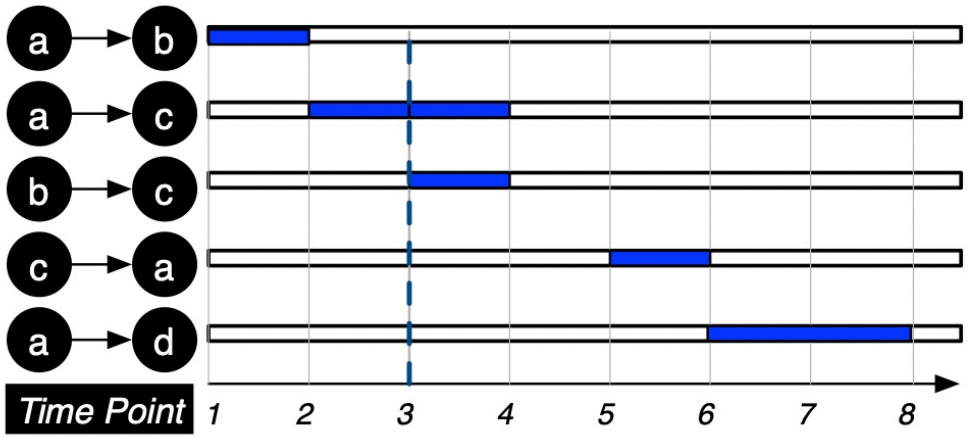
Hybrid Sampling Algorithms for Dynamic Graph Sampling

Chengying Huan
Institute of Software, Chinese Academy of Sciences

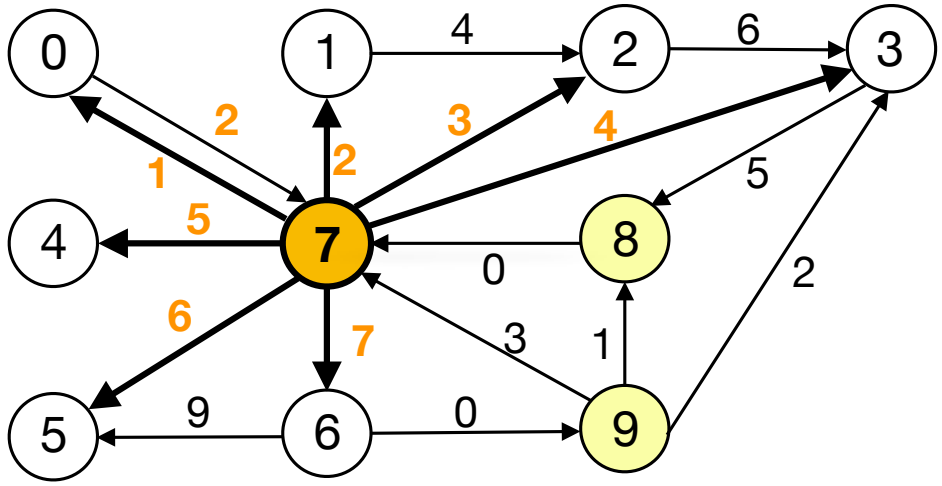
Background

- When do we need to use the dynamic graph sampling?
- Temporal Graph Random Walk
  - Dynamic graph sampling caused by time limit
- Evolving Graph Random Walk
  - Dynamic graph sampling caused dynamic edge modification

What's temporal graphs ?



Flight map



Social network

Temporal graph: Graph with time instances

Temporal path: Time increasing on paths

Random walks on temporal graphs

Random walk: A temporal path starting from a certain edge by randomly sampling

Candidate edge set: candidate edges for each sampling (satisfy time constraints)

Decided by previous arriving time

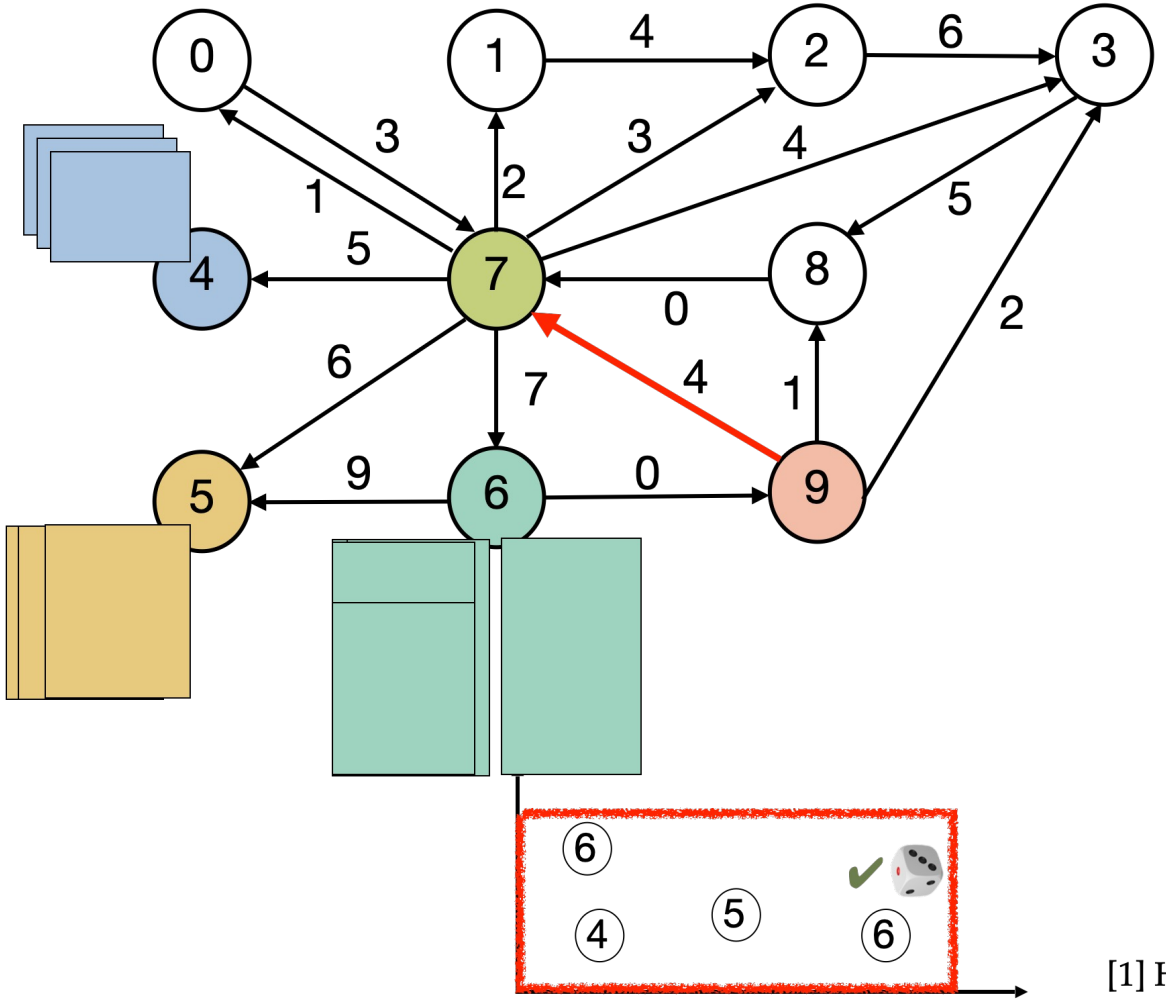
$$\Gamma_t(u) = \{(u, v_i, t_i) | \ (u, v_i, t_i) \in N(u), \ t_i > t\}$$

