Succinct Data Structure for Path Graphs

Data Compression Conference 2022

24 March 2022

Girish Balakrishnan[1], Sankardeep Chakraborty[2], N S Narayanaswamy[1] and Kunihiko Sadakane[2]

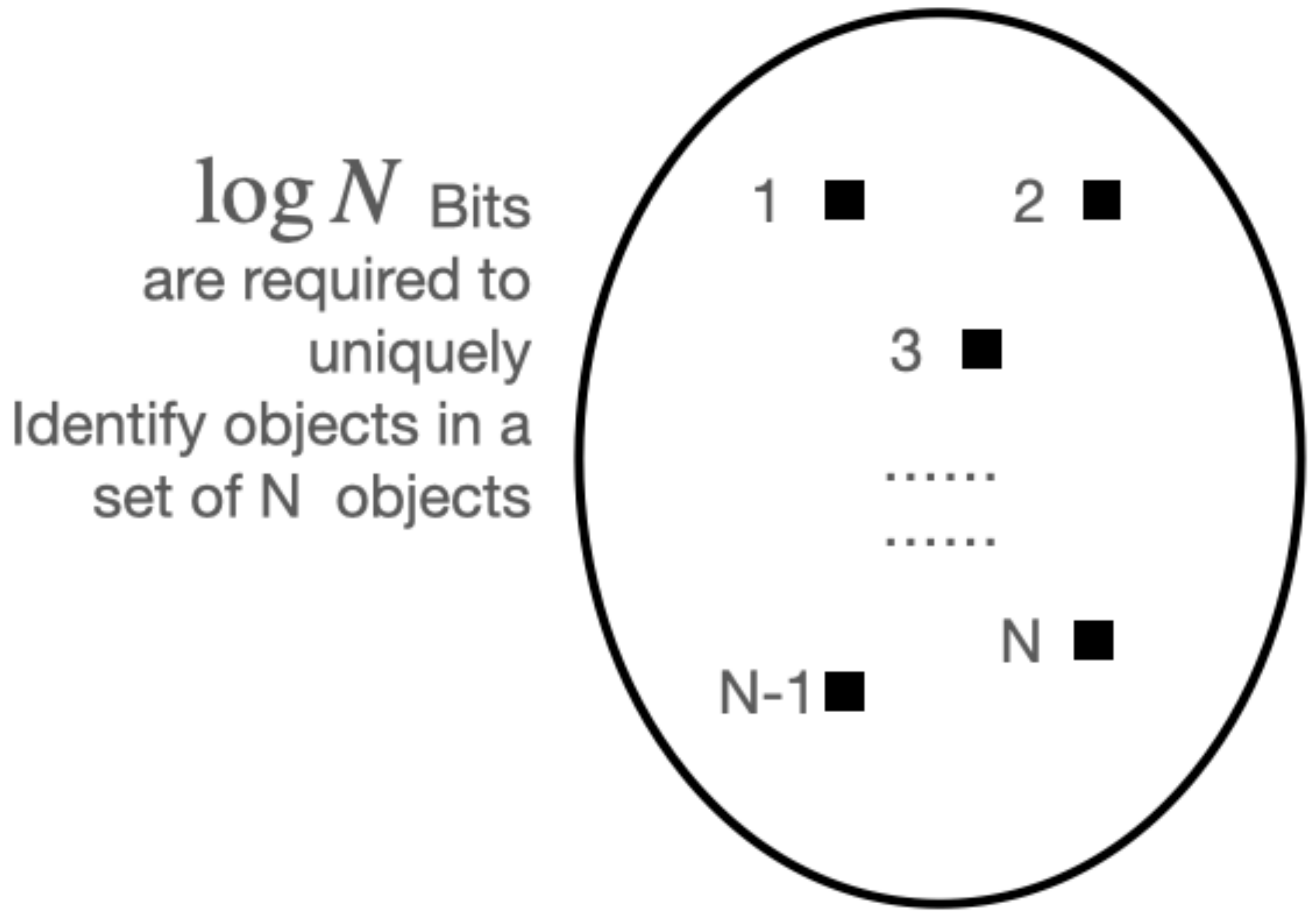
1 - Indian Institute of Technology Madras, India
2 - University of Tokyo, Japan

PROBLEM STATEMENT **:** Design a *succinct data structure* for path graphs

Basic Definitions

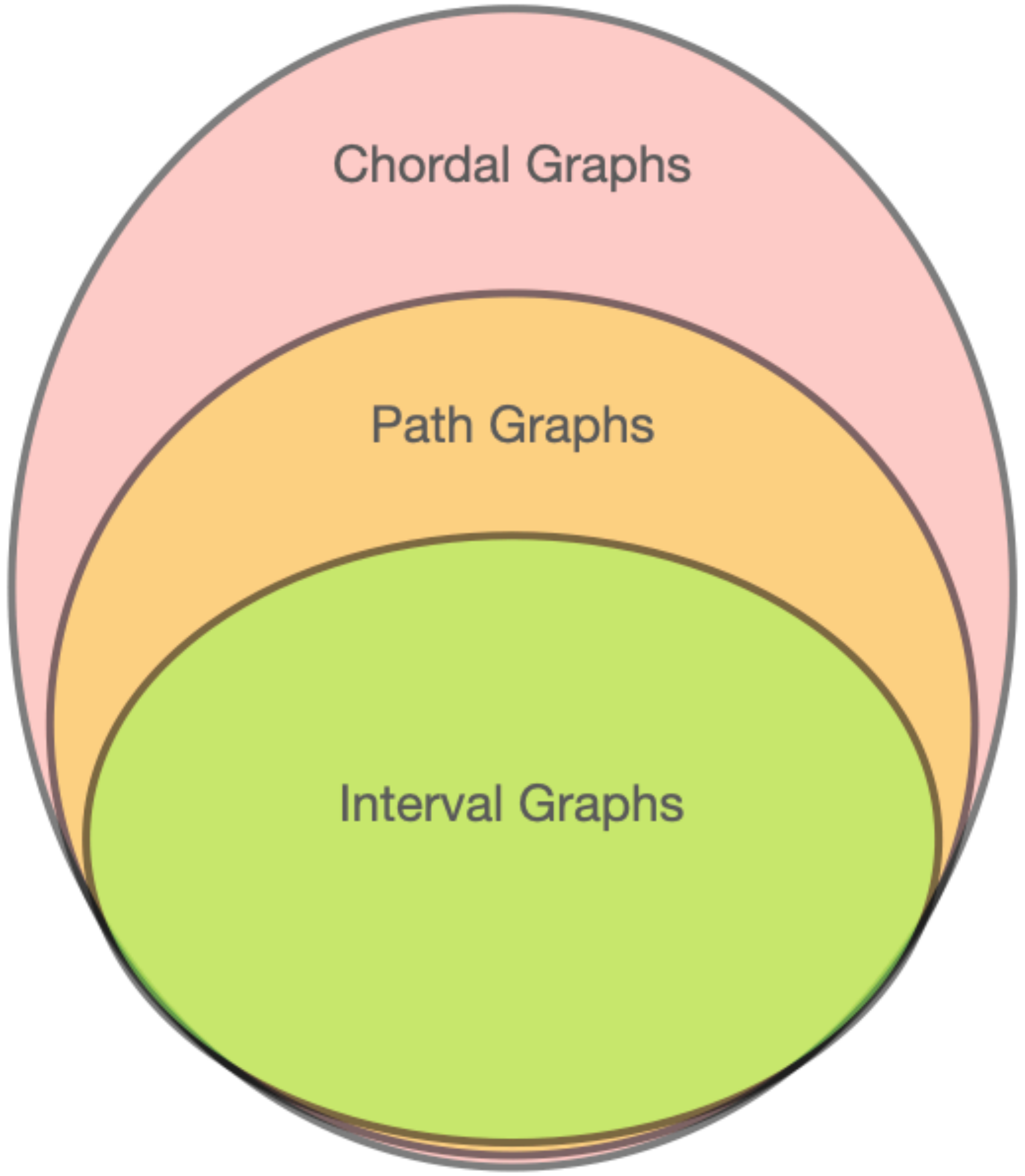
What is a succinct data structure?

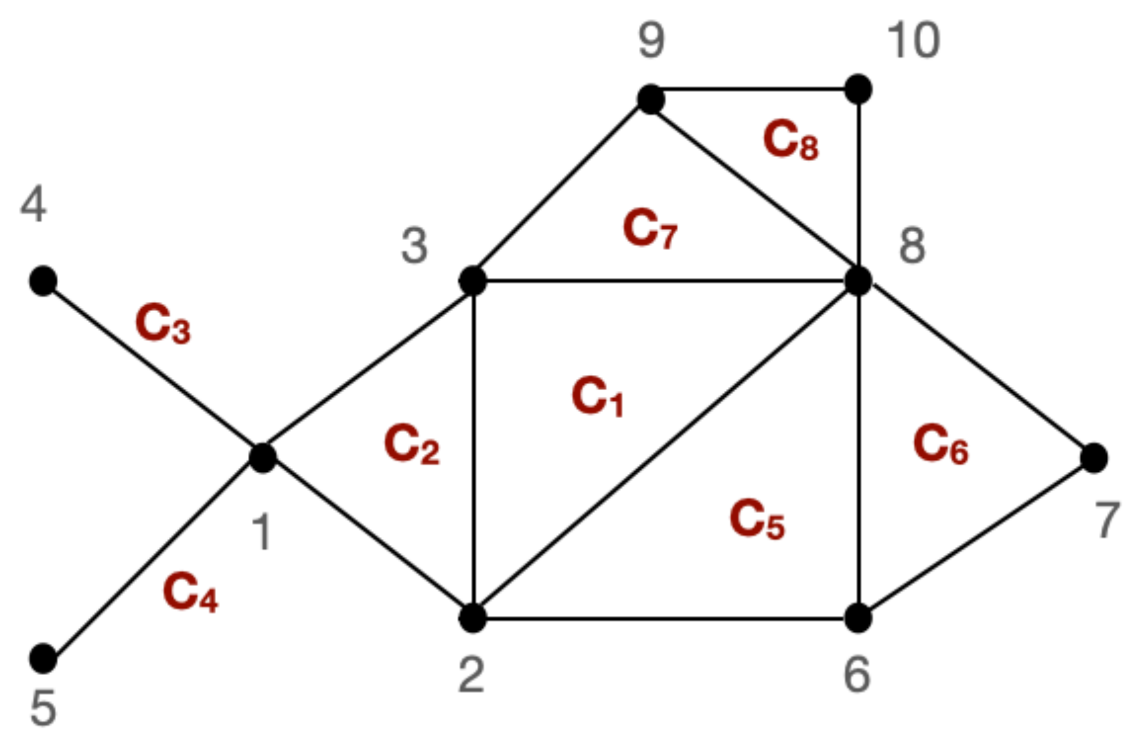
- Defined for an object in a set e.g. a graph class
- Based on *worst-case entropy* *to within a lower order terms*
- Each object in the graph class $\mathscr{C}$ to be stored using $\log|\mathscr{C}| + o(\log|\mathscr{C}|)$ bits
- Common queries like adjacency, neighbourhood to be solved efficiently.



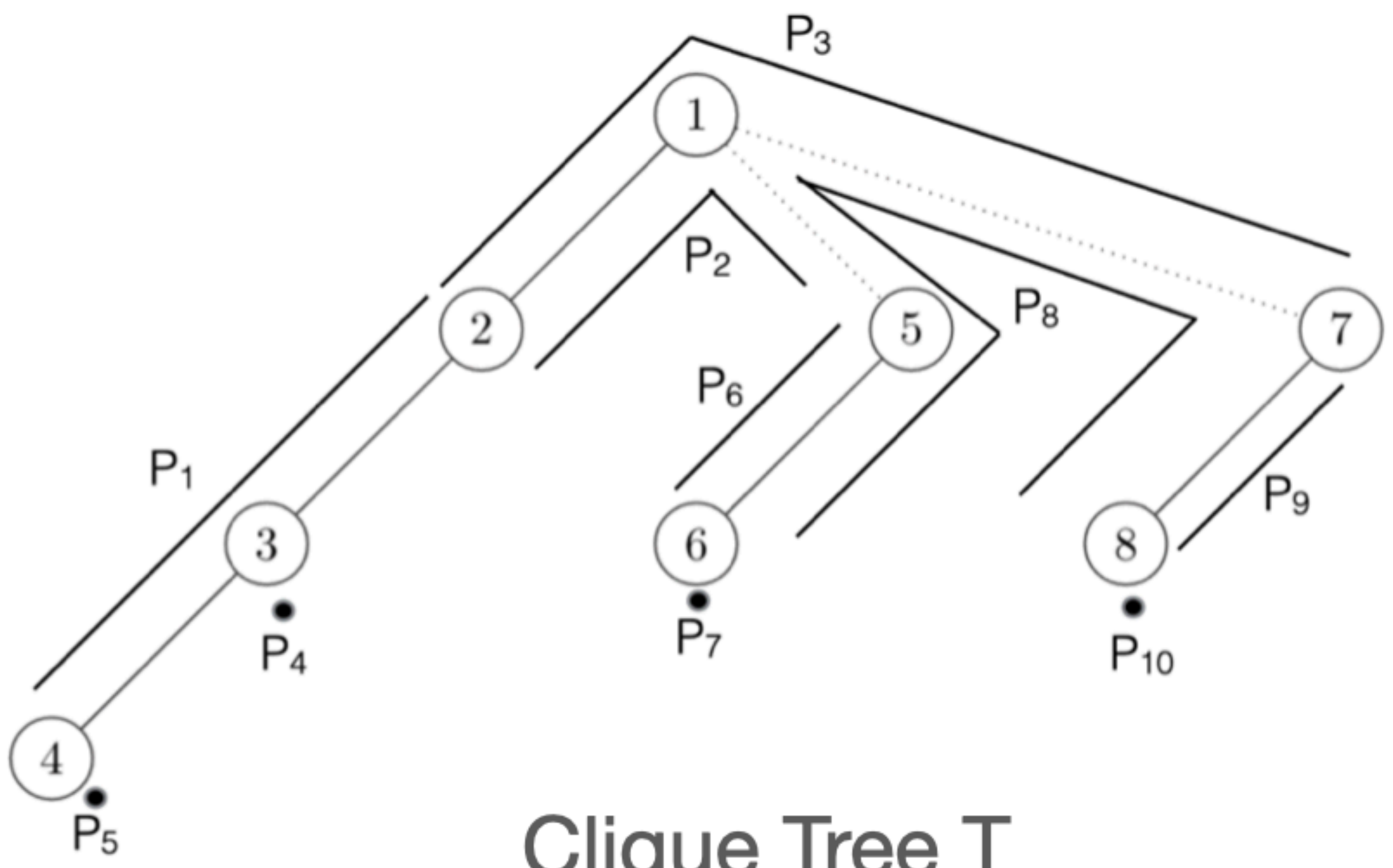
What are Path Graphs?

