Stable Tree Labelling for Accelerating Distance Queries on Dynamic Road Networks

Henning Koehler [1] Muhammad Farhan [2] Qing Wang [2]

[2]School of Computing, Australian National University

[1]School of Mathematical and Computational Sciences, Massey University

Australian National University

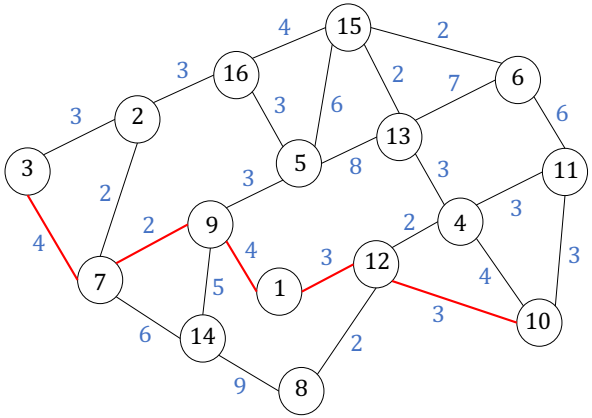
MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Problem Definition

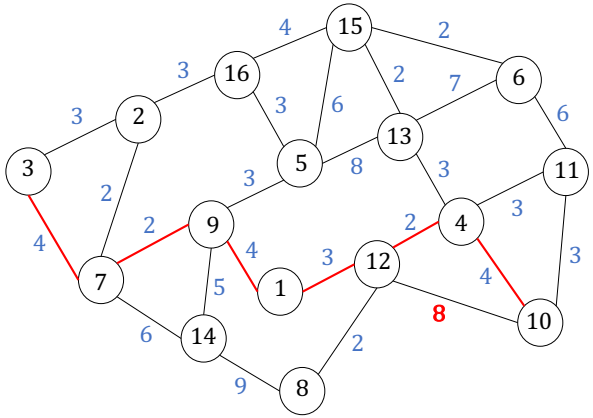
- Given a weighted graph $G = (V, E, \omega)$ that undergo edge weight updates, how to efficiently compute the correct distance of a shortest path between two vertices?

Problem Definition

- Given a weighted graph $G = (V, E, \omega)$ that undergo edge weight updates, how to efficiently compute the correct distance of a shortest path between two vertices?



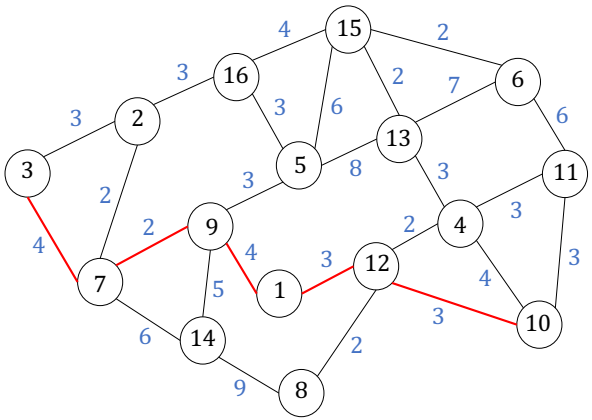
Before update



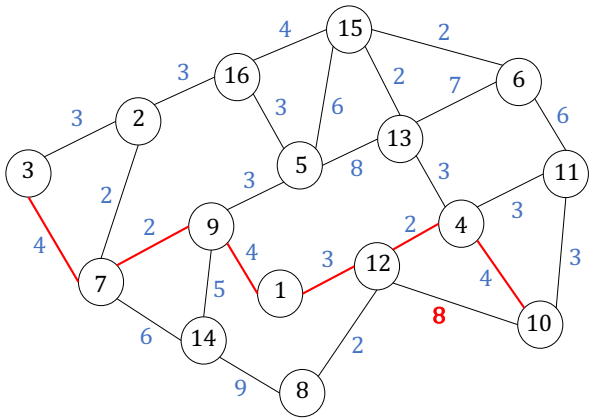
After update

Problem Definition

- Given a weighted graph $G = (V, E, \omega)$ that undergo edge weight updates, how to efficiently compute the correct distance of a shortest path between two vertices?



Before update



After update

– A shortest path between 3 and 10:

Before update: $\langle 3, 7, 9, 1, 12, 10 \rangle$

After update: $\langle 3, 7, 9, 1, 12, 4, 10 \rangle$

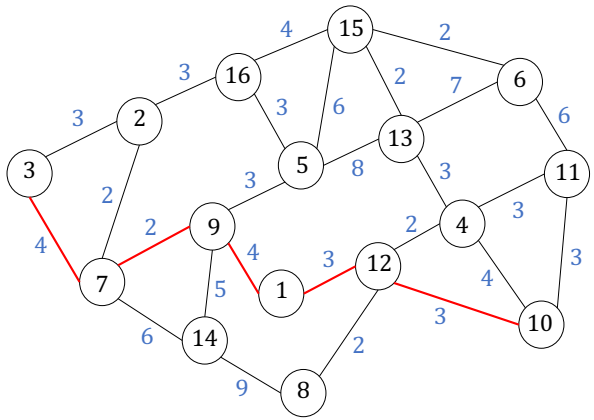
– The distance between 3 and 10:

Before update: $d_G(3, 10) = 16$

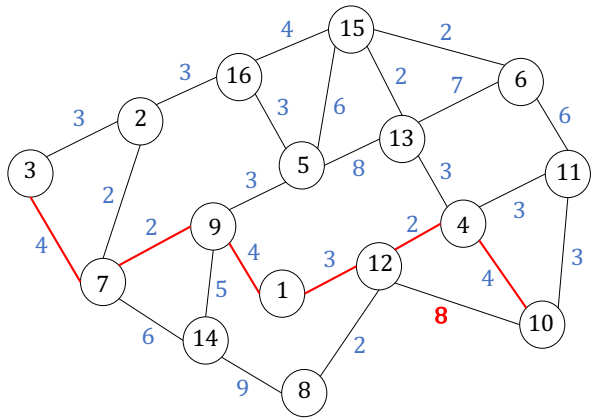
After update: $d_G(3, 10) = 19$

Problem Definition

- Given a weighted graph $G = (V, E, \omega)$ that undergo edge weight updates, how to efficiently compute the correct distance of a shortest path between two vertices?