Transition probability: probability distribution of edges on each candidate edge set

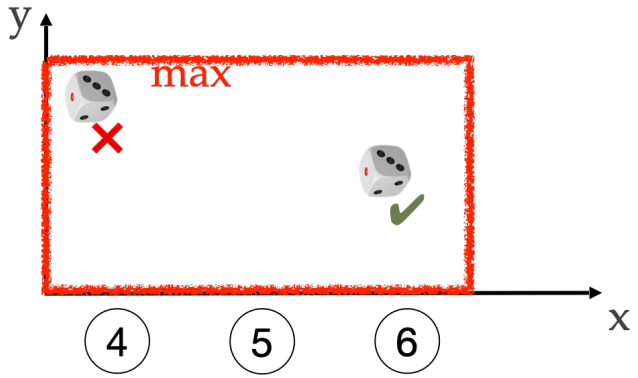
Decide by previous arriving time

$$P((u, v_i, t_i)) = \beta_{(u,v_i)} \cdot \frac{\delta((u, v_i, t_i))}{\sum_{(u,v_j,t_j)\in\Gamma_t(u)} \delta((u, v_j, t_j))}$$

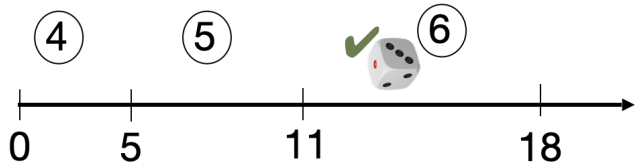
Question: Random Walk 9 → 7 → ?



Alias method



Rejection method

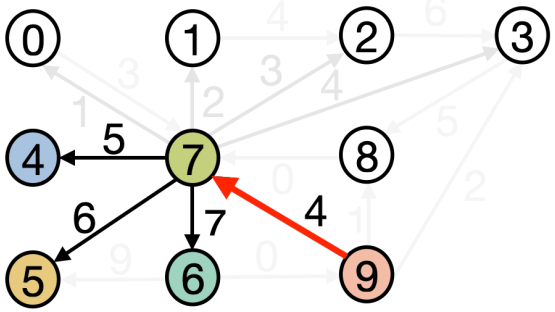


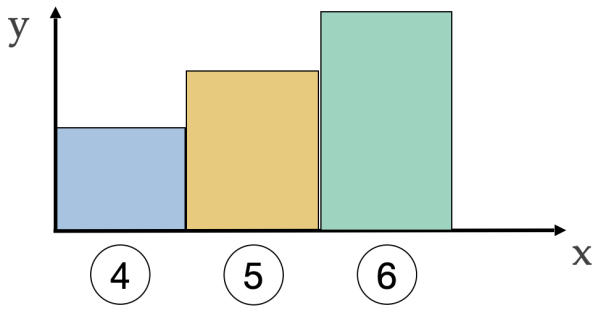
Inverse Transform Sampling (ITS)

[1] Huan, C and Pandey, S. and Liu, H etc. 2023. TEA: A General Purpose Temporal Graph Random Walk Engine. Eurosys '23

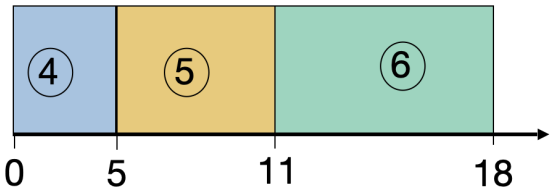
x

Challenges

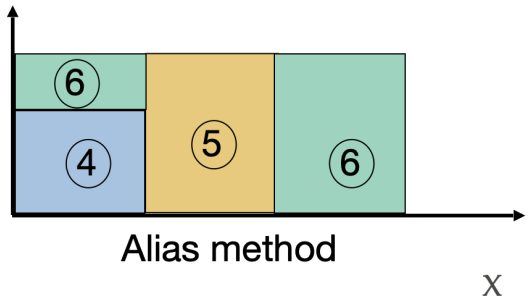




Rejection method

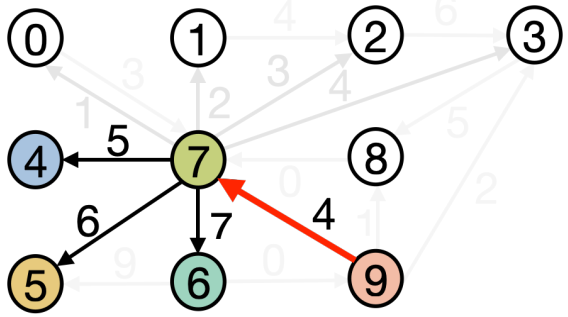


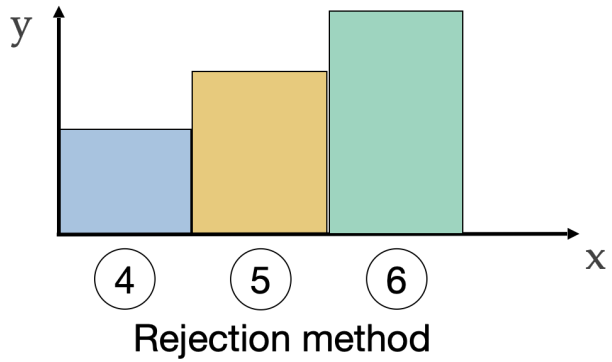
Inverse Transform Sampling (ITS)



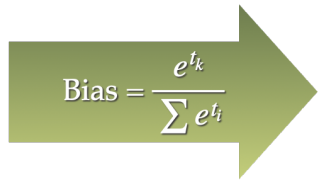
$$\text{Bias} = \frac{e^{t_k}}{\sum e^{t_i}}$$

