreas Lommatzsch ANDREAS@DAI-LAB.DE
DAI-Labor, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Link prediction

Article Talk

From Wikipedia, the free encyclopedia

Conclusion

- **Practicality of link-prediction:**
  - Lots of work on insertion link prediction
  - **Link deletions** much less well-studied

Conclusion

- **Practicality of link-prediction:**
  - Lots of work on insertion link prediction
  - **Link deletions** much less well-studied
    - Lack of accurate real-world temporal dynamic graph data

Conclusion

- **Practicality of link-prediction:**
  - Lots of work on insertion link prediction
  - **Link deletions** much less well-studied
    - Lack of accurate real-world temporal dynamic graph data
    - Interesting data science problem: obtain data and techniques for link deletion prediction

Conclusion

- **Practicality of link-prediction:**
  - Lots of work on insertion link prediction
  - **Link deletions** much less well-studied
    - Lack of accurate real-world temporal dynamic graph data
    - Interesting data science problem: obtain data and techniques for link deletion prediction

Ongoing Sridharbaskari-Srinivas-**L** ‘24