Interval Stabbing Problems in Small Integer Ranges

Jens M. Schmidt

Outline

1. Problem Definitions

2. Data Structure

Interval Stabbing

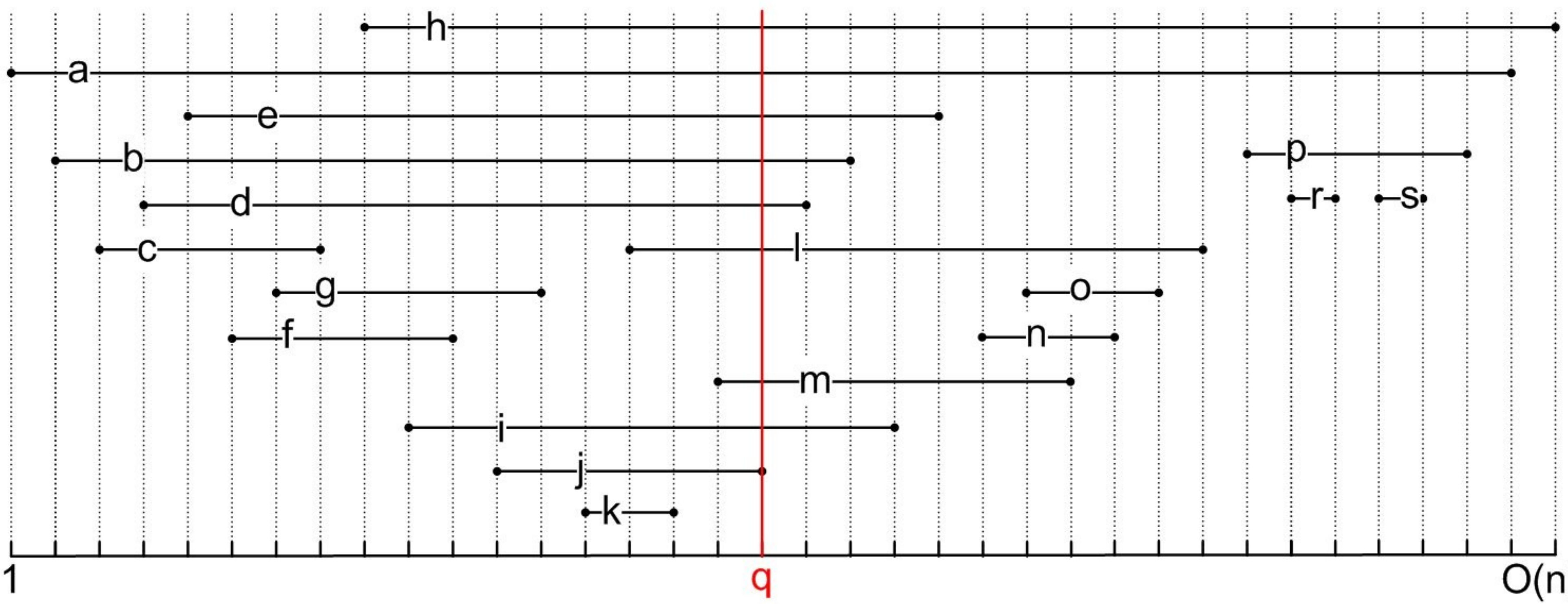
- ***I*** = set of *n* intervals $[l_i, r_i]$ with $l_i \leq r_i$