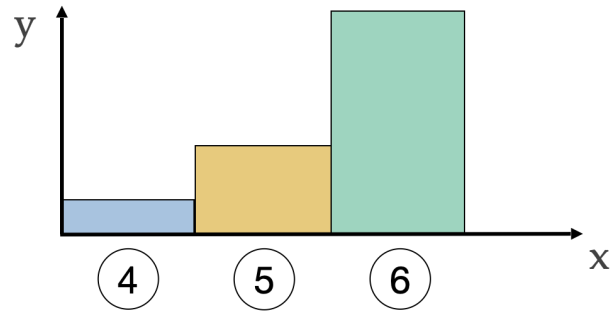
Challenges

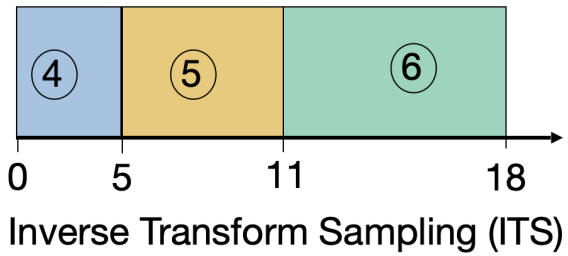




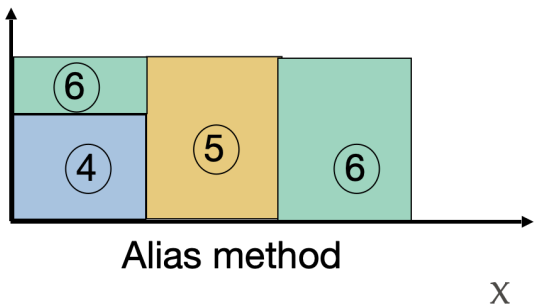
Rejection method



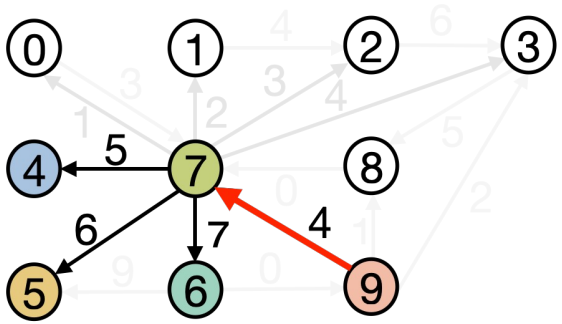


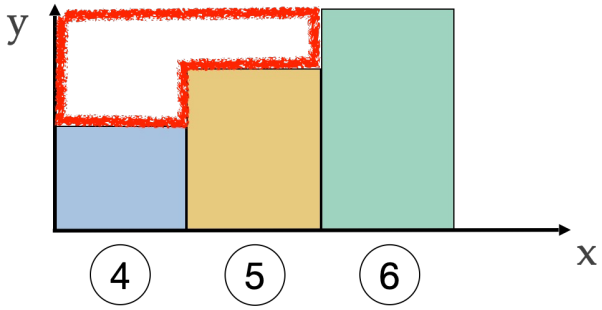


Inverse Transform Sampling (ITS)



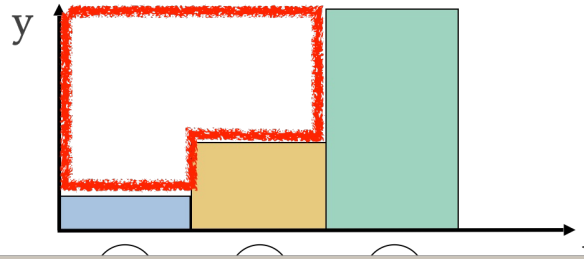
Challenges



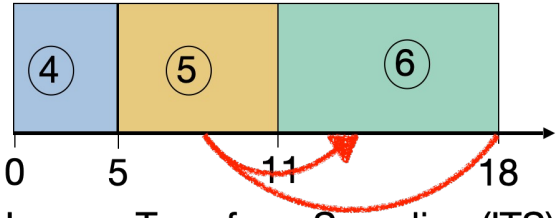


Rejection method

Bias = $\frac{e^{t_k}}{\sum e^{t_i}}$



Very high rejection ratio!!!

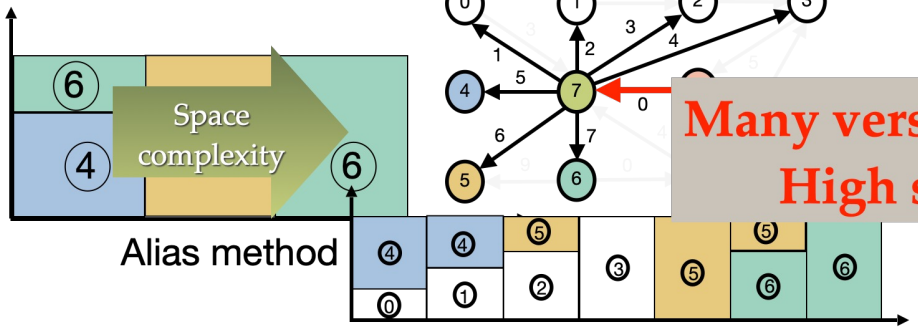


Inverse Transform Sampling (ITS)

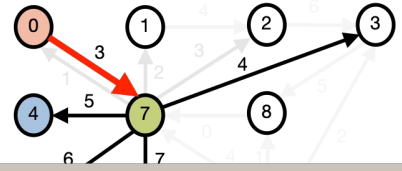
Time complexity

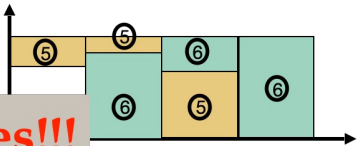
ALWAYS LogN time complexity!