- Intersection graph of paths on a tree
- Clique tree T
- Relation to Class of Interval Graphs and Chordal Graphs





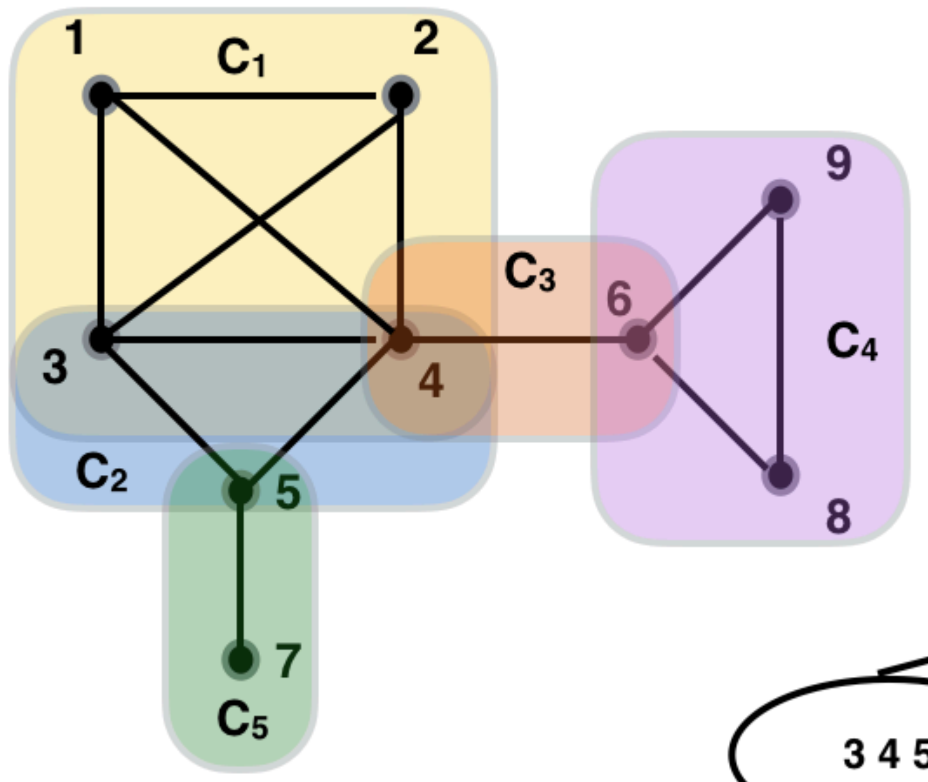
Graph G



Clique Tree T

Chordal Graphs

Interval Graphs

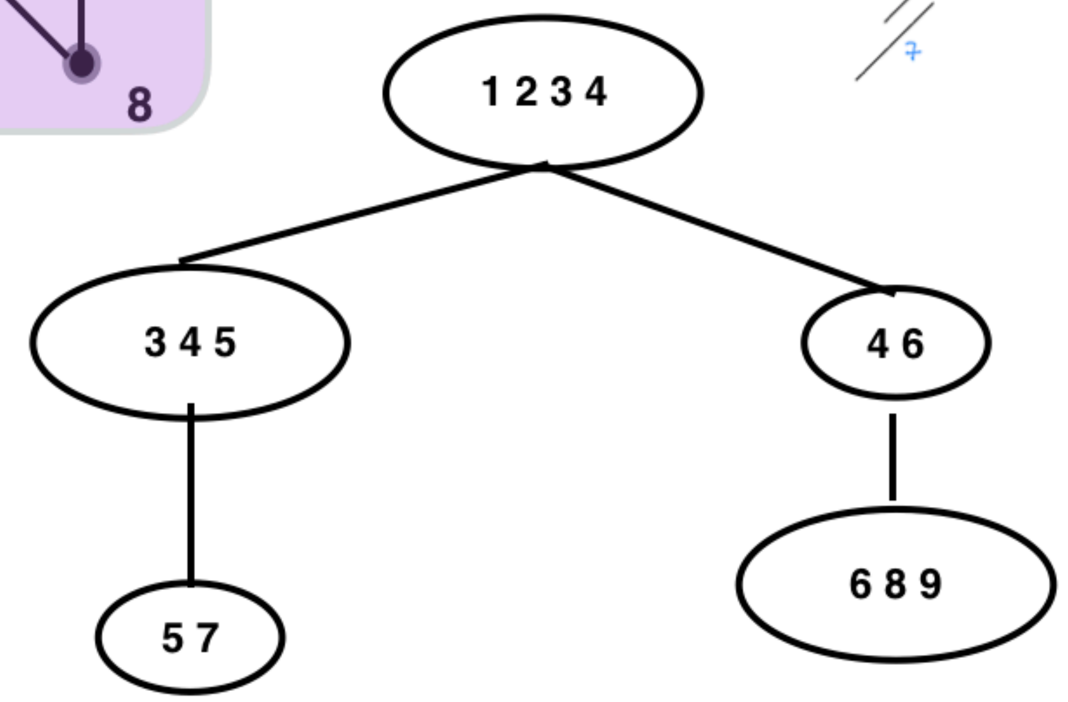


1. Chordal Graph

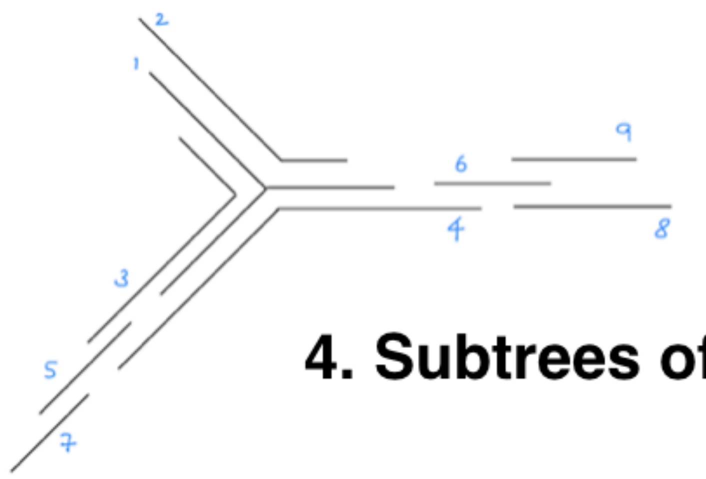
2. PEO

1,2,3,4,5,**6**,7,**8**,**9**

pred(9)={6,8}
B(9)={6,8,9}

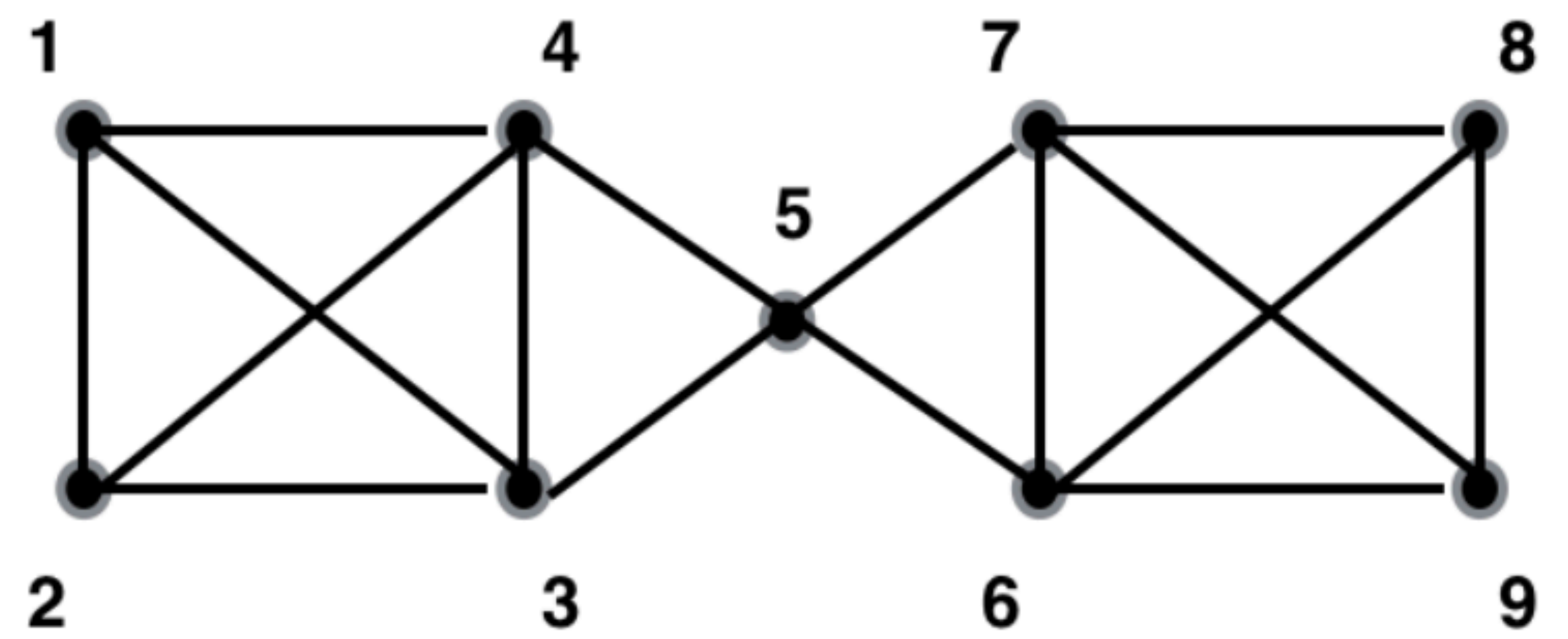


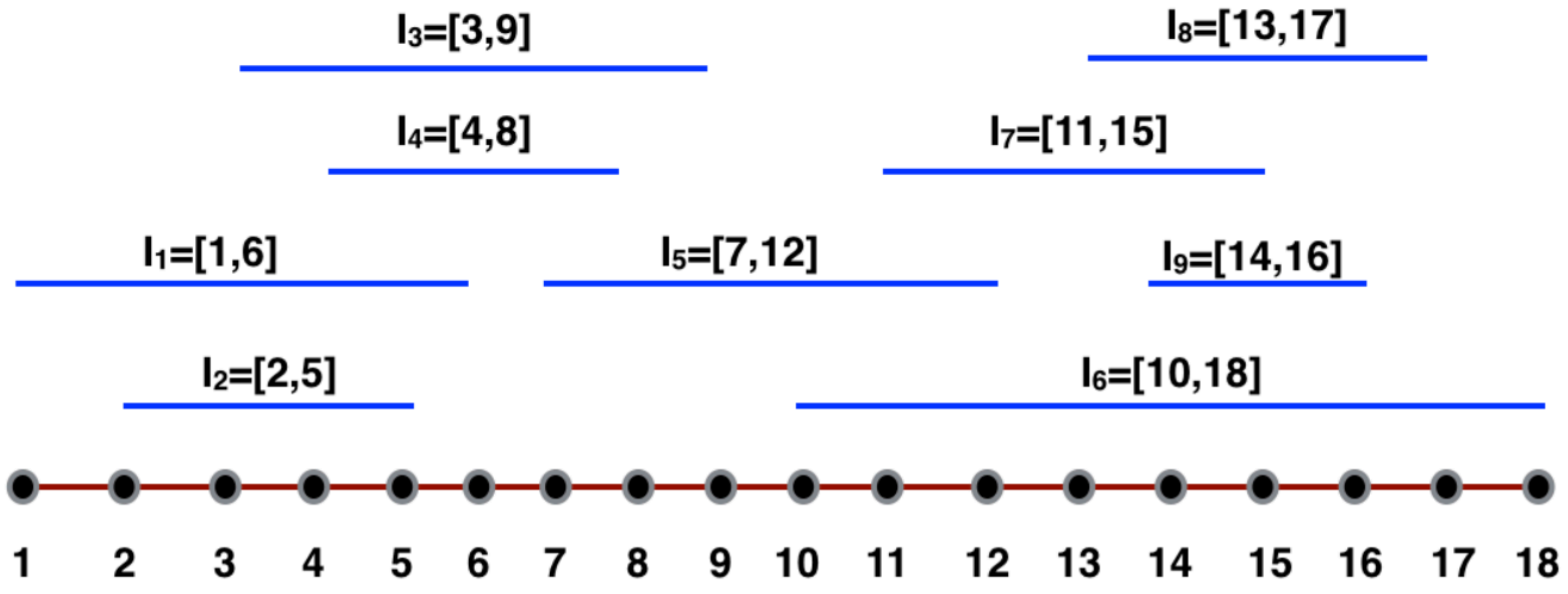
3. Clique Tree



4. Subtrees of a Tree

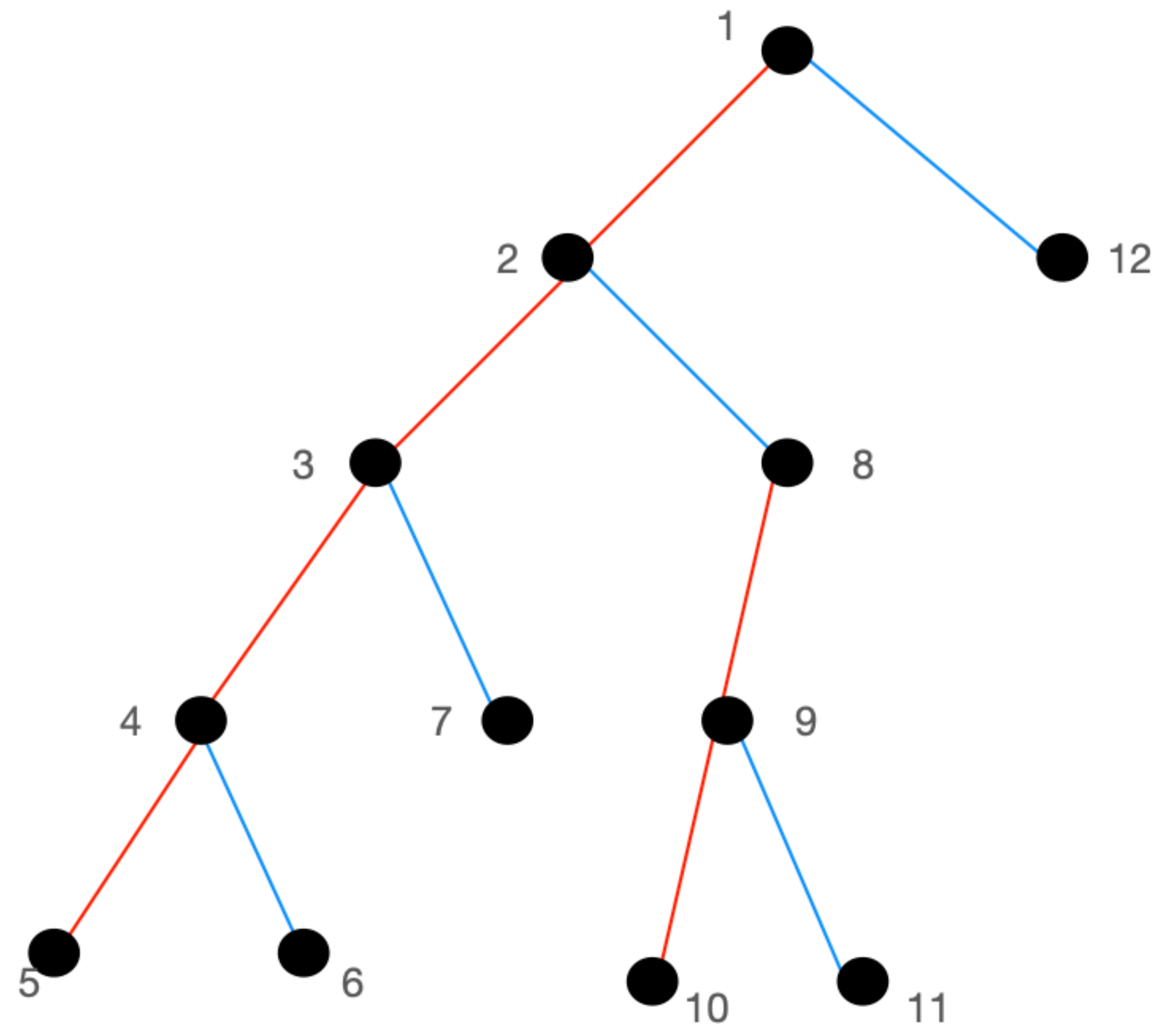
1. Interval Graph G



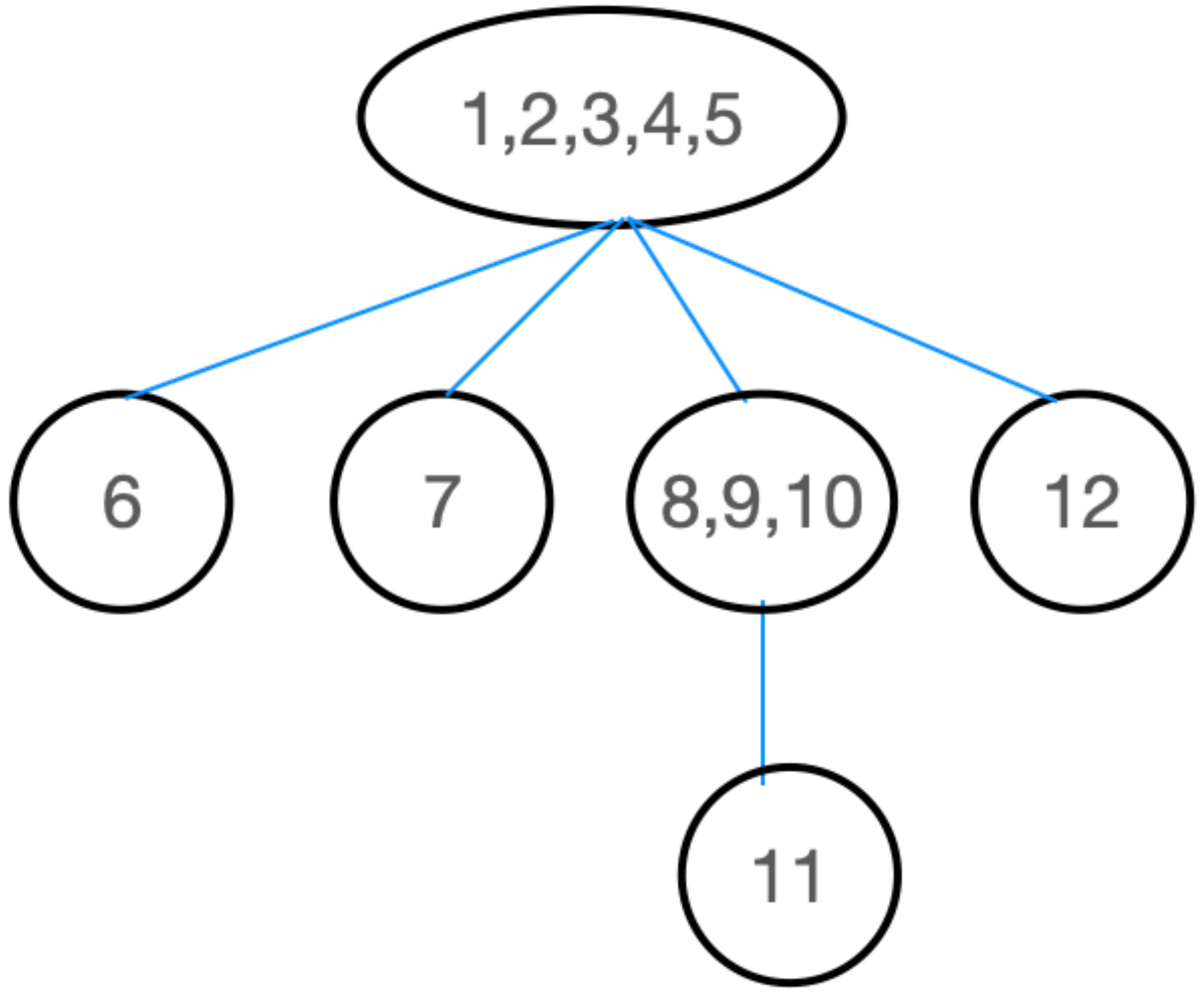


2. Interval Representation

Heavy Path Decomposition



Heavy Path Decomposition



Heavy Path Tree $\mathscr{T}$

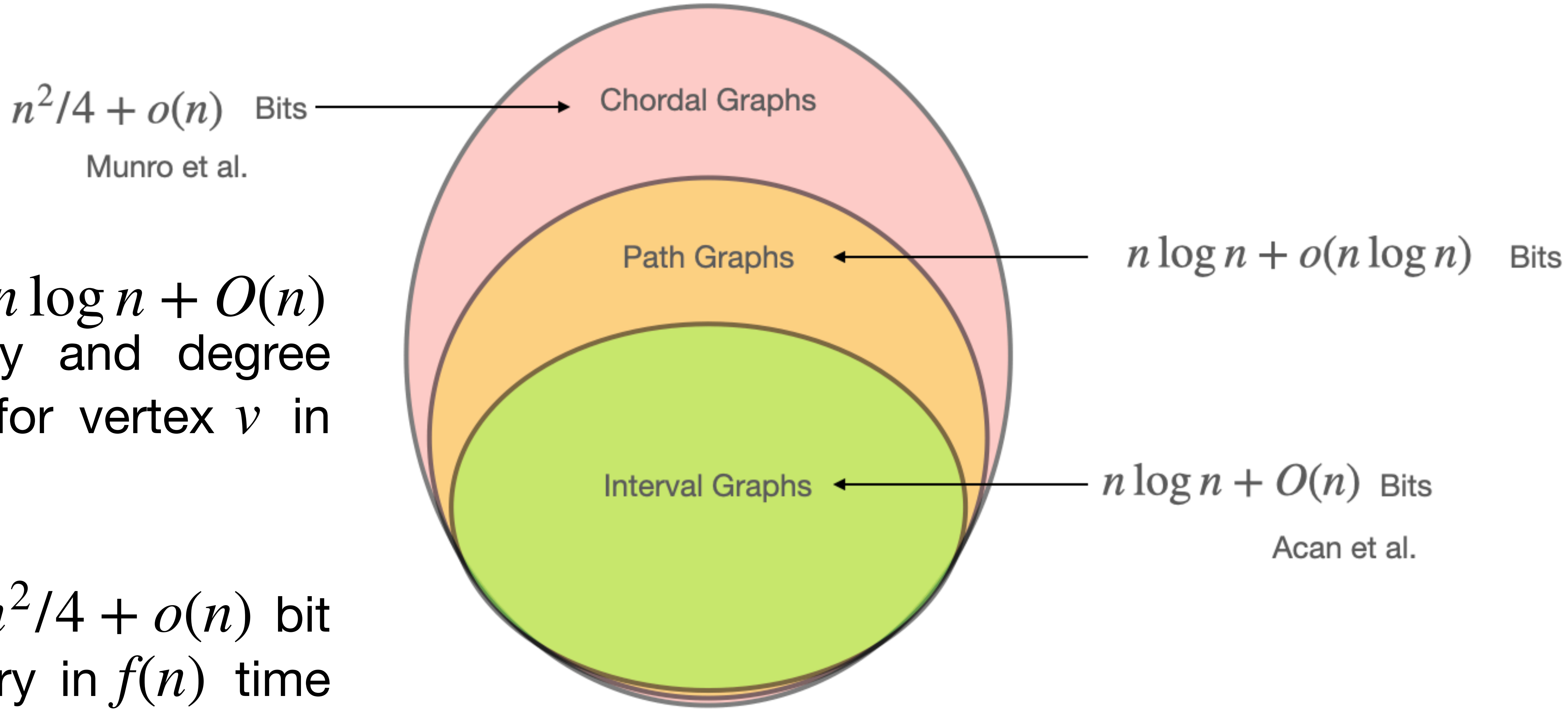
[ST] D. D. Sleator and R. E. Tarjan, “A data structure for dynamic trees,”Proceedings ofthe Thirteenth Annual ACM Symposium on Theory of Computing, p. 114–122, 1981.