Before update



After update

– A shortest path between 3 and 10:

Before update: $\langle 3, 7, 9, 1, 12, 10 \rangle$

After update: $\langle 3, 7, 9, 1, 12, 4, 10 \rangle$

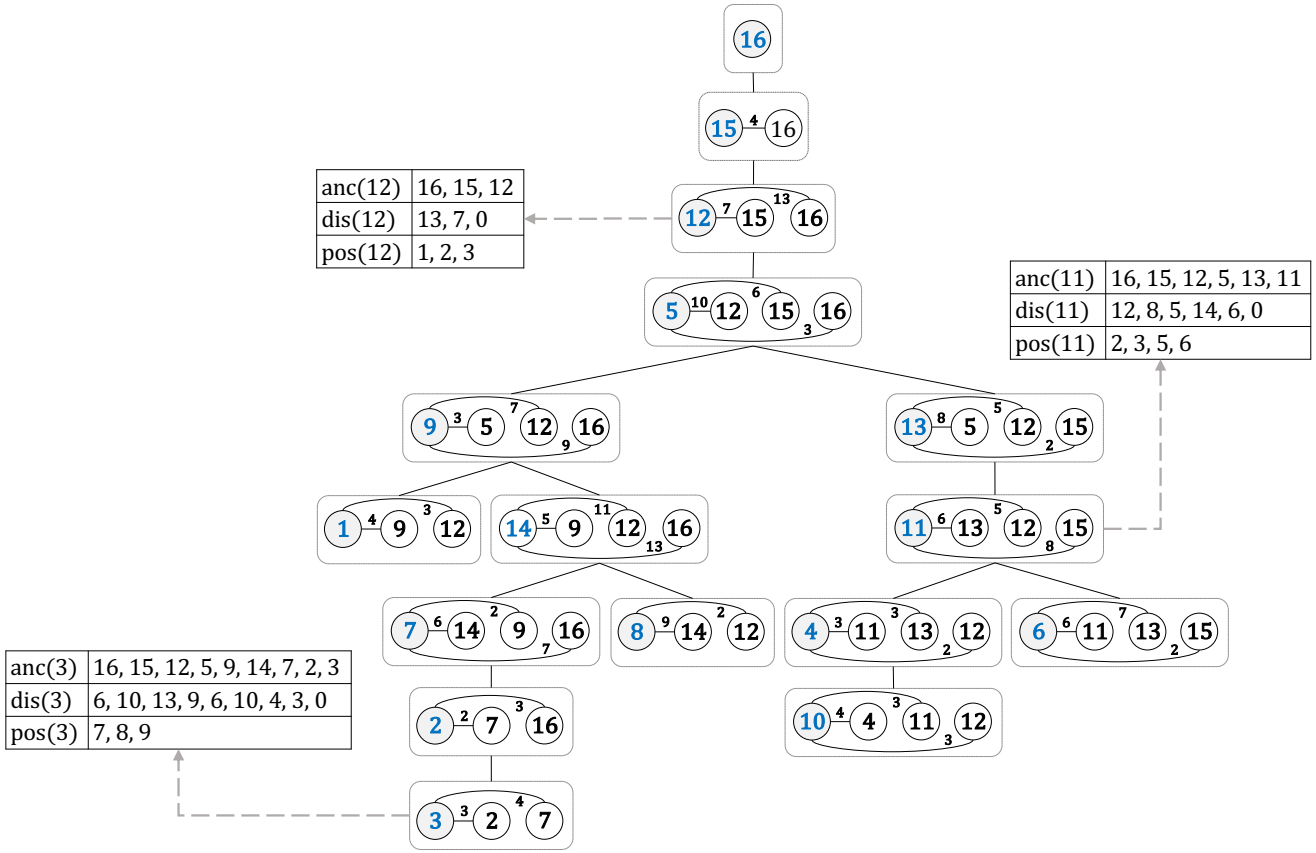
– The distance between 3 and 10:

Before update: $d_G(3, 10) = 16$

After update: $d_G(3, 10) = 19$

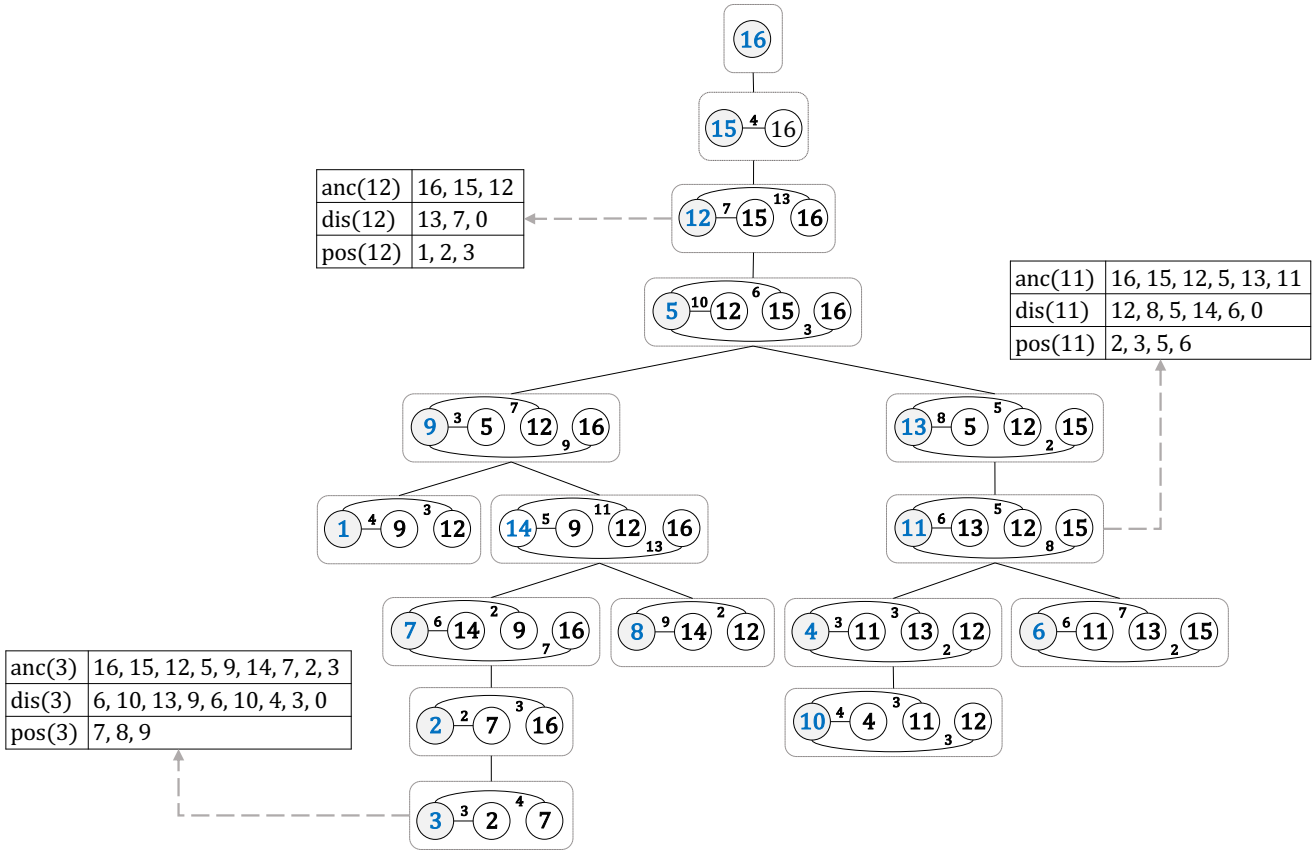
Applications: GPS navigation, route planning, traffic monitoring, etc.