Rand=13



Alias method

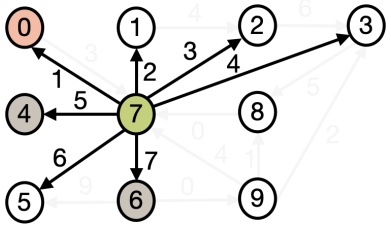


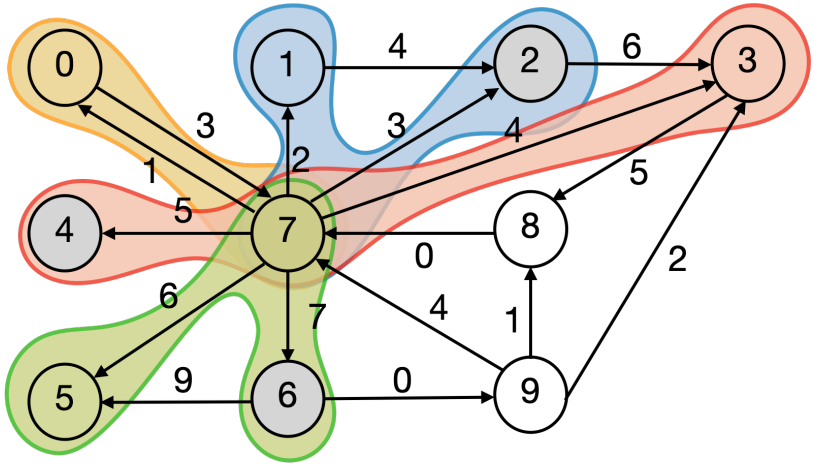


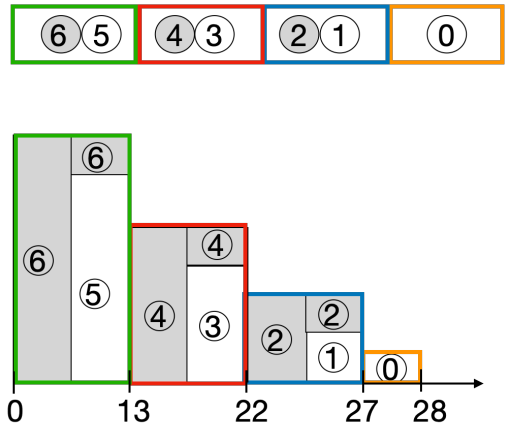
Many versions of alias tables!!!
High space complexity

x

Persistent Alias Table [Eurosys '23]

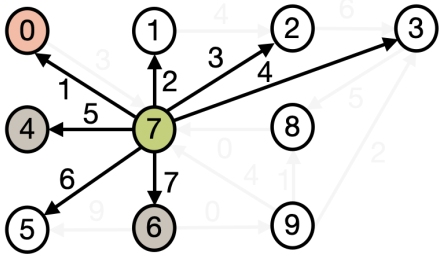


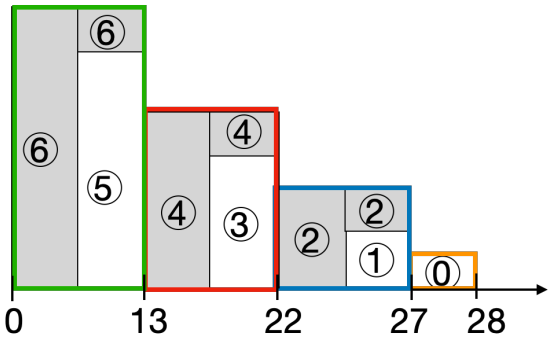




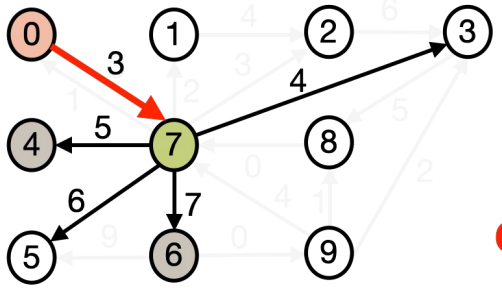
x

Sampling on Persistent Alias Table



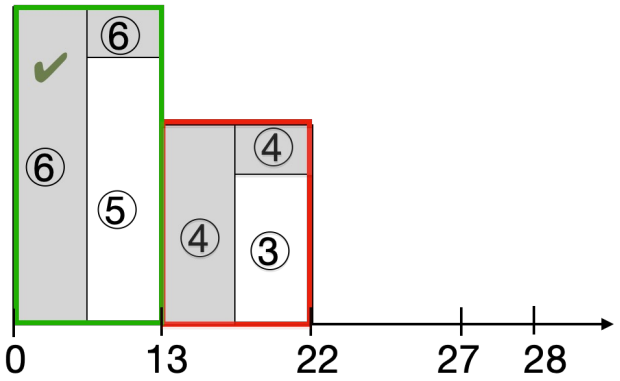


Sampling on Persistent Alias Table



Complete alias tables

O(log (N/|AliasTable|))

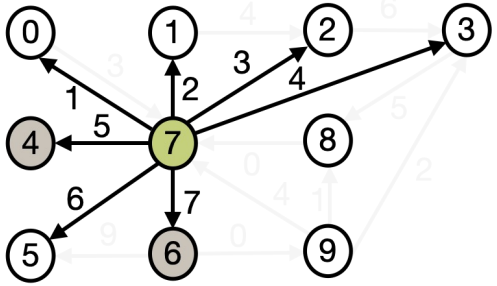


Step 1: Using ITS to select the alias table of interest.

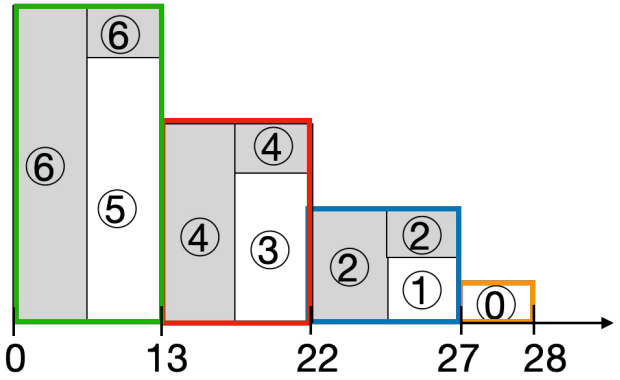
Random = 12

Step 2. Using alias method to select the vertex.

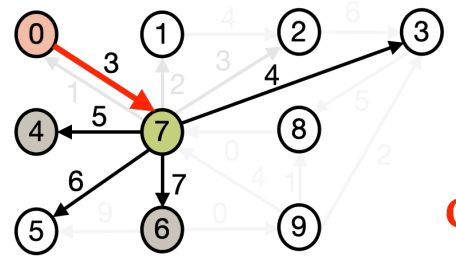
Vertex = ⑥



Incomplete alias tables

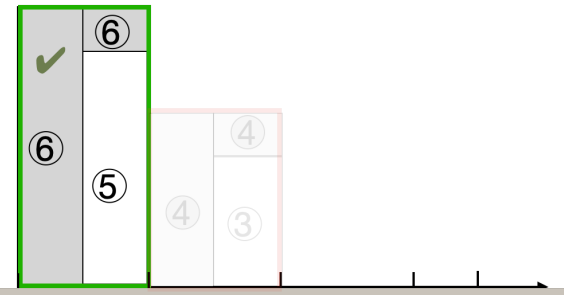


Sampling on Persistent Alias Table



Complete alias tables

O(log (N/|AliasTable|))



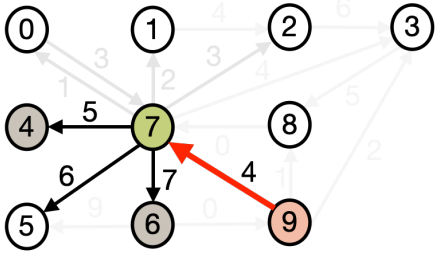
Step 1: Using ITS to select the alias table of interest.