Motivation

Motivation 1

- **THEOREM:** Interval graphs with n vertices have an $n \log n + O(n)$ bit succinct representation that supports adjacency and degree queries in constant time and neighbourhood query for vertex v in $O(d)$ time where d is the degree of v. [HSSS]

- **THEOREM:** Chordal graphs with n vertices have an $n^2/4 + o(n)$ bit succinct representation that supports adjacency query in $f(n)$ time where $f(n) \in \omega(1)$, degree of a vertex in $O(1)$ time and neighbourhood query in $(f(n))^2$ time per neighbour. [MW]



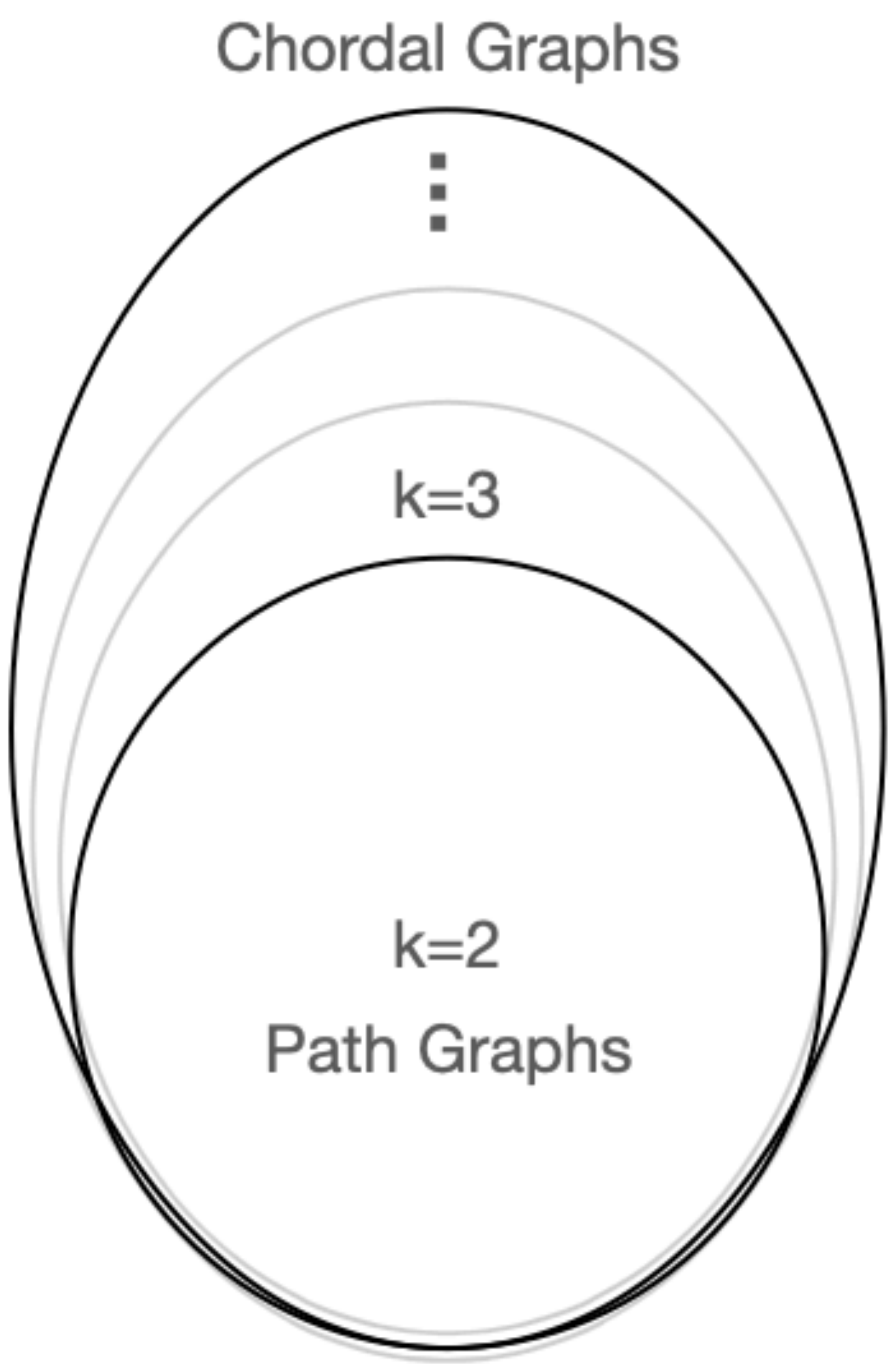
[HSSS] Succinct Data Structures for Families of Interval Graphs, Acan H., Chakraborty S. and Jo S., Satti S.R., Algorithms and Data Structures. WADS 2019. Lecture Notes in Computer Science, vol 11646. Springer, 2019.