Incremental Hierarchical 2-Hop (IncH2H)



H2H index

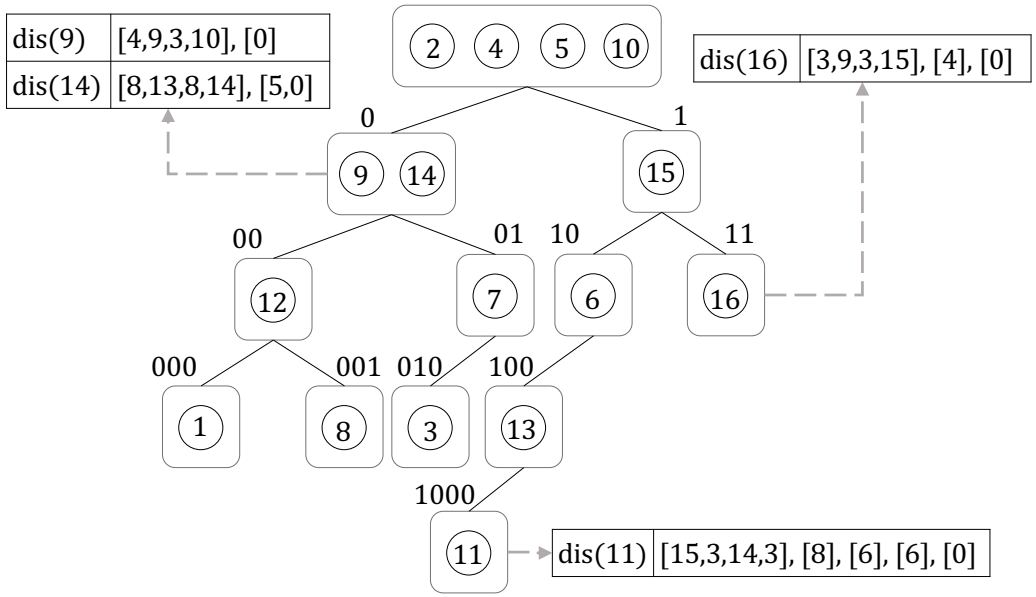
Incremental Hierarchical 2-Hop (IncH2H)



H2H index

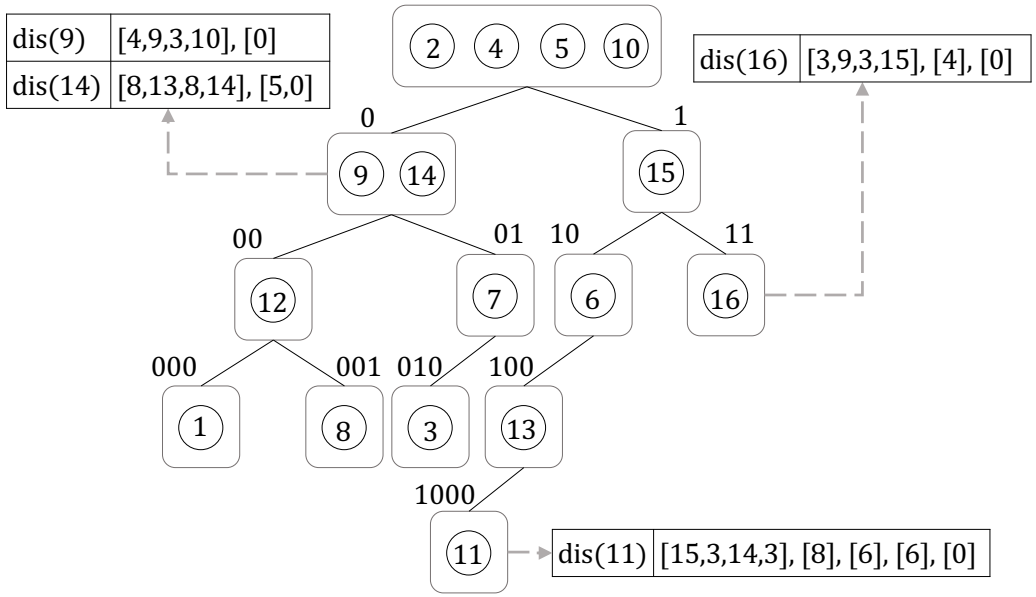
- **Limitations:**
  - Huge index to maintain due to very large tree width and height
  - Additional computational cost to compute LCA(s,t)

Hierarchical Cut 2-Hop Labelling (HC2L)



HC2L index.

Hierarchical Cut 2-Hop Labelling (HC2L)



HC2L index.

- **Limitations:**
  - The distance-preserving property hinders efficient maintenance of the HC2L index.