Random = 12

Step 2. Using alias method to select the vertex.

Since |AliasTable| affects the complexity,

We further optimize |AliasTable|!



Incomplete alias tables

O(log (N/|AliasTable|) + log |AliasTable|)

ct the alias table of interest.

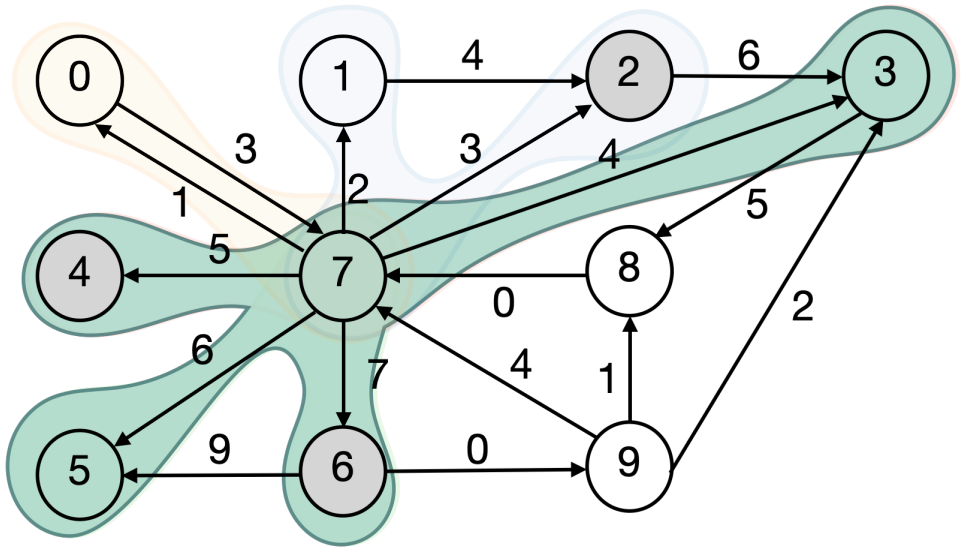
Random = 14

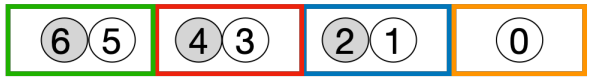
Step 2. Using ITS method to select the vertex from incomplete alias table.