Stabbing query on a value *q:*

- Asks for all intervals in ***I*** that contain *q.*

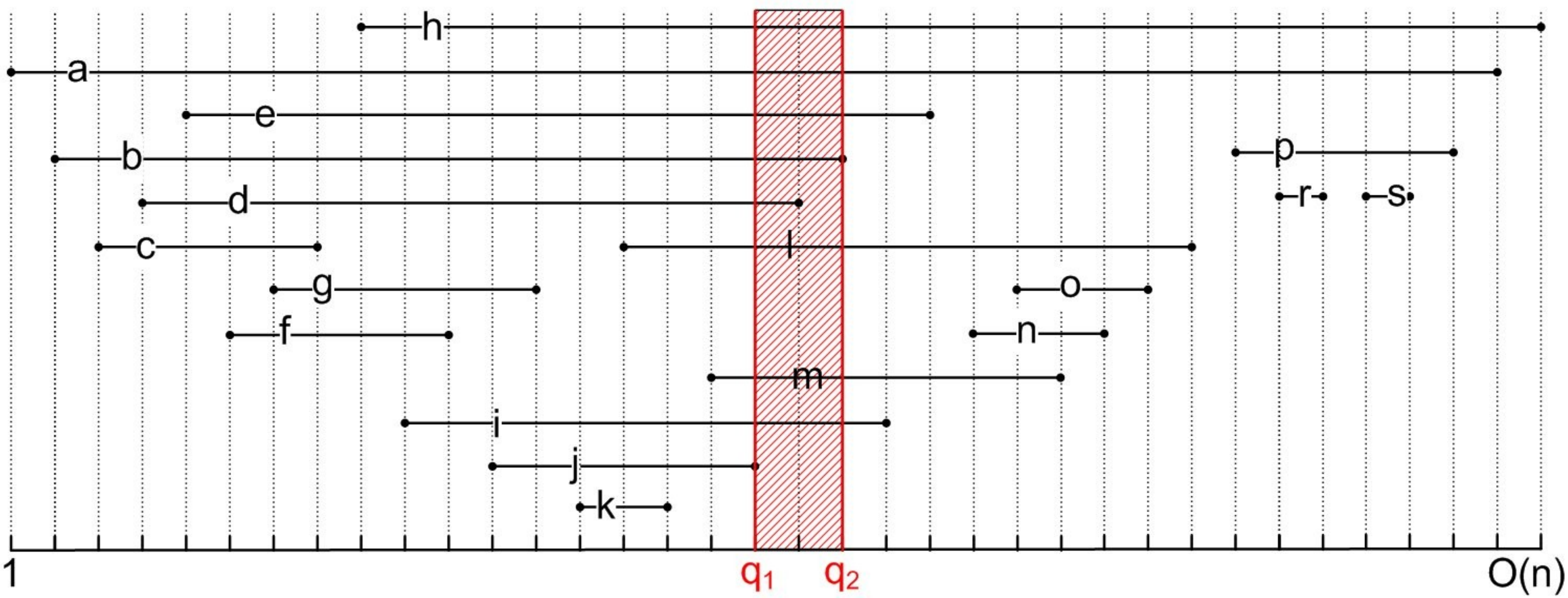
Wanted:

- Data structure that supports queries efficiently.