[MW] Succinct Data Structures for Chordal Graphs, J. Ian Munro and Kaiyu Wu, 29th International Symposium on Algorithms and Computation (ISAAC 2018), Article No. 67, pp,67:1-67:12.

Motivation 2

- The *vertex leafage* $vl(G)$ of a chordal graph G is the smallest number k such that there exists a tree model of G in which every sub-tree has at most k leaves. [SJ]

- From succinct representation for $\mathscr{G}_2$(path graphs) to space-efficient representation for $\mathscr{G}_k, k \geq 3$.



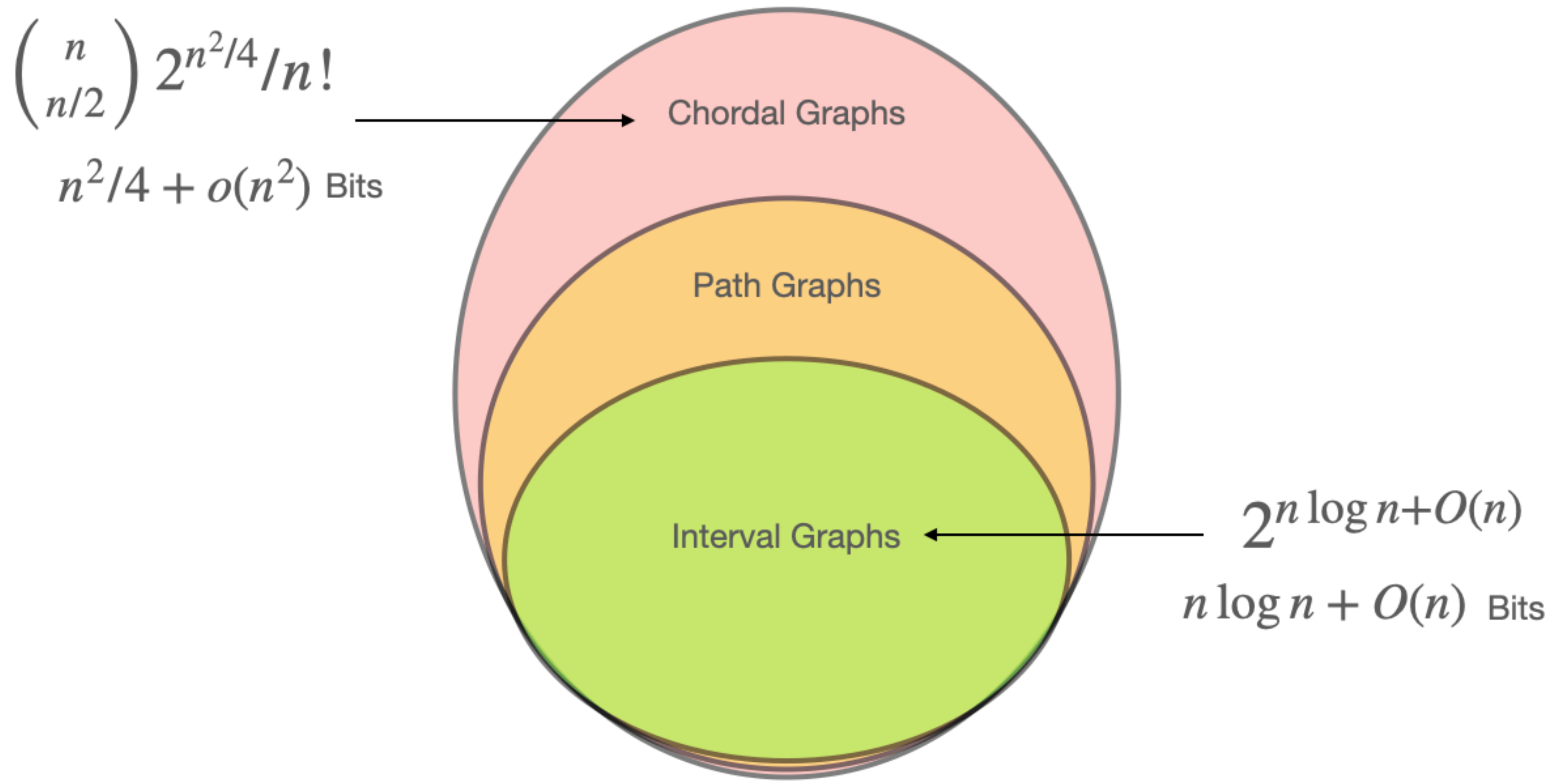
[SJ] The vertex leafage of chordal graphs, S Chaplick and J Stacho, Discrete Appl. Math., vol. 168, pp. 14–25, May 2014.

Our Results

Succinct Data Structure Result

- **THEOREM:** Path graphs with n vertices have an $n \log n + o(n \log n)$ bit succinct representation that can answer the adjacency query in $O(\log n)$ time, and the neighbourhood and degree queries for vertex v in $O(d \log^2 n)$ time where d is the degree of the vertex v.

- Is $n \log n + o(n \log n)$ bit representation succinct?

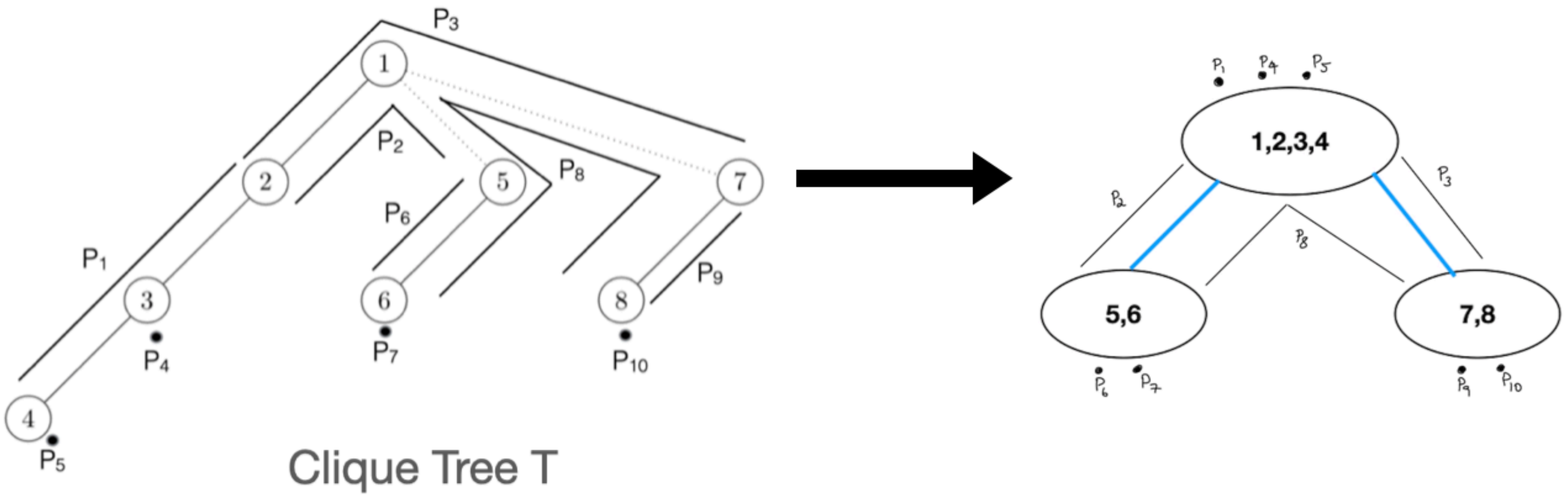


Agenda

- Problem Statement
- Introduction
- Motivation
- Our Results
- Construction
- Queries
  - Adjacency Query
  - Neighbourhood Query
- Conclusion

Construction

- Perform Heavy Path Decomposition
- Align heavy edge as left most
- Label the nodes of Clique Tree T using Preorder



Why heavy path decomposition?

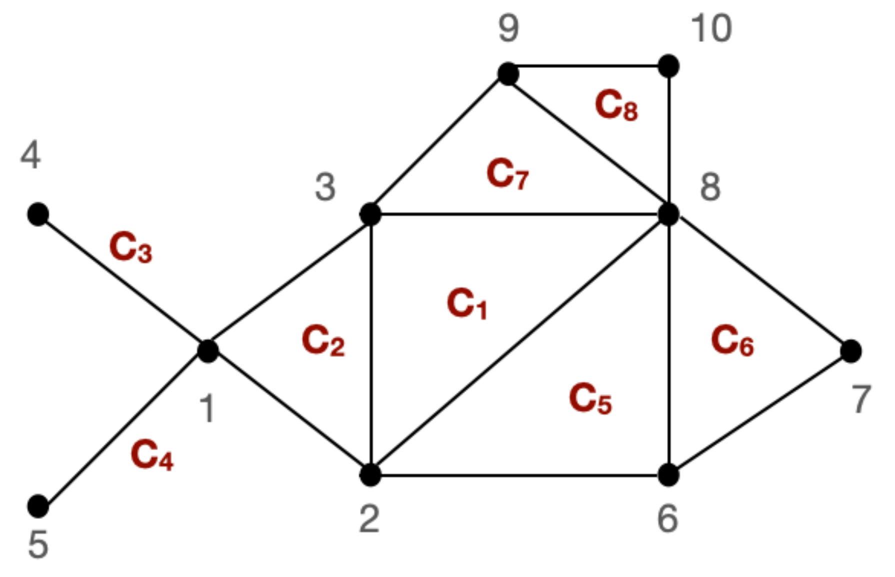
- Heavy sub-paths have contiguous numbering
- Heavy path tree has $\lceil \log n \rceil$ levels
- Each path P can be divided into a sequence of $O(\log n)$ heavy sub-paths and light edges i.e. $P^1, \ldots, P^k$ (**LEMMA 19**)

[G] F. Gavril, "A recognition algorithm for the intersection graphs of paths in trees," 1978

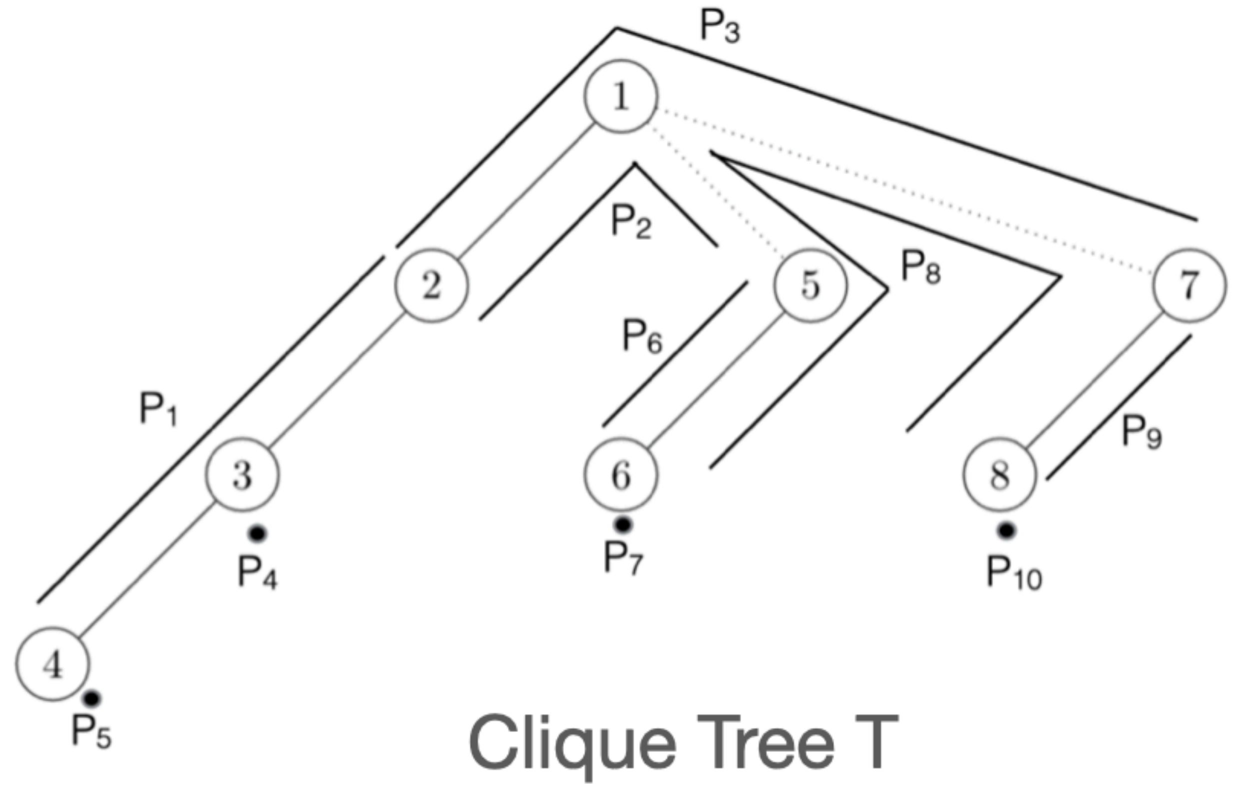
Space Complexity

Input is $(T, P_1, \ldots, P_n)$ obtained using Gavril's Method [G]
Each path is of the form $P_i \equiv (l_i, r_i), 1 \le i \le n$

- Three Components of Succinct Representation
  - Clique tree T stored using $2n + o(n)$ bit [NS]
  - Sorted $l_i, 1 \le i \le n$ are stored as bit string F using differential encoding taking $2n + o(n)$ bits [RSS]
  - $r_i, 1 \le i \le n$ of paths is stored in a Wavelet tree S using $n \log n + o(n \log n)$ bits [MN]
- Space complexity is $n \log n + o(n \log n)$ bits



Graph G



Clique Tree T

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
l_i	2	2	2	3	4	5	6	6	7	8
r_i	4	5	7	3	4	6	6	8	8	8