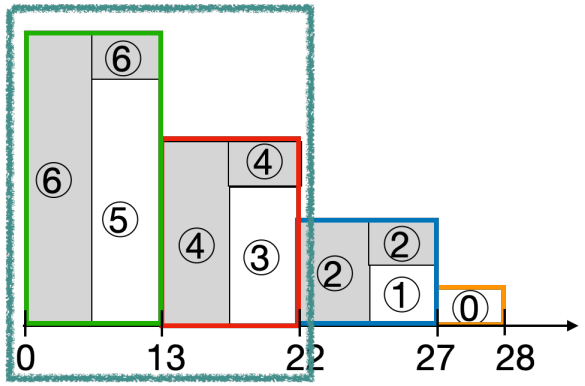
Vertex = ④

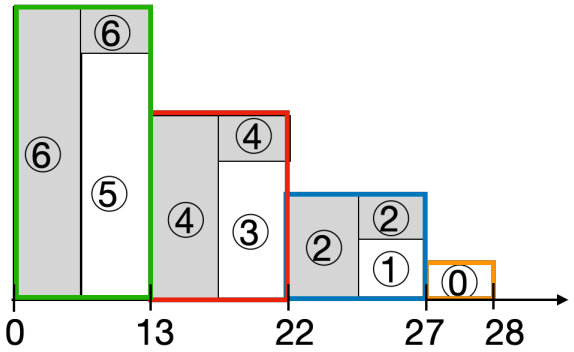
x

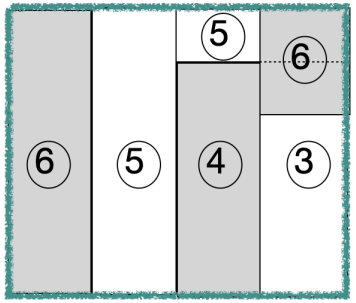
Hierarchical Persistent Alias Table

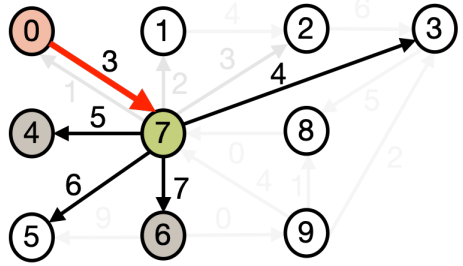






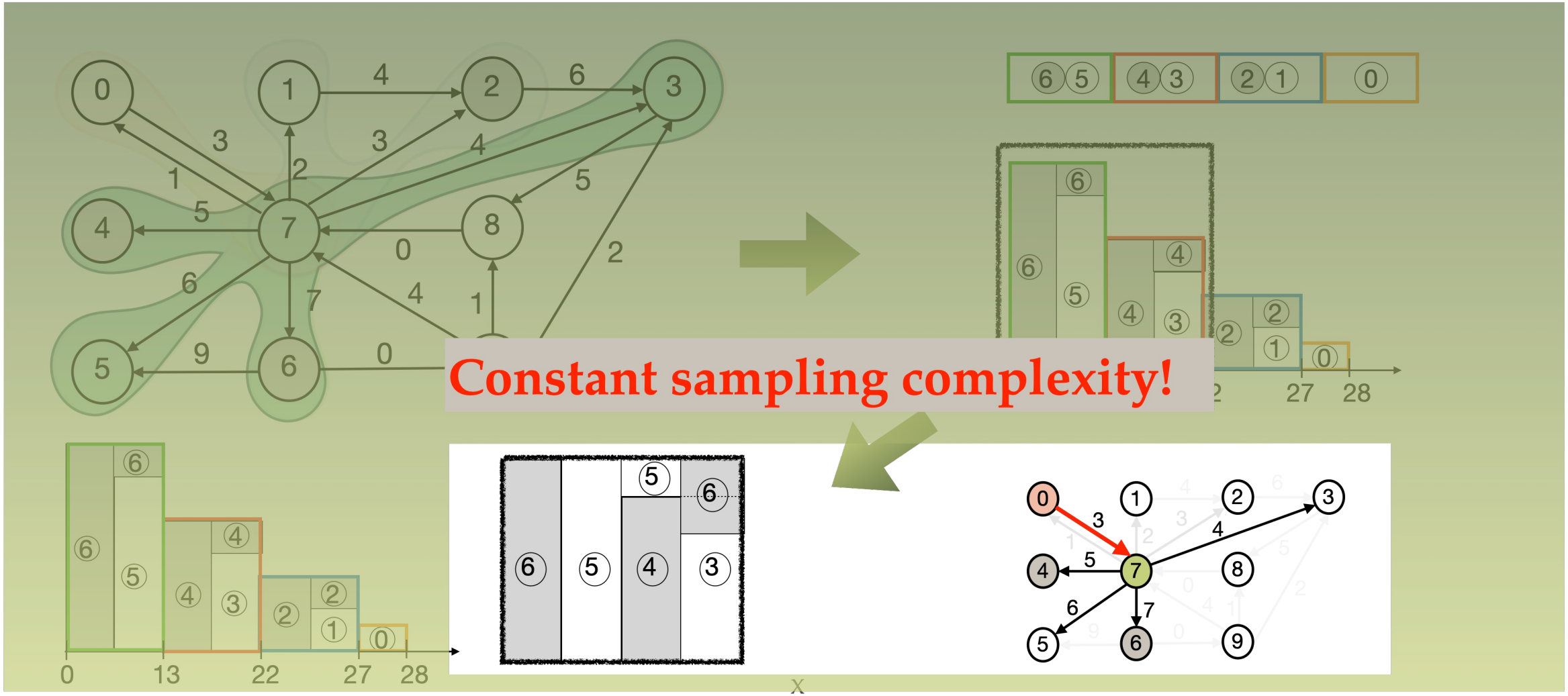






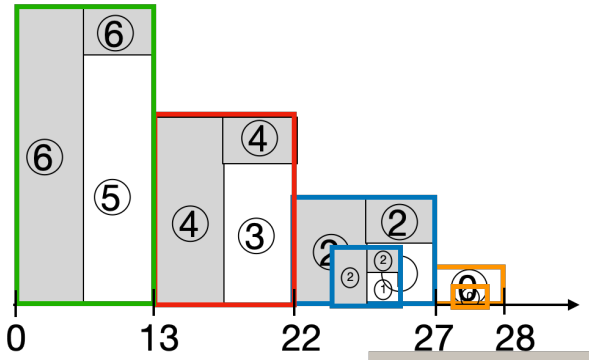
x

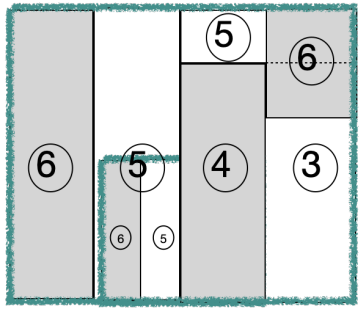
Hierarchical Persistent Alias Table

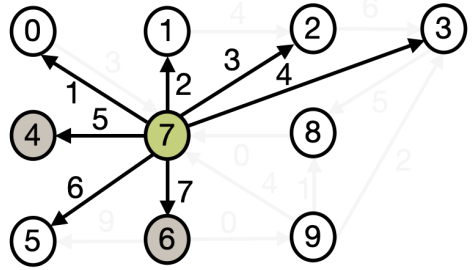


How to rapidly identify the candidate alias tables?

Auxiliary Index for Fast Alias Table Identification





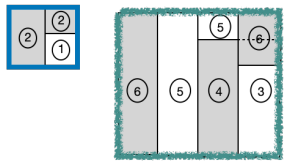


Temporal range:

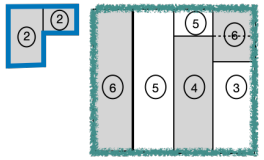
1-7 = (1) + (2-3) + (4-7)

Constant time alias table identification!

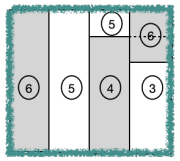
2-7 = (2-3) + (4-7)



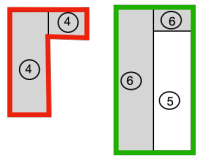
3-7 = (3) + (4-7)



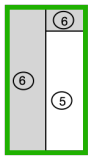
4-7 = (4-7)



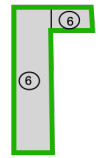
5-7 = (5) + (6-7)



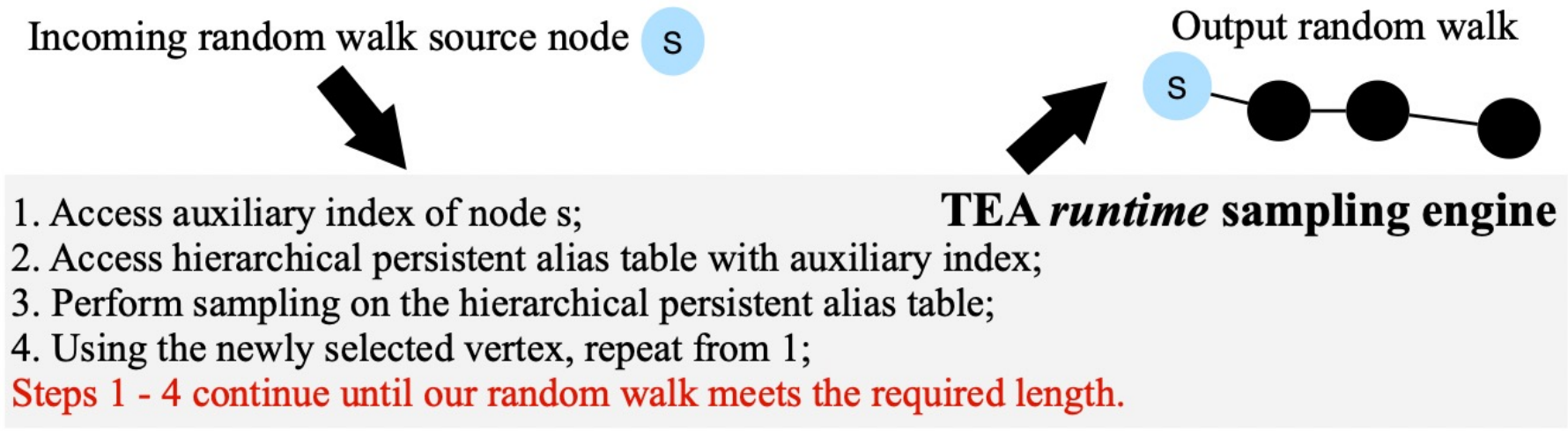
6-7 =(6-7)