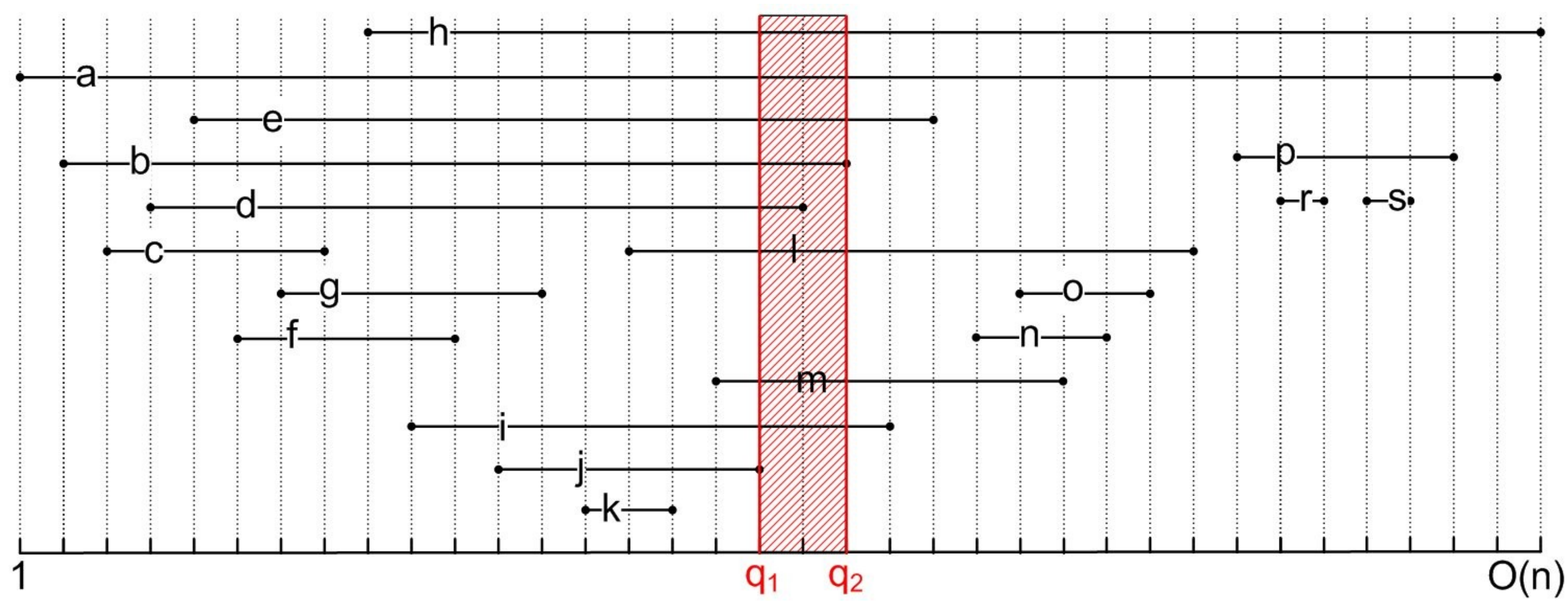
Interval Stabbing Problems

- **Interval Stabbing Problem**
- **Interval Intersection Problem:**
  - Given a query interval $[q_1,q_2]$, report all intervals in ***I*** that intersect $[q_1,q_2]$.



Interval Stabbing Problems

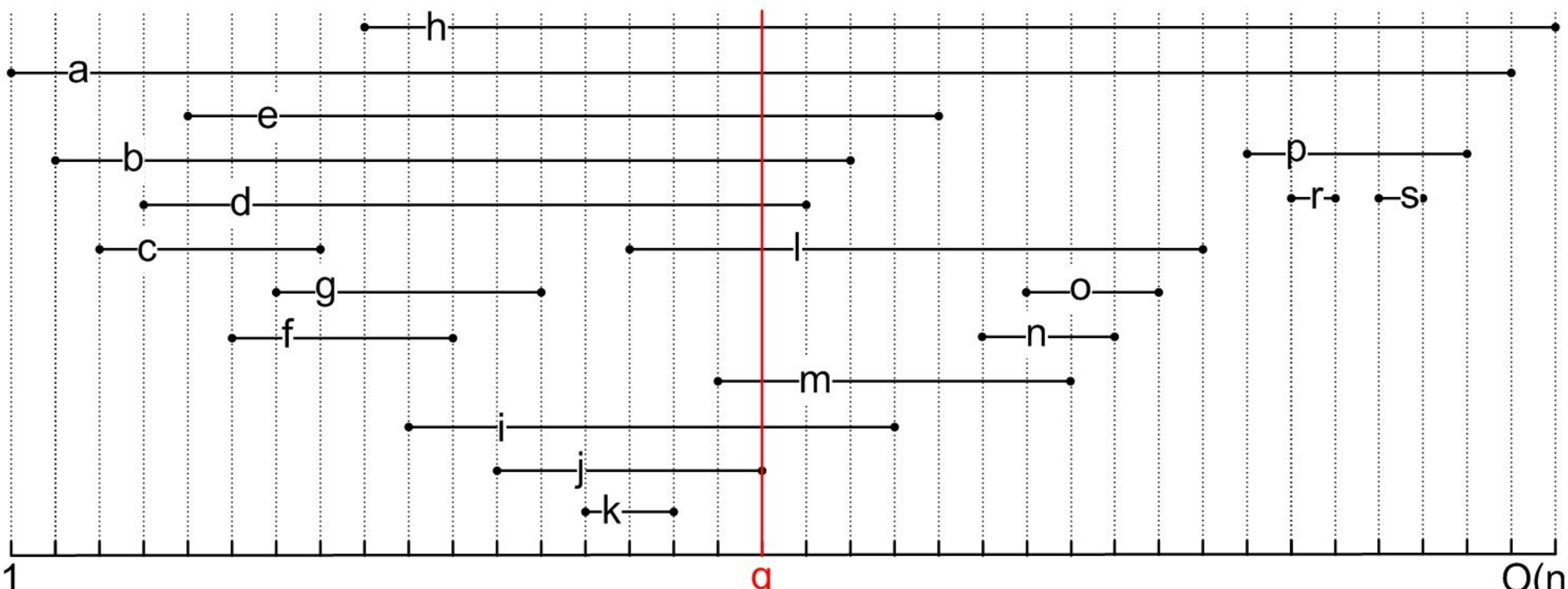
- **Interval Cover Problem:**
  - Given a query interval $[q_1,q_2]$ in I, report all intervals in I that contain $[q_1,q_2]$.
- **Multiple Query Problems:**
  - Given multiple queries sorted in lexicographic order, extend each prior problem to report intervals being contained in the union of outputs.