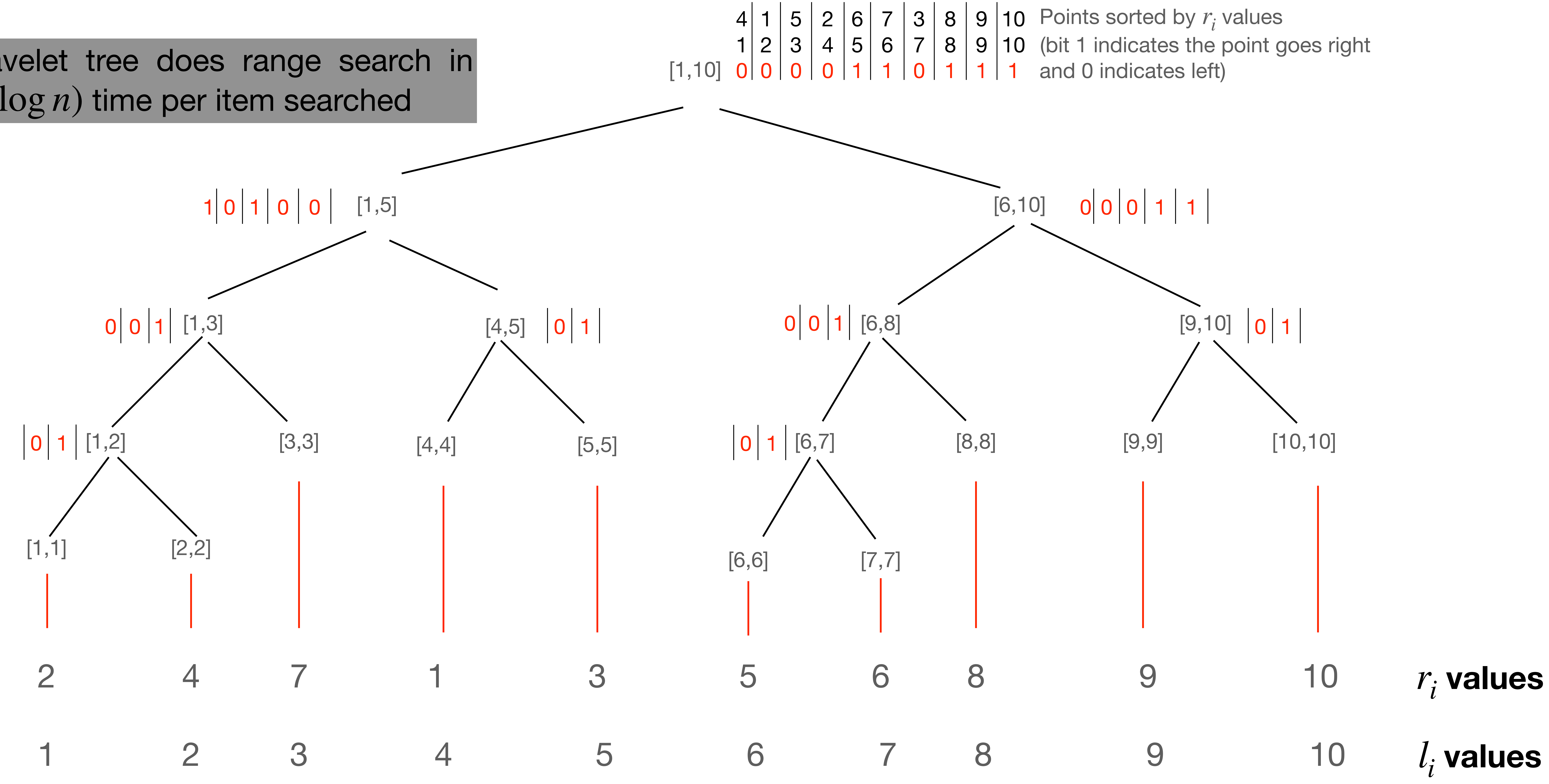
F

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0

[NS] Fully functional static and dynamic succinct trees, G. Navarro and K. Sadakan, ACM Trans. Algorithms, vol. 10, no. 3, May 2014.
[RRS] Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets, R. Raman, V. Raman, and S. R. Satti, ACM Trans. Algorithms,vol. 3, no. 4, pp. 43, 2007.
[MN] Rank and select revisited and extended, V.Makinen and G. Navarro, Theoretical Computer Science, vol.387, no.3, pp. 332-347, 2007.

Wavelet Tree

Wavelet tree does range search in $O(\log n)$ time per item searched

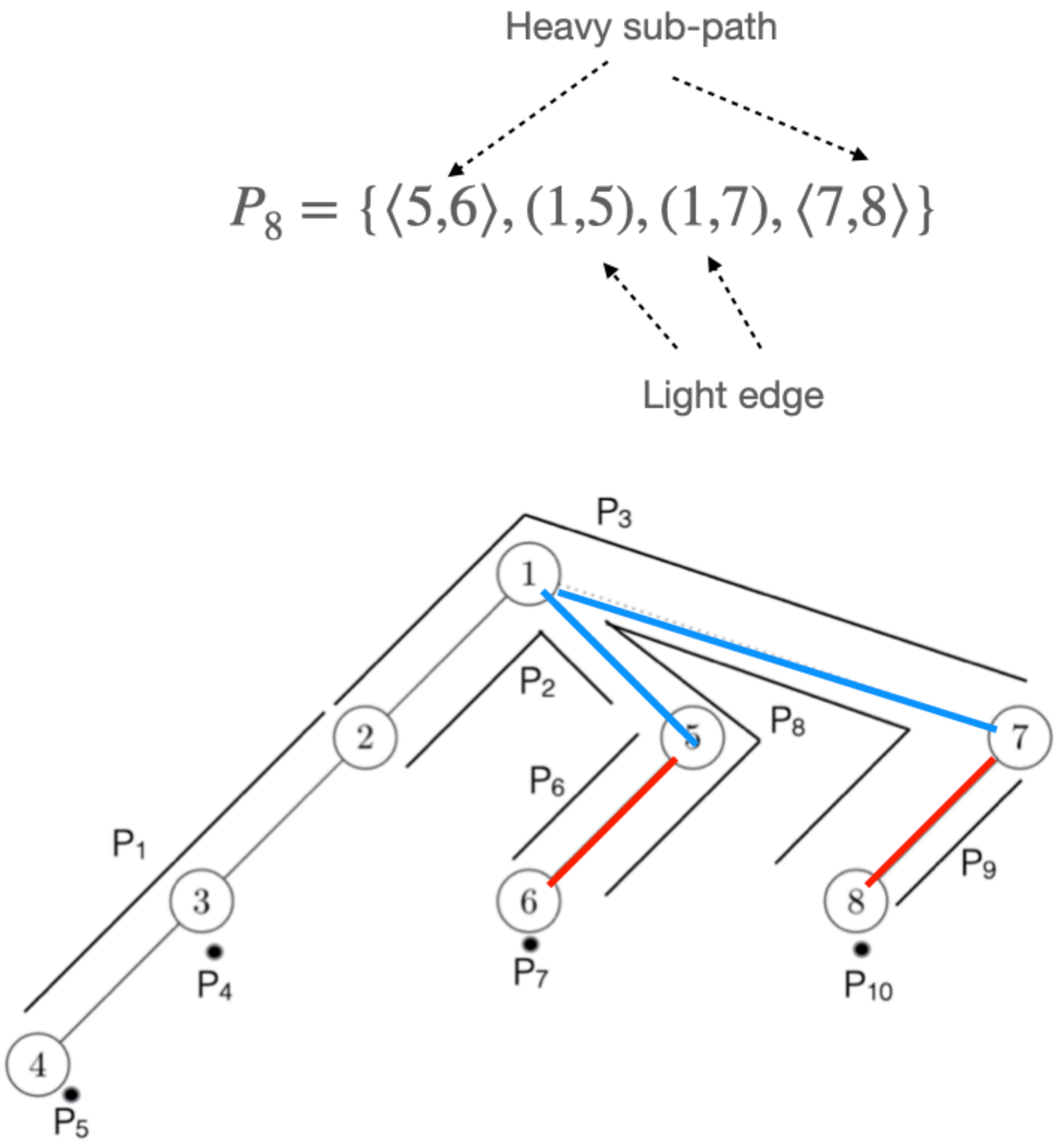


2D points to store

Queries

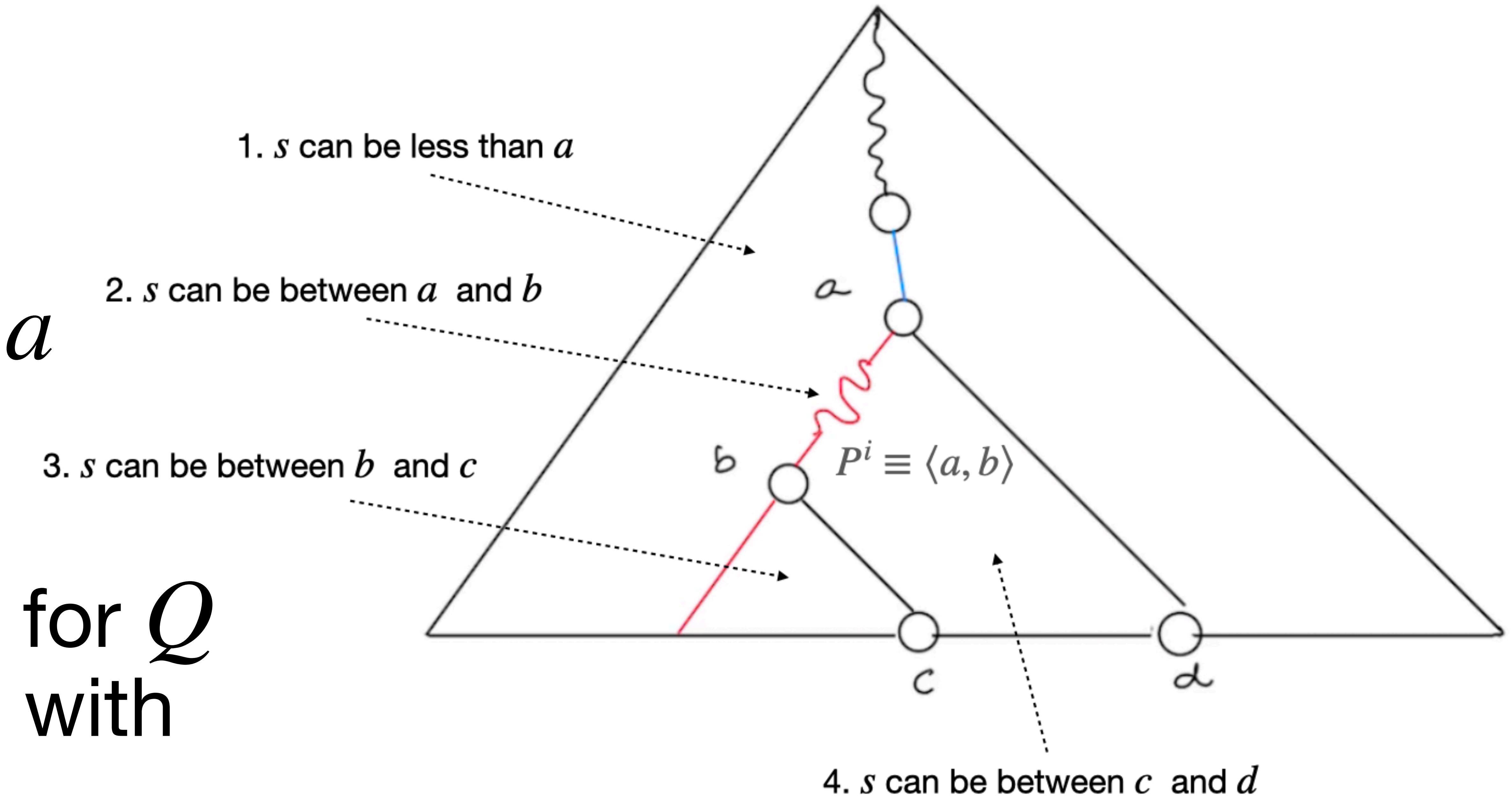
Adjacency Query

- Adjacency of paths P and Q
- Divide path P into at most $O(\log n)$ heavy sub-paths and light edges, $P^1, \ldots, P^k$
- For each P^i, check if it overlaps with Q (How to check overlap?)
- Time Complexity $O(\log n)$