Our Solution - Stable Tree Labelling

Our Solution - Stable Tree Labelling

- **Observations**:
  - Eliminating the distance-preserving property from HC2L yields a binary tree that is structurally independent of edge weights$\hookrightarrow$ Stable Tree Hierarchy

Our Solution - Stable Tree Labelling

- **Observations**:
  - Eliminating the distance-preserving property from HC2L yields a binary tree that is structurally independent of edge weights$\hookrightarrow$ Stable Tree Hierarchy
  - Stable Tree Hierarchy is still balanced as it is inherited from HC2L

Our Solution - Stable Tree Labelling

- **Observations**:
  - Eliminating the distance-preserving property from HC2L yields a binary tree that is structurally independent of edge weights$\hookrightarrow$ Stable Tree Hierarchy
  - Stable Tree Hierarchy is still balanced as it is inherited from HC2L
  - Every path between any two vertices contains at least one of their common ancestors.

Our Solution - Stable Tree Labelling

- **Observations**:
  - Eliminating the distance-preserving property from HC2L yields a binary tree that is structurally independent of edge weights$\hookrightarrow$ Stable Tree Hierarchy
  - Stable Tree Hierarchy is still balanced as it is inherited from HC2L
  - Every path between any two vertices contains at least one of their common ancestors.
  - 2-hop labelling is constructed over Stable Tree Hierarchy and labels only store distances within subgraphs, not in the entire graph.

Our Solution - Stable Tree Labelling

(a) Stable Tree Hierarchy T

- At least one vertex in the set of common ancestors $CA(s, t)$ lies on a shortest path between s and t

Our Solution - Stable Tree Labelling

(a) Stable Tree Hierarchy T

- At least one vertex in the set of common ancestors $CA(s, t)$ lies on a shortest path between s and t

(b) 2-Hop Labelling L

- Hierarchical labelling w.r.t. the *vertex partial-order* induced by T
- a hub $r \in CA(s, t)$ exists for any two vertices $\{s, t\} \subseteq V$

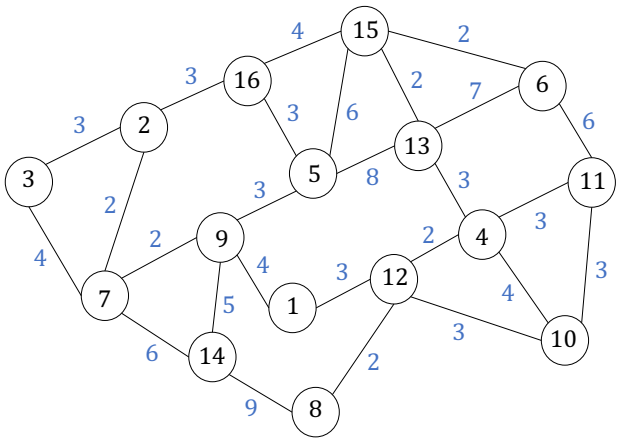
Our Solution - Stable Tree Labelling

(a) Stable Tree Hierarchy T

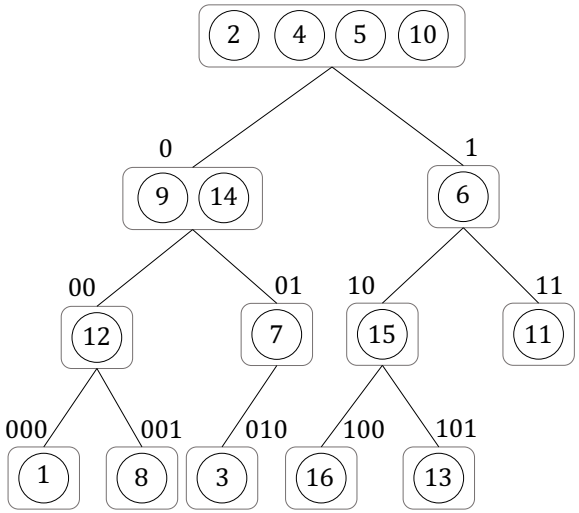
- At least one vertex in the set of common ancestors $CA(s,t)$ lies on a shortest path between s and t

(b) 2-Hop Labelling L

- Hierarchical labelling w.r.t. the *vertex partial-order* induced by T
- a hub $r \in CA(s,t)$ exists for any two vertices $\{s,t\} \subseteq V$



G



T

Label	Distance Entries
L(1)	[8,5,7,6], [4,14], [3], [0]
L(2)	[0]
L(3)	[3,15,9,16], [6,10], [4], [0]
L(4)	[12,0]
L(5)	[6,11,0]
L(6)	[9,7,8,9], [0]
L(7)	[2,11,5,12], [2,6], [0]
L(8)	[13,4,12,5], [9,9], [2], [0]
L(9)	[4,9,3,10], [0]
L(10)	[14,4,13,0]
L(11)	[15,3,14,3], [6], [0]
L(12)	[11,2,10,3], [7,11], [0]
L(13)	[9,3,8,13], [4], [2], [0]
L(14)	[8,13,8,14], [5,0]
L(15)	[7,5,6,11], [2], [0]
L(16)	[3,9,3,15], [6], [4], [0]

L

Our Solution - Stable Tree Labelling

- **Challenge**:

Our Solution - Stable Tree Labelling

- **Challenge**:
  - How to design dynamic algorithms that can quickly maintain 2-hop labelling L to ensure efficient querying?

Our Solution - Stable Tree Labelling

- **Challenge**:
  - How to design dynamic algorithms that can quickly maintain 2-hop labelling L to ensure efficient querying?
- **Main ideas**:

Our Solution - Stable Tree Labelling

- **Challenge**:
  - How to design dynamic algorithms that can quickly maintain 2-hop labelling L to ensure efficient querying?
- **Main ideas**:
  - For an edge update (a, b), labels w.r.t. common ancestors of a and b need to be maintained.

Our Solution - Stable Tree Labelling

- **Challenge**:
  - How to design dynamic algorithms that can quickly maintain 2-hop labelling L to ensure efficient querying?
- **Main ideas**:
  - For an edge update (a, b), labels w.r.t. common ancestors of a and b need to be maintained.

    $\hookrightarrow$ Develop *Label Search algorithms* – search from a very small set of ancestors to maintain 2-hop labelling.