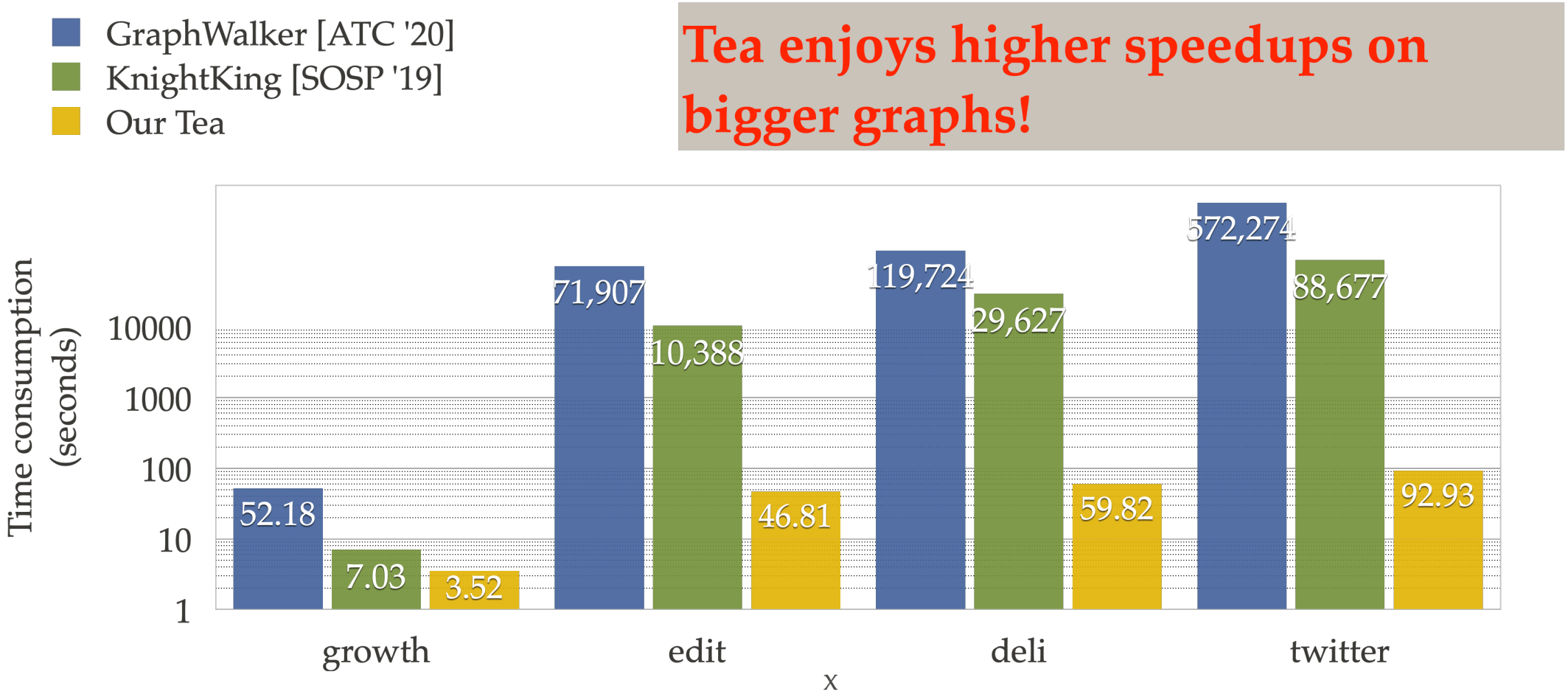
7 =(7)



System overview:



Tea vs. the State-of-the-art

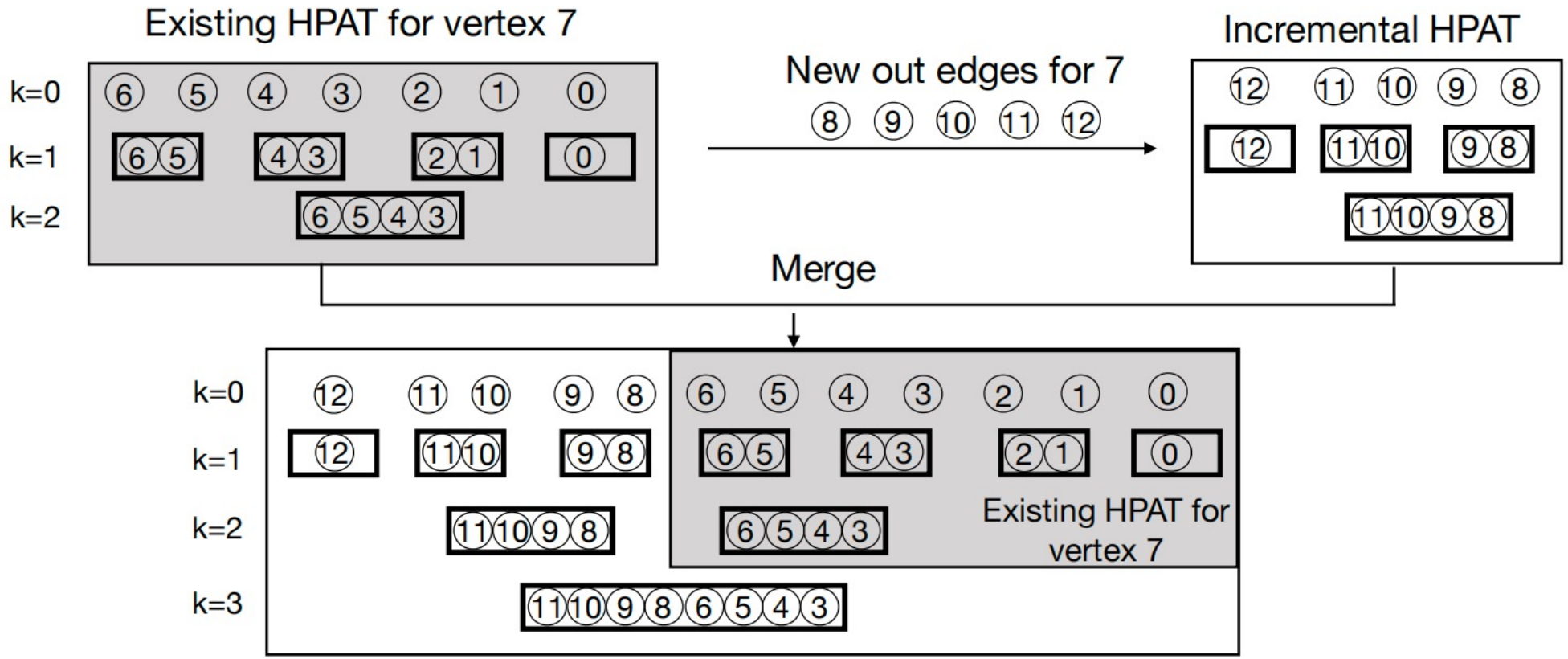


How to deal with dynamic edge modification?