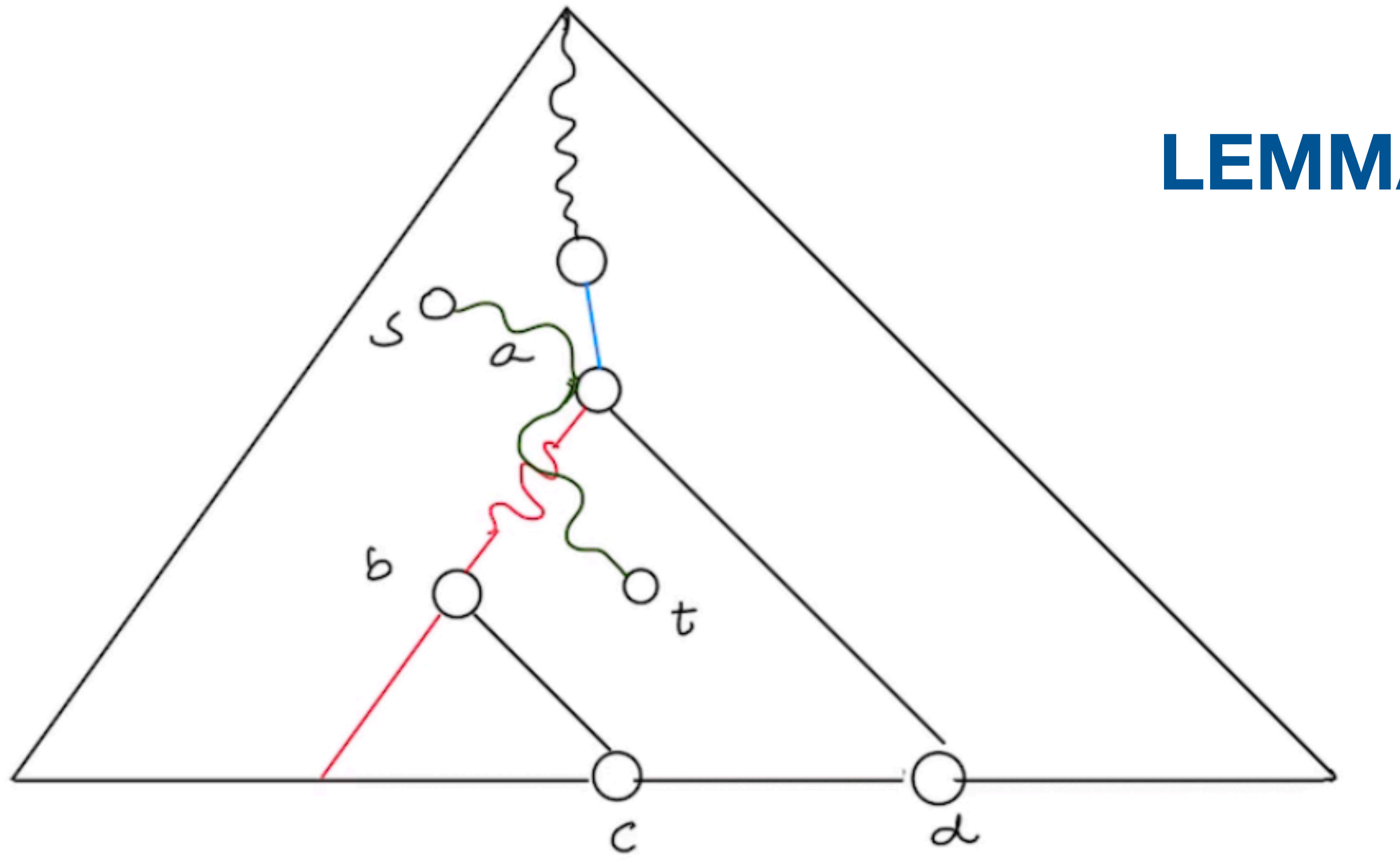
Technique for Checking Overlap

- Let $Q \equiv (s, t)$
- $P^i \equiv (a, b)$ is a heavy sub-path of P
- There are four places for s relative to a and b
- For each of these four starting points for Q specific conditions allow it to overlap with P^i