Our Solution - Stable Tree Labelling

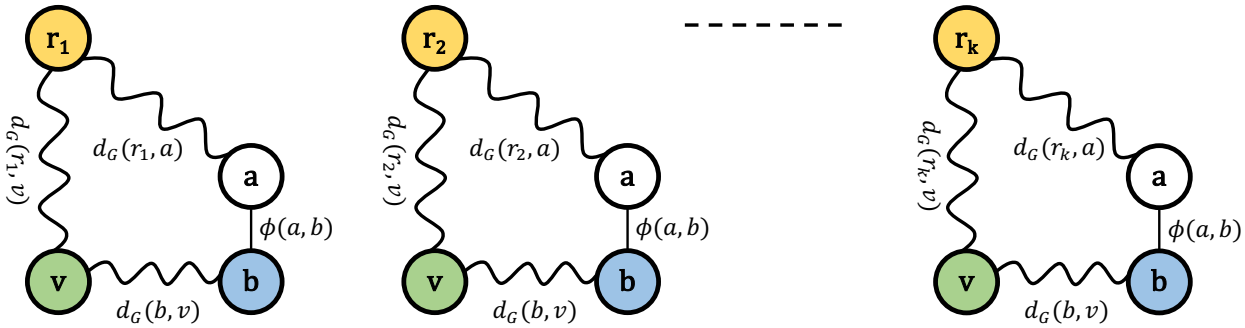
- **Challenge**:
  - How to design dynamic algorithms that can quickly maintain 2-hop labelling L to ensure efficient querying?
- **Main ideas**:
  - For an edge update (a, b), labels w.r.t. common ancestors of a and b need to be maintained.

    $\hookrightarrow$ Develop *Label Search algorithms* – search from a very small set of ancestors to maintain 2-hop labelling.

    $\hookrightarrow$ Develop *Pareto Search algorithms* which improve *Label Search algorithms* – explore the interaction between search spaces of different ancestors.

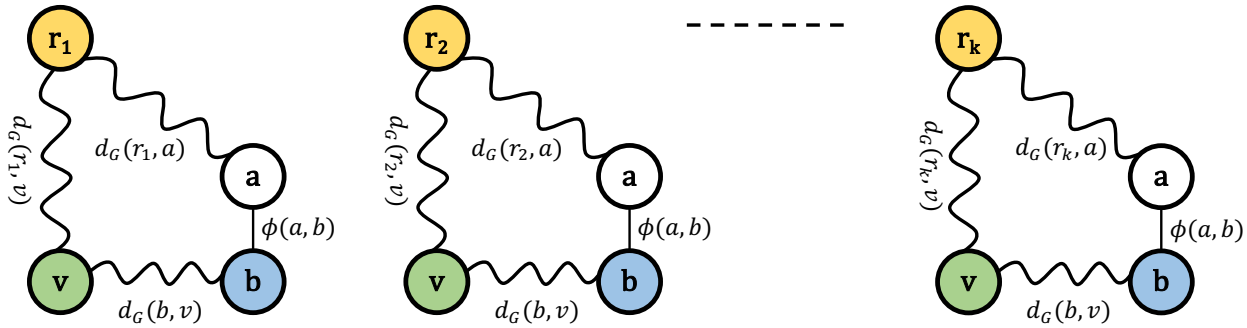
Label Search Algorithms

- For each ancestor, identify affected labels to update using triangle inequality

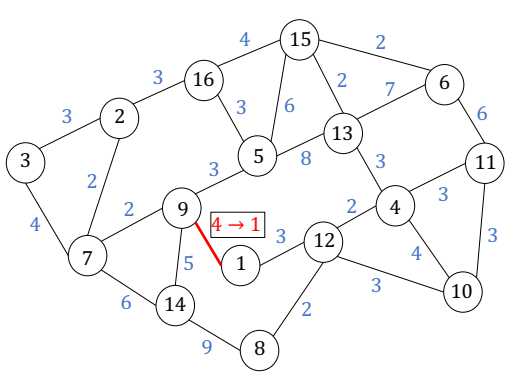


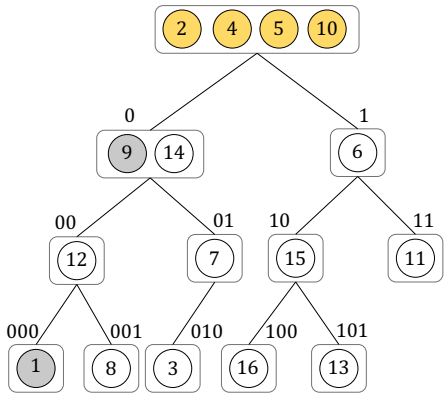
Label Search Algorithms

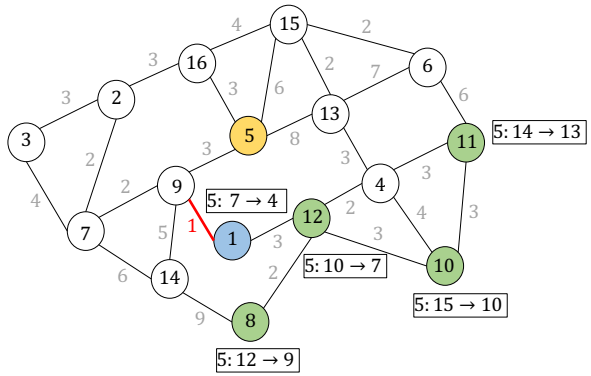
- For each ancestor, identify affected labels to update using triangle inequality



- Label search algorithm for handling weight decrease (STL-L$^-$)