Incremental HPAT Updating

- Edge Insertion
  - Create new HPATs and merge the new HPATs into old HPATs
- Edge Deletion
  - Delete the corresponding HPATs

Edge Insertion

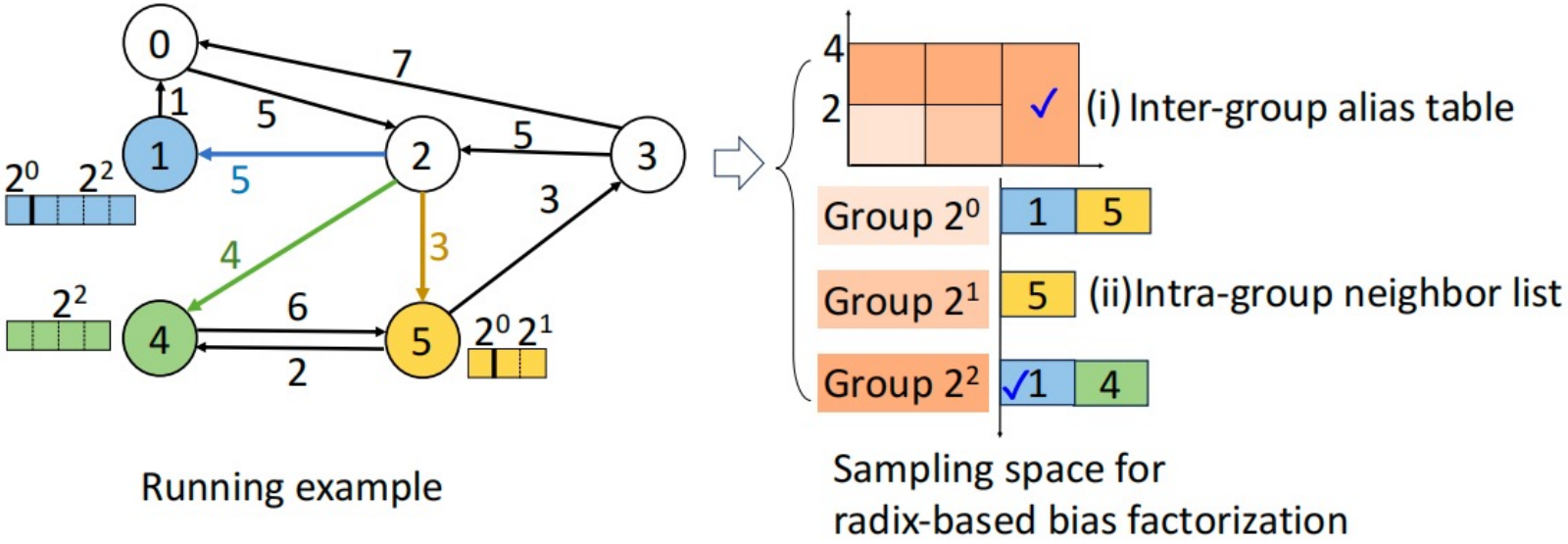


Edge Deletion

- Remove the trunks containing the deleted edge
- For example:
  - when deleting the edge from vertex 7 to {0,1,2,3}
  - the trunks containing {0,1,2,3} will be directly deleted
  - this is correct because we arrange the edges by time decreasing and each candidate edge set lenght can be binary decomposition
- The time complexity of each edge deletion is O(1)

How to deal with evolving graph without time instances?

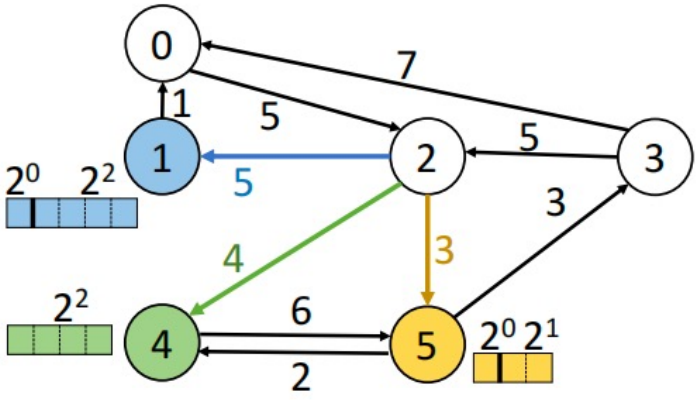
Radix-based Bias Factorization



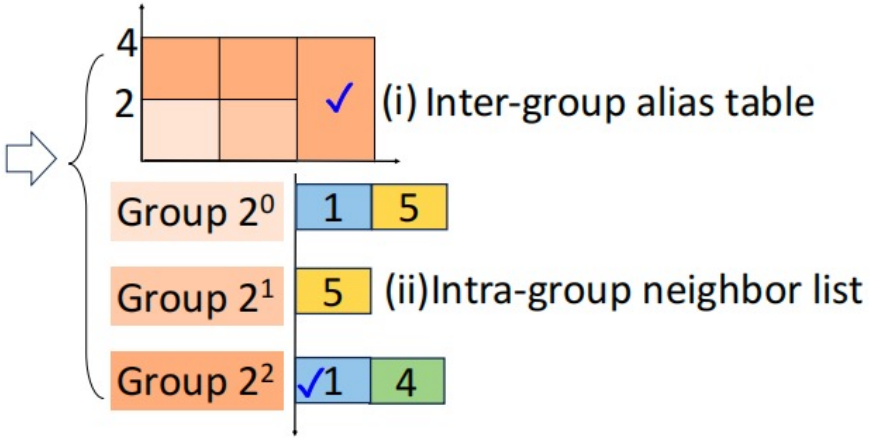
Binary decompositi
for edge weight

O(log(log(D))) sampling complexity

Edge insertion



Running example



Sampling space for
radix-based bias factorization

O(log(D)) time complexity

Thank you ~~