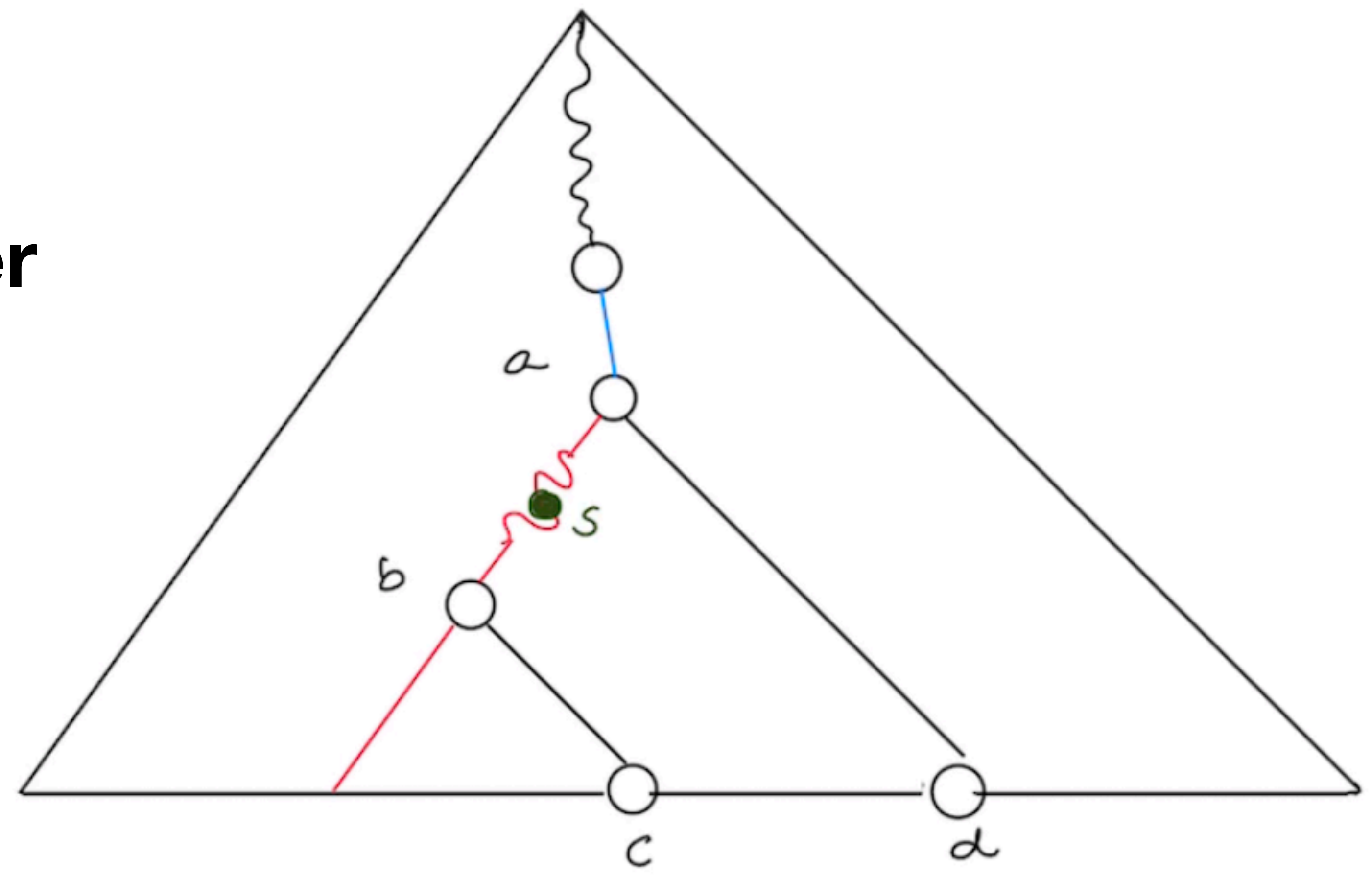


Conditions for Overlap with Heavy Sub-Path

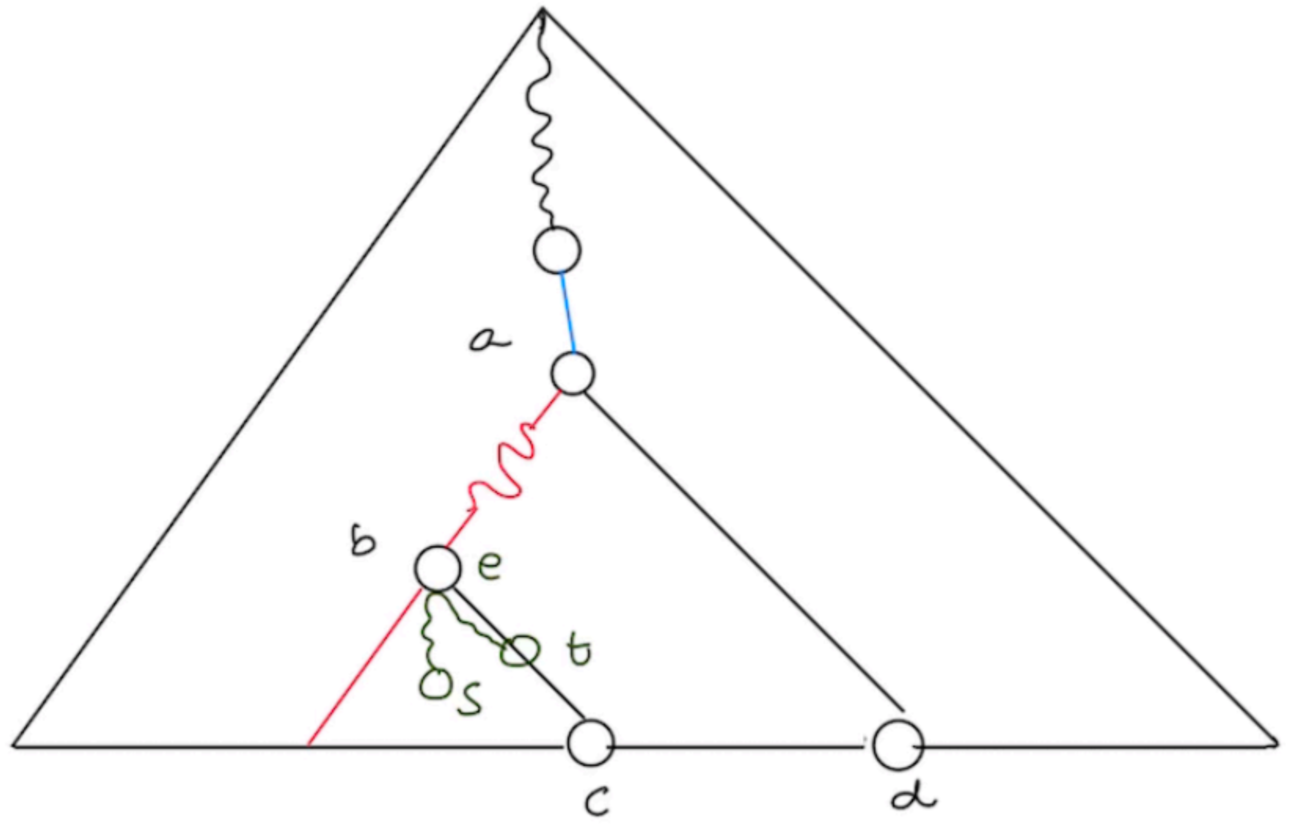
LEMMA 17 in the paper

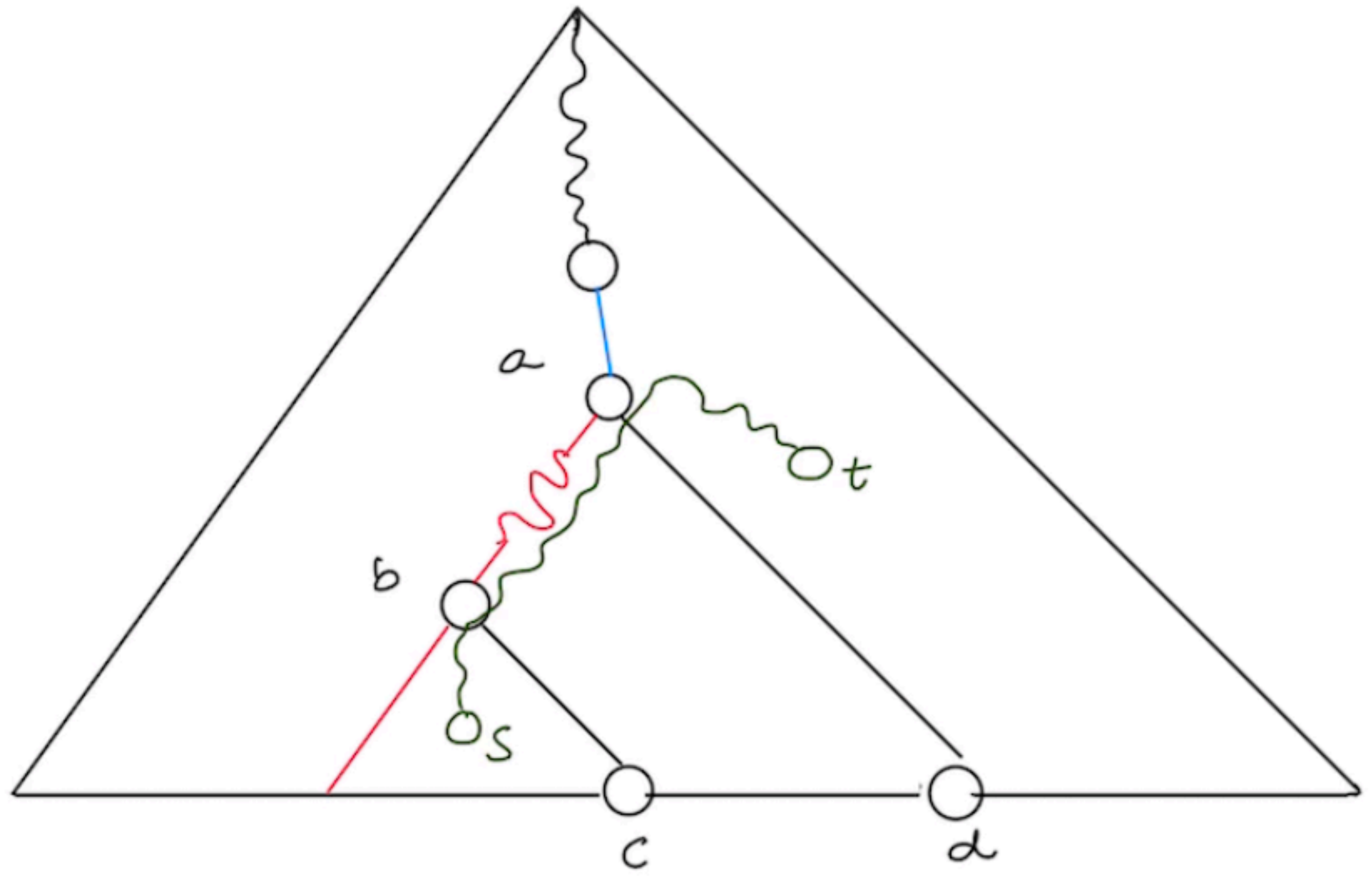


1. $s < a \wedge a \leq t \leq d$

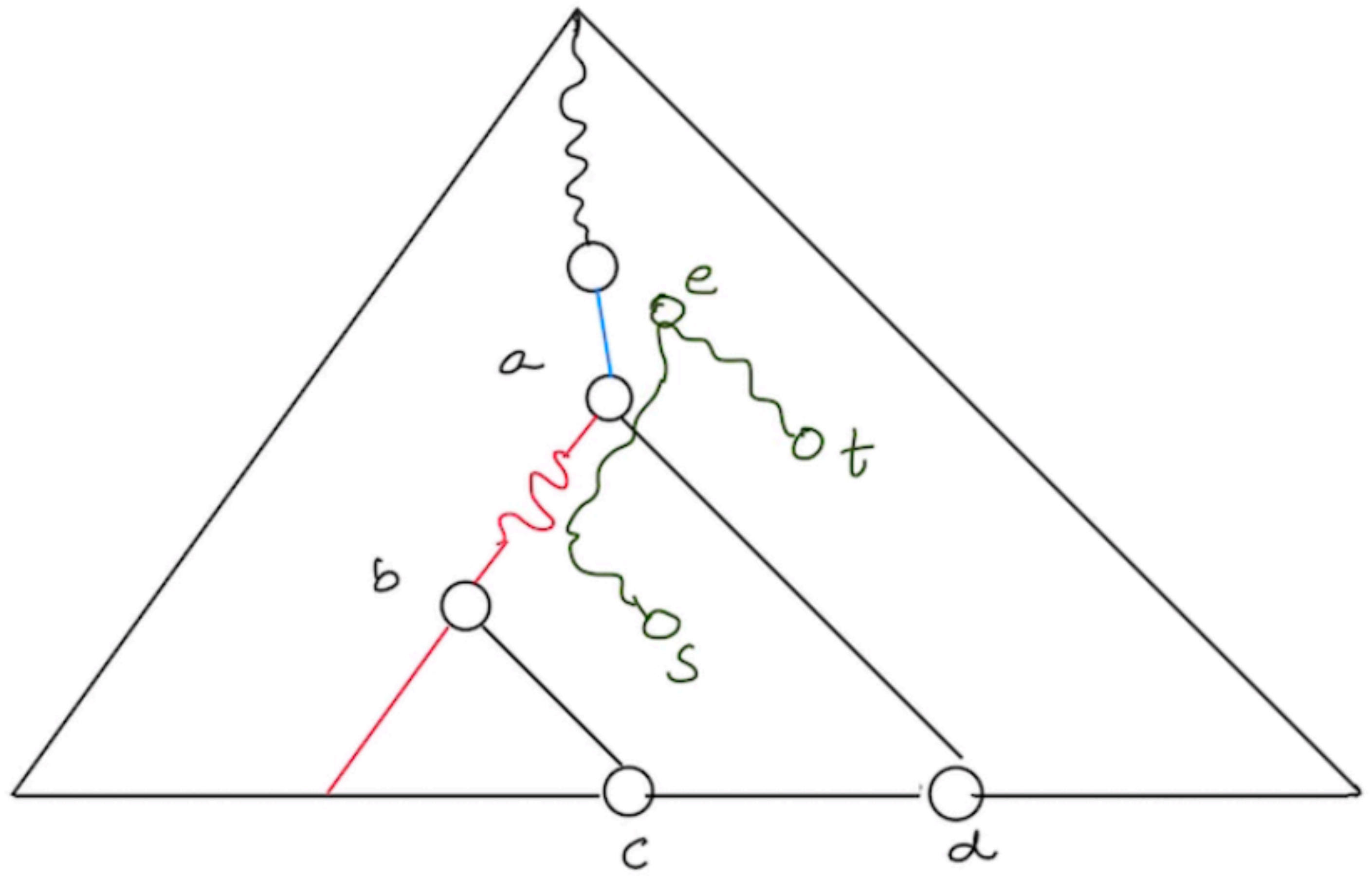


2. $a \leq s \leq b$





3. $b < s \leq c \wedge \mathsf{lca}(s, t) \leq b$

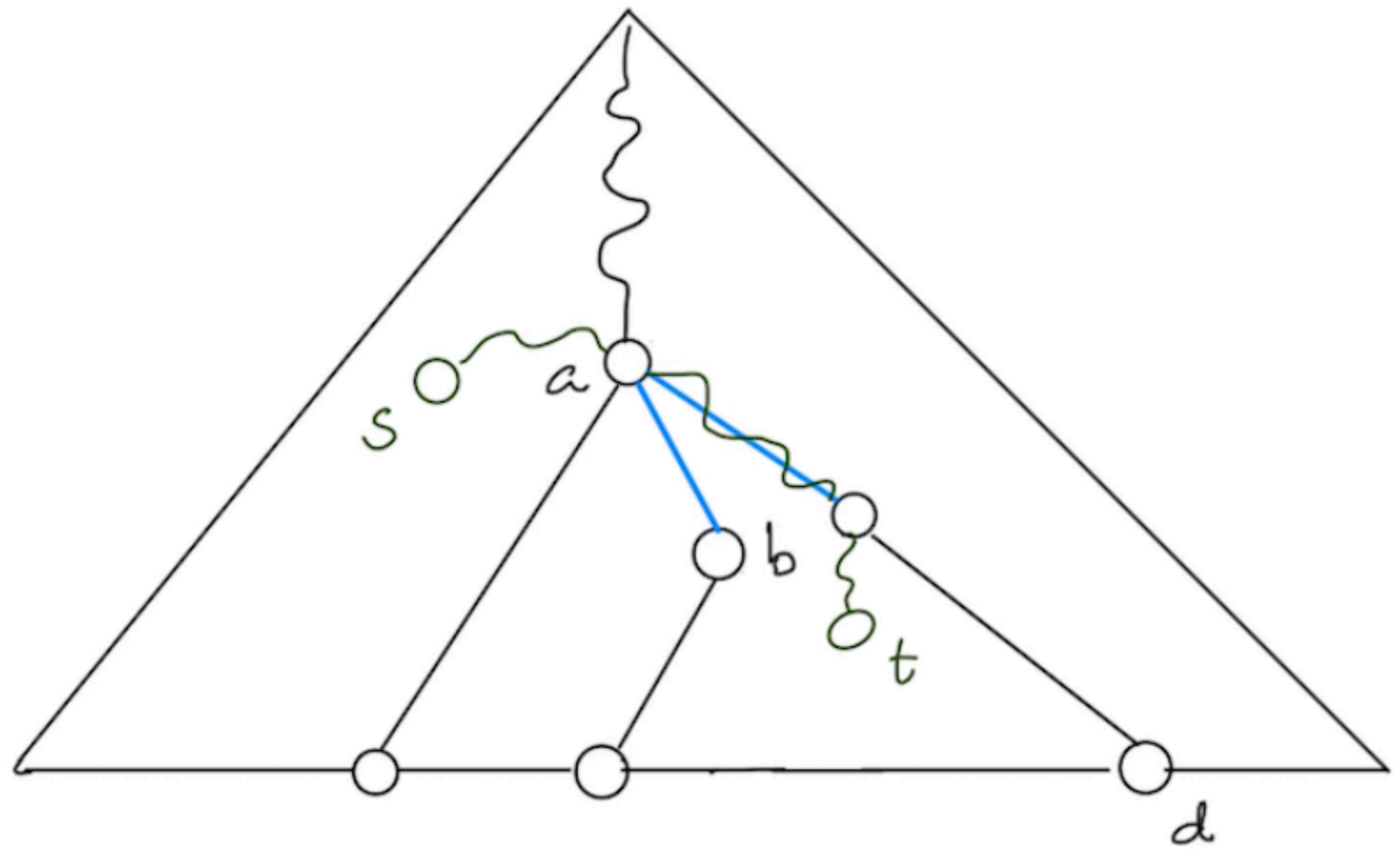


4. $c < s \leq d \wedge \mathsf{lca}(s, t) < b$

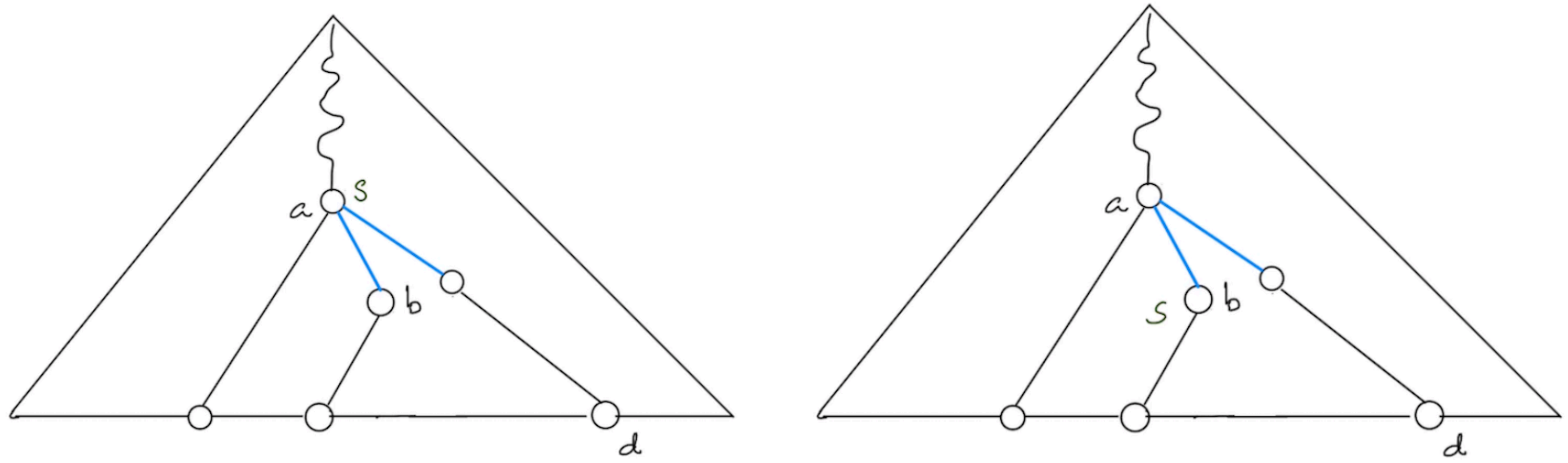
Back to Slide 26

* Consecutive numbering of heavy sub-path helps with these conditions

Conditions for Overlap with Light Edge

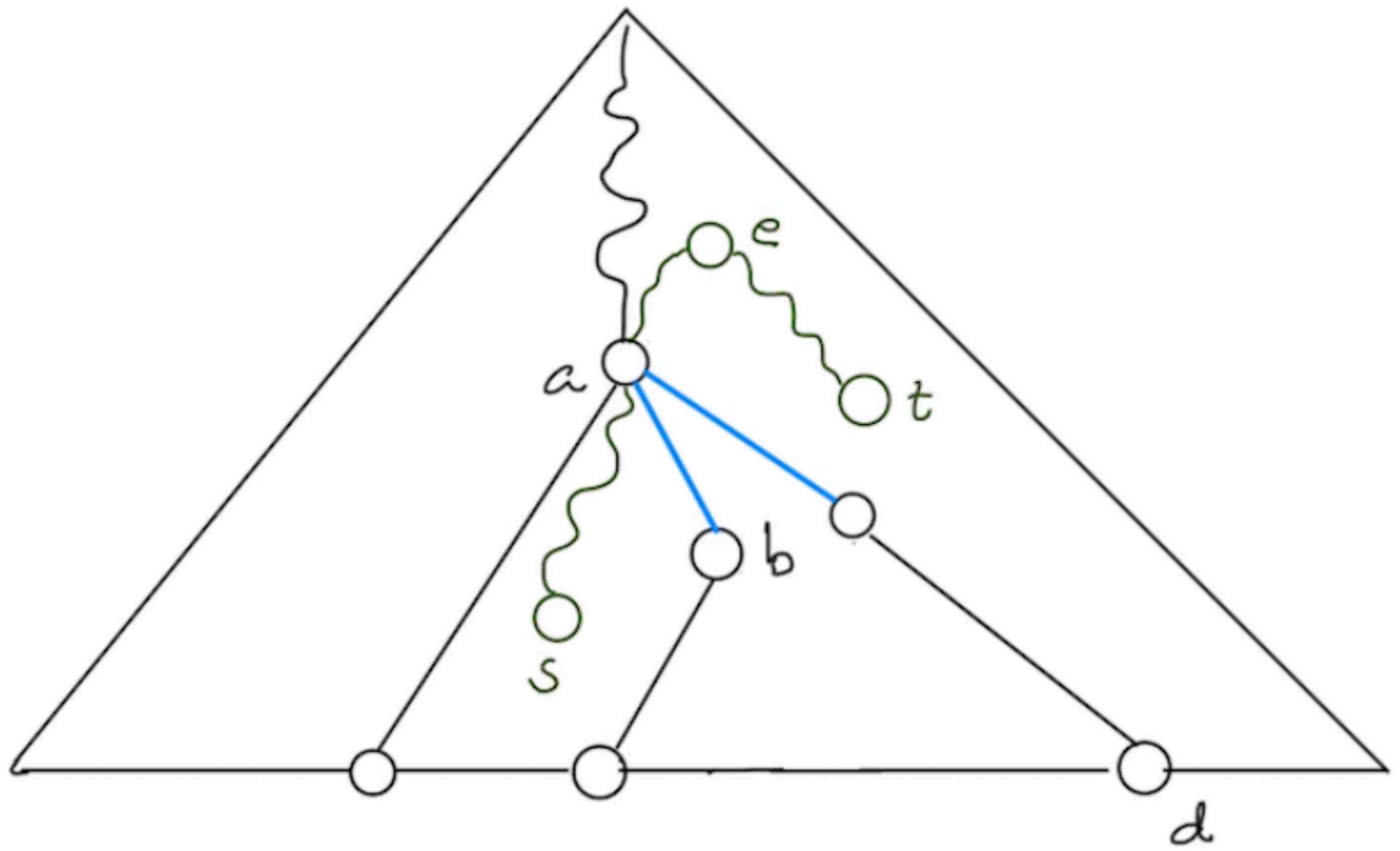


1. $s < a \wedge a \leq t \leq d$

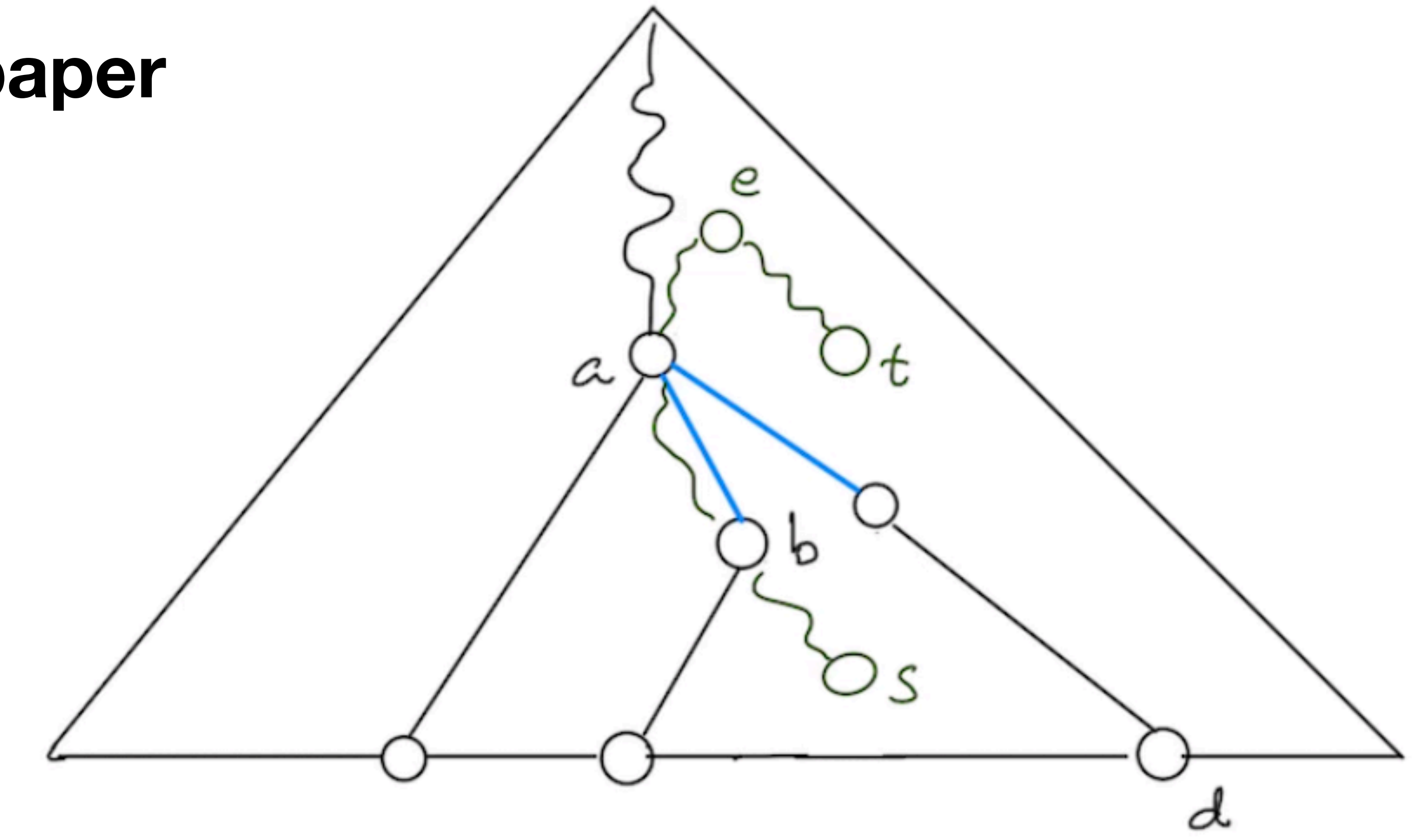


2. $s = a \vee s = b$

LEMMA 18 in the paper



3. $a < s < b \wedge \text{lca}(s, t) \leq a$



4. $b < s \leq d \wedge \text{lca}(s, t) \leq b$

Back to Slide 26

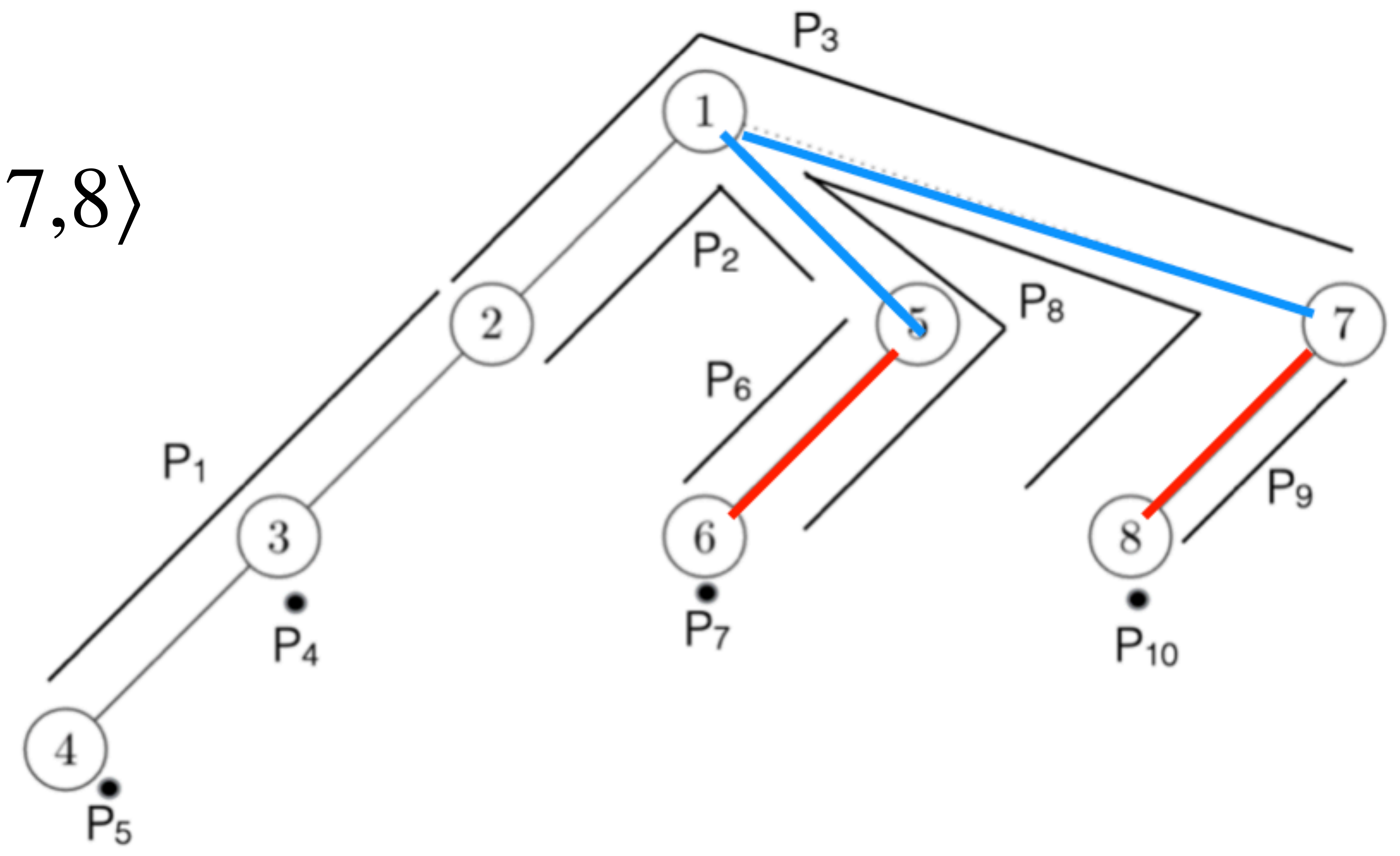
Adjacency Query - Summary

Heavy sub-path

$$P_8 = \{\langle 5,6\rangle, (1,5), \langle 1,7\rangle, (7,8)\}$$

Light edge

- Adjacency of paths P_8 and P_3
- Divide path P_8 into at most $O(\log n)$ heavy sub-paths and light edges, $P^1 = \langle 5,6\rangle, P^2 = (1,5), P^3 = (1,7), P^4 = \langle 7,8\rangle$
- For each P^i, check if it overlaps with P_3



Neighbourhood Query

- Divide path P into at most $O(\log n)$ heavy sub-paths and light edges, $P^1, \ldots, P^k$
- For each $P^i \equiv (a, b)$ enumerate the paths that overlap it
- Enumeration of paths is done by issuing range queries on Wavelet tree
- Range queries are designed based on Conditions used for adjacency
- E.g. $s < a \wedge a \leq t \leq d$ will translate into “All paths with l_i in range [1,a-1] and r_i in the range [a, d]”
- Range queries take $O(\log n)$ time per point identified.
- Each P^i takes $O(d \log n)$ time, so total time is $O(d \log^2 n)$.

Example Orthogonal Range Search

- Consider condition $s < a \wedge a \leq t \leq d$ same as $[1, a - 1] \times [a, d]$
- Pick internal node $[z, z']$ of wavelet tree only if there is a path in that range with $a \leq r_i \leq d$
- How to check if $a \leq r_i \leq d$ optimally [M]