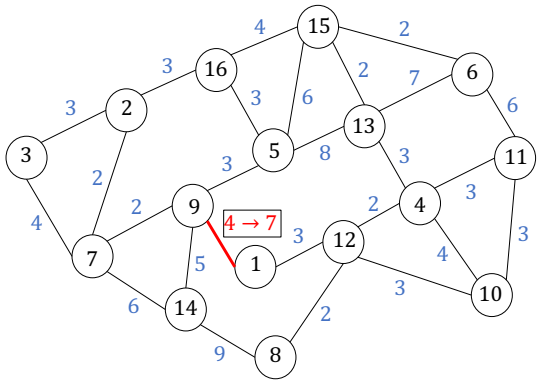


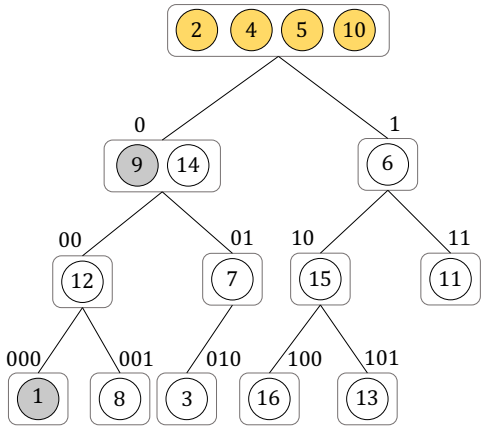




Label Search Algorithms

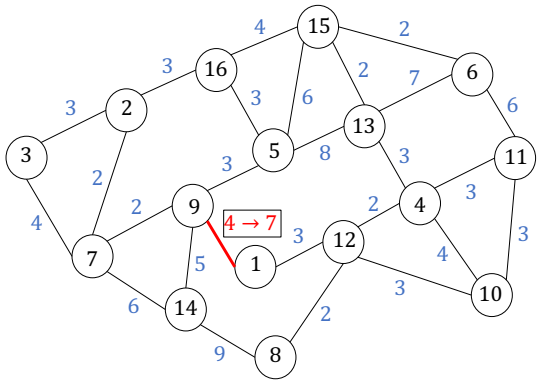
- Label search algorithm for handling weight increase (STL-L$^+$)

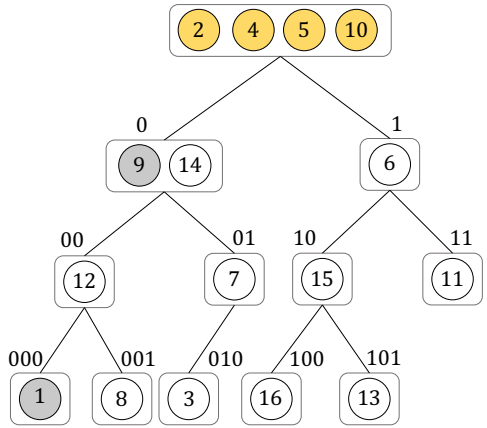




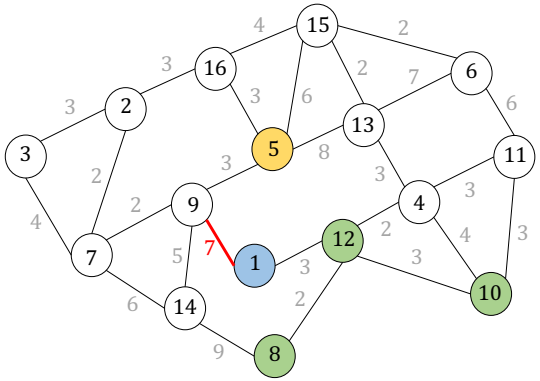
Label Search Algorithms

- Label search algorithm for handling weight increase (STL-L$^+$)





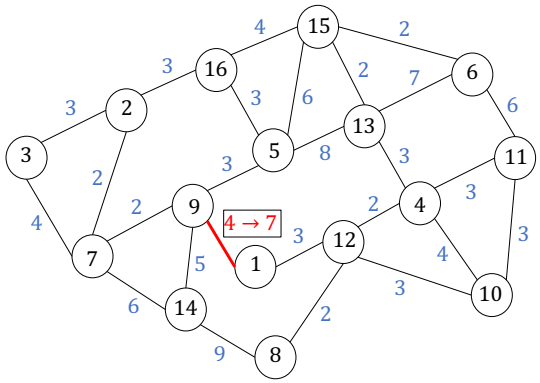
- Proceeds in two steps:

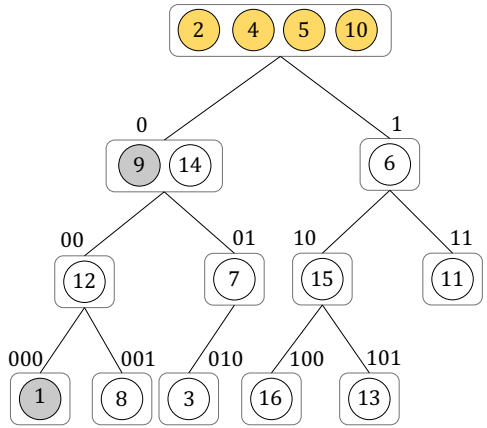


Find Affected Labels

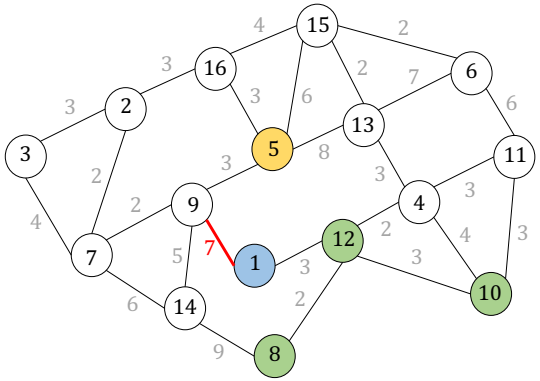
Label Search Algorithms

- Label search algorithm for handling weight increase (STL-L$^+$)

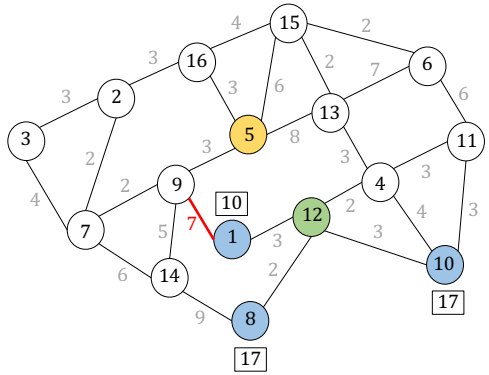


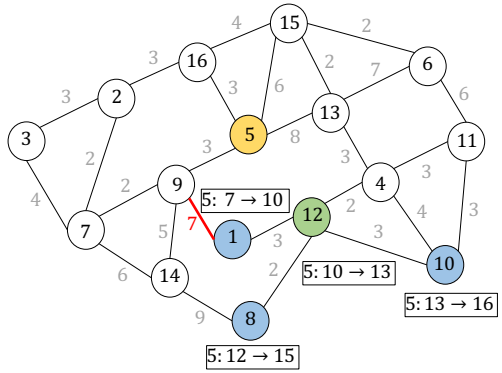


- Proceeds in two steps:



Find Affected Labels





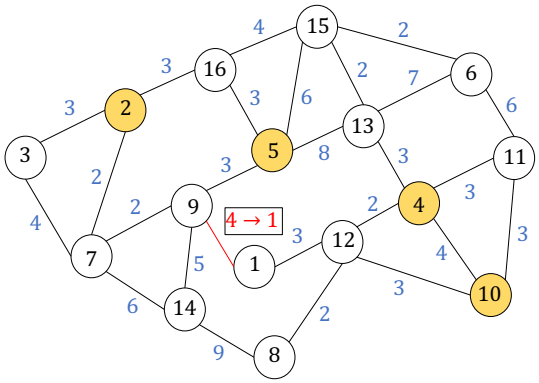
Repair Affected Labels

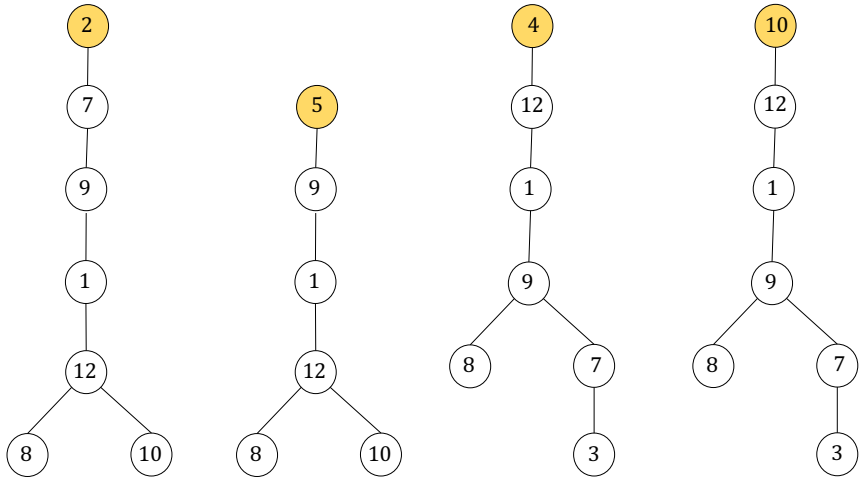
Pareto Search Algorithms

- **Observation:** In Label Search algorithms, searches from different ancestors may lead to common sub-paths being traversed multiple times

Pareto Search Algorithms

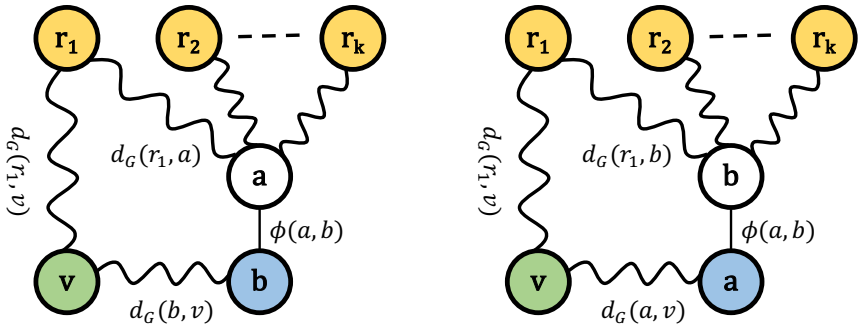
- **Observation:** In Label Search algorithms, searches from different ancestors may lead to common sub-paths being traversed multiple times





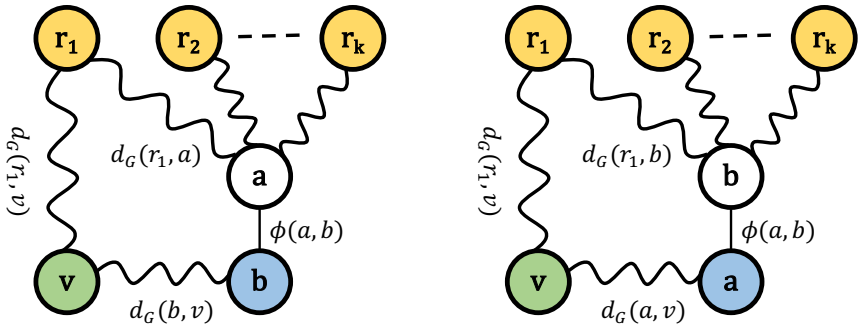
Pareto Search Algorithms

- For each update (a, b), combine searches from multiple ancestors into only two searches

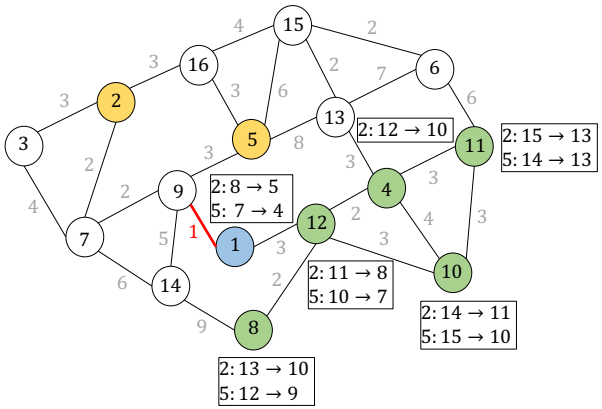


Pareto Search Algorithms

- For each update (a, b), combine searches from multiple ancestors into only two searches



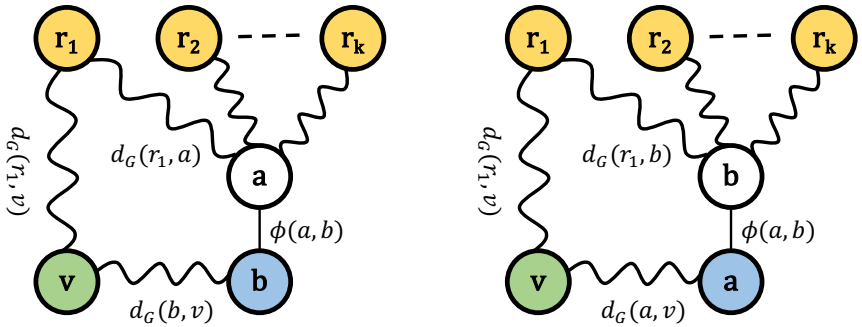
- Pareto search algorithms handle weight decrease and increase via STL-P$^-$ and STL-P$^+$, respectively



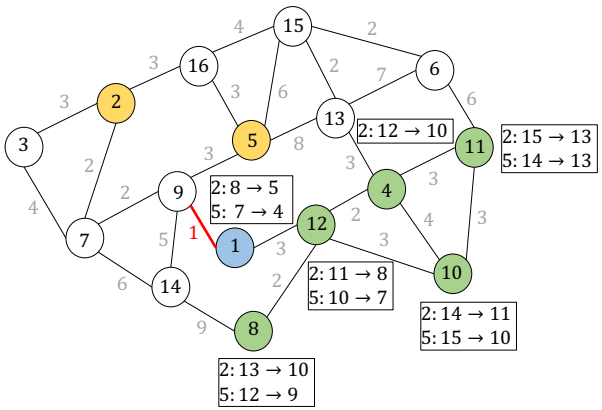
Edge Weight Decrease

Pareto Search Algorithms

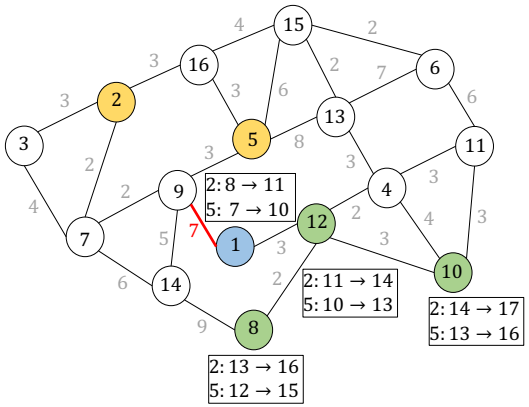
- For each update (a, b), combine searches from multiple ancestors into only two searches

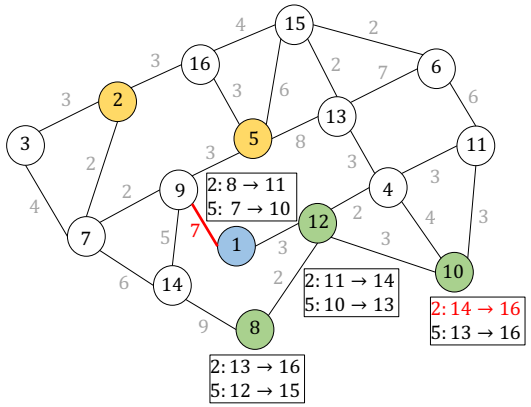


- Pareto search algorithms handle weight decrease and increase via STL-P$^-$ and STL-P$^+$, respectively



Edge Weight Decrease





Edge Weight Increase

Empirical Evaluation

Network	Update Time - Decrease [ms]				Update Time - Increase [ms]			
	STL-P$^-$	STL-L$^-$	IncH2H$^-$	DTDHL$^-$	STL-P$^+$	STL-L$^+$	IncH2H$^+$	DTDHL$^+$
NY	0.845	1.978	2.006	11.40	1.712	3.561	2.900	13.87
BAY	0.917	1.788	1.769	8.899	1.695	3.233	2.498	14.53
COL	1.898	3.882	3.306	12.74	3.456	6.977	4.613	34.35
FLA	2.303	5.209	3.585	32.45	4.109	9.554	4.981	34.22
CAL	4.975	16.67	13.89	99.24	10.11	31.04	20.20	106.4
E	7.996	39.21	29.33	261.5	17.48	73.76	43.57	273.1
W	12.26	52.71	47.76	604.9	25.14	100.2	68.99	1,292
CTR	27.23	164.4	213.1	2,329	54.03	314.5	309.7	5,347
USA	32.67	216.4	239.8	–	82.78	412.9	356.3	–
EUR	13.68	68.25	66.97	–	61.57	131.4	96.63	–

- *5-7 times faster in terms of update time on large road networks compared to SOTA IncH2H*
- *several orders of magnitude faster in terms of update time compared to DTDHL*

Empirical Evaluation

Network	Query Time [μs]				Labelling Size				Construction Time [s]			
	STL	HC2L	IncH2H	DTDHL	STL	HC2L	IncH2H	DTDHL	STL	HC2L	IncH2H	DTDHL
NY	0.287	0.264	0.913	0.852	129 MB	172 MB	850 MB	391 MB	2	3	4	9
BAY	0.299	0.258	0.841	0.785	104 MB	134 MB	814 MB	377 MB	2	3	3	5
COL	0.349	0.318	1.018	0.988	175 MB	238 MB	1.37 GB	587 MB	4	6	5	7
FLA	0.396	0.349	1.019	0.958	423 MB	561 MB	2.43 GB	1.30 GB	11	16	11	17
CAL	0.490	0.484	1.333	1.380	1.03 GB	1.48 GB	8.21 GB	3.91 GB	28	44	30	48
E	0.630	0.550	1.683	1.585	2.92 GB	4.22 GB	20.7 GB	9.68 GB	75	129	74	111
W	0.664	0.601	1.702	1.819	4.82 GB	7.01 GB	36.3 GB	20.6 GB	120	249	126	194
CTR	0.812	0.702	2.483	2.658	19.7 GB	30.2 GB	178 GB	80.3 GB	540	1,140	858	766
USA	0.834	0.734	3.428	–	35.6 GB	53.6 GB	308 GB	–	852	1,721	1,081	–
EUR	1.185	0. 879	3.888	–	36.4 GB	51.2 GB	322 GB	–	1,236	2,354	1,254	–

Compared to IncH2H

- *about 3 times faster in terms of query time*
- *consuming about an order of magnitude less space*

Compared to DTDHL

- *significantly faster in terms of query time*
- *consuming 20%-30% of space for labelling*

Concluding Remarks

- Stable tree labeling (STL)

  serve as the foundation for designing efficient dynamic algorithms.

Concluding Remarks

- Stable tree labeling (STL)

  serve as the foundation for designing efficient dynamic algorithms.

- Label Search and Pareto Search algorithms from different perspectives

  Label Search is ancestor-centric while Pareto Search is update-centric.

Concluding Remarks

- Stable tree labeling (STL)

  serve as the foundation for designing efficient dynamic algorithms.

- Label Search and Pareto Search algorithms from different perspectives

  Label Search is ancestor-centric while Pareto Search is update-centric.

- Our solutions are scalable on large and dynamic road networks

  The power lies in "stable tree hierarchy".

Questions

Thank You