  - Using Range Minimum and Maximum Query on r_i values of paths [HSSS]
  - Let minimum and maximum value in range $[z, z']$ be r_{min} and r_{max}
  - If $r_{min} > d$ or $r_{max} < a$ then ignore the range $[z, z']$

[M] S. Muthukrishnan, “Efficient algorithms for document retrieval problems,” Proceedings of ACM-SIAM SODA, 2002, pp. 657–666.
[HSSS] Succinct Data Structures for Families of Interval Graphs, Acan H., Chakraborty S. and Jo S., Satti S.R., Algorithms and Data Structures. WADS 2019. Lecture Notes in Computer Science, vol 11646. Springer, 2019.

Agenda

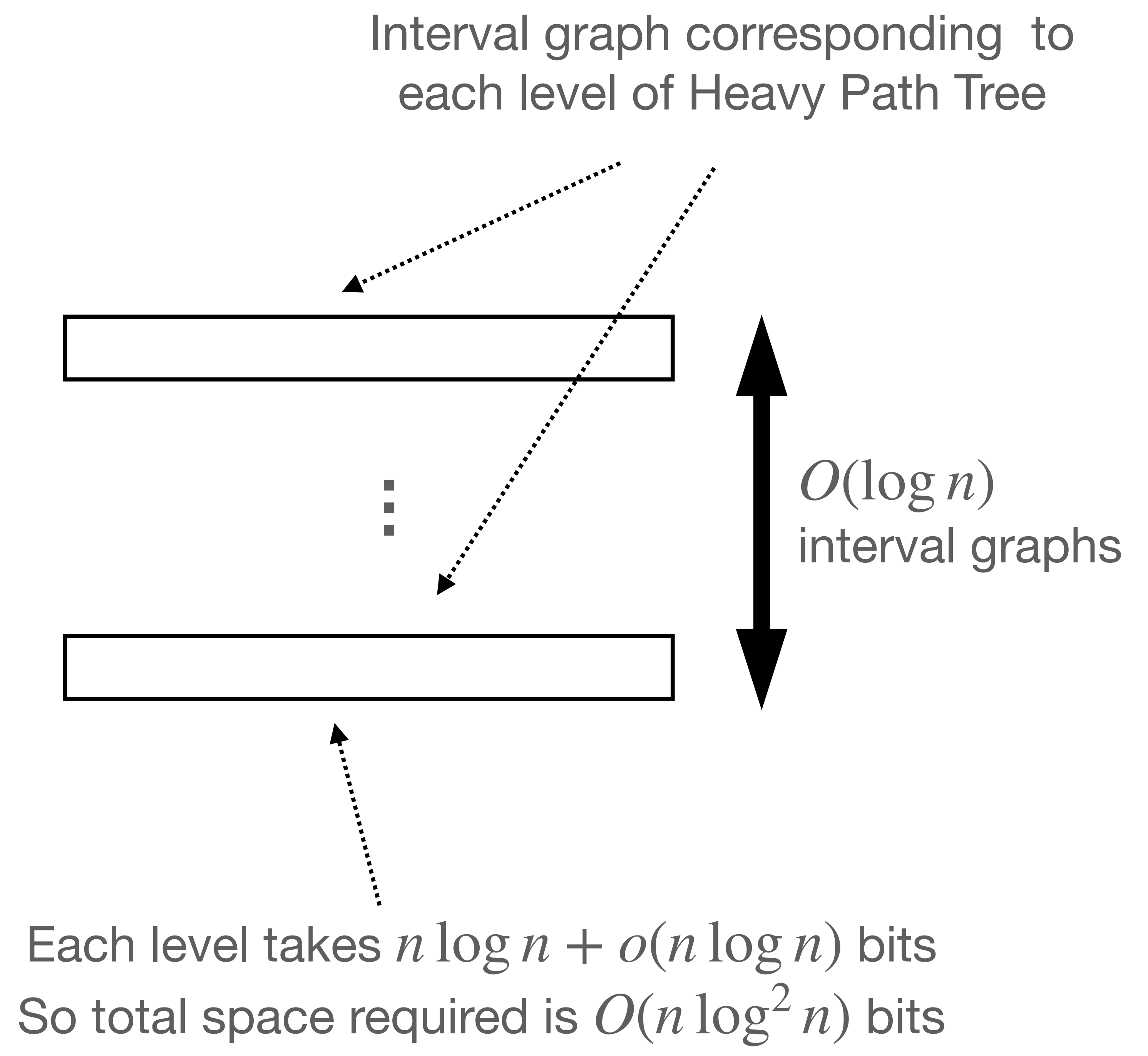
- Problem Statement
- Introduction
- Motivation
- Our Results
- Construction
- Queries
    - Adjacency Query
    - Neighbourhood Query
- Conclusion

Important Ideas

- Heavy Path Decomposition
  - Gives the $\lceil \log n \rceil$ level tree
  - Contiguous numbering on heavy paths
  - Both queries rely on dividing paths into $O(\log n)$ heavy sub-paths and light edges
  - A total ordering on heavy paths and light edges
- Orthogonal Range Search (Wavelet Tree)
  - Allows searching using $n \log n + o(n \log n)$ bit succinct data structure
  - Augmented method of searching (Range Minimum and Maximum Query) gives optimal search

Additional Results

- **THEOREM:** There exists a space-efficient representation for path graphs with n vertices using $O(n \log^2 n)$ bits that can answer the adjacency and degree queries in $O(1)$ time and the neighbourhood query for vertex v in $O(d)$ time where d is the degree of vertex v.
- **THEOREM:** For chordal graphs with vertex leafage k there exists a $(k-1)n \log n + O(n)$ bit space-efficient representation that can answer adjacency query in $O(k^2 \log n)$ time.



Summary

Graph Class	Succinct Representation	Space Complexity	Adjacency Query	Neighbourhood Query	Degree Query	Reference
Chordal Graphs	Y	$n^2/4 + o(n)$	$f(n) \in \omega(1)$	$f(n)^2$	$O(1)$	[MW]
Interval Graphs	Y	$n \log n + O(n)$	$O(1)$	$O(d)$	$O(1)$	[HSSS]
Path Graphs	Y	$n \log n + o(n \log n)$	$O(\log n)$	$O(d \log^2 n)$	$O(d \log^2 n)$	
Path Graphs	N	$O(n \log^2 n)$	$O(1)$	$O(d)$	$O(1)$	