Interval Stabbing Problems

- Worst case running time for a query is *O(n).*
  ⇒ output-sensitive complexity

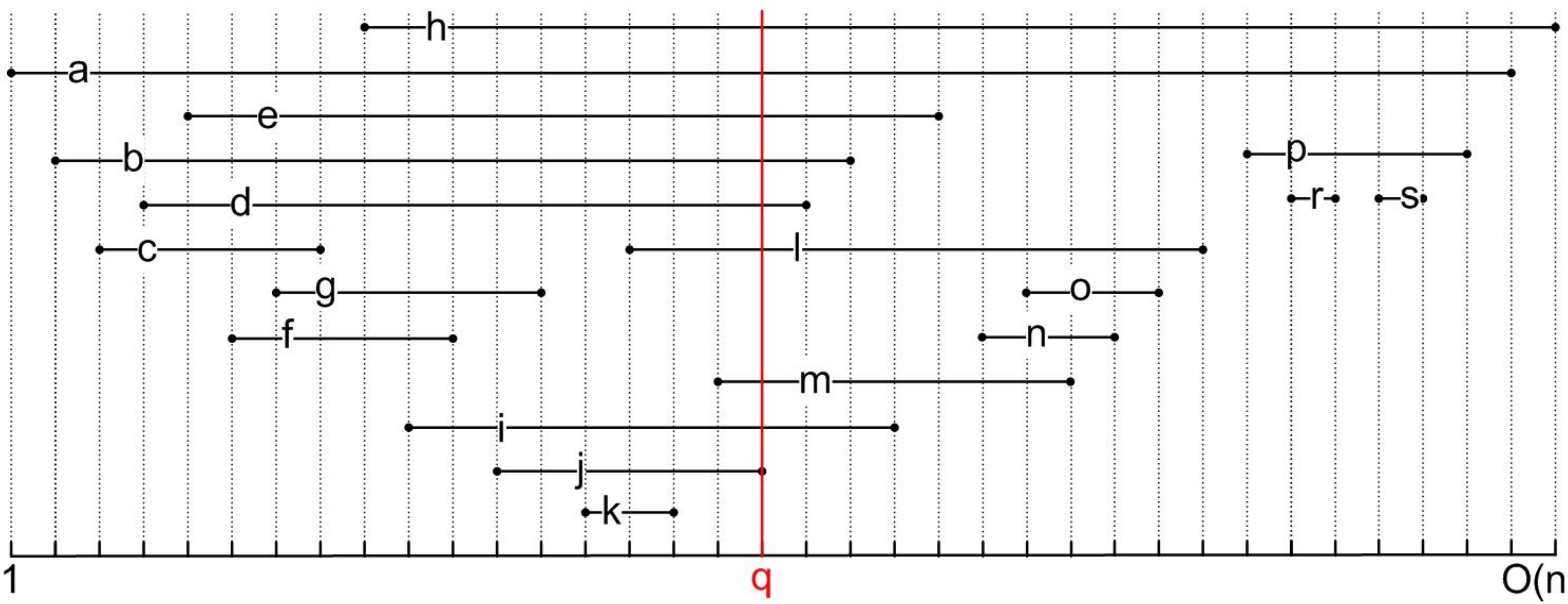
- We want optimal time *O(1+k)* for *k* intervals in the output.



Why Small Integer Ranges?

Let all interval endpoints be in *{1,...,N}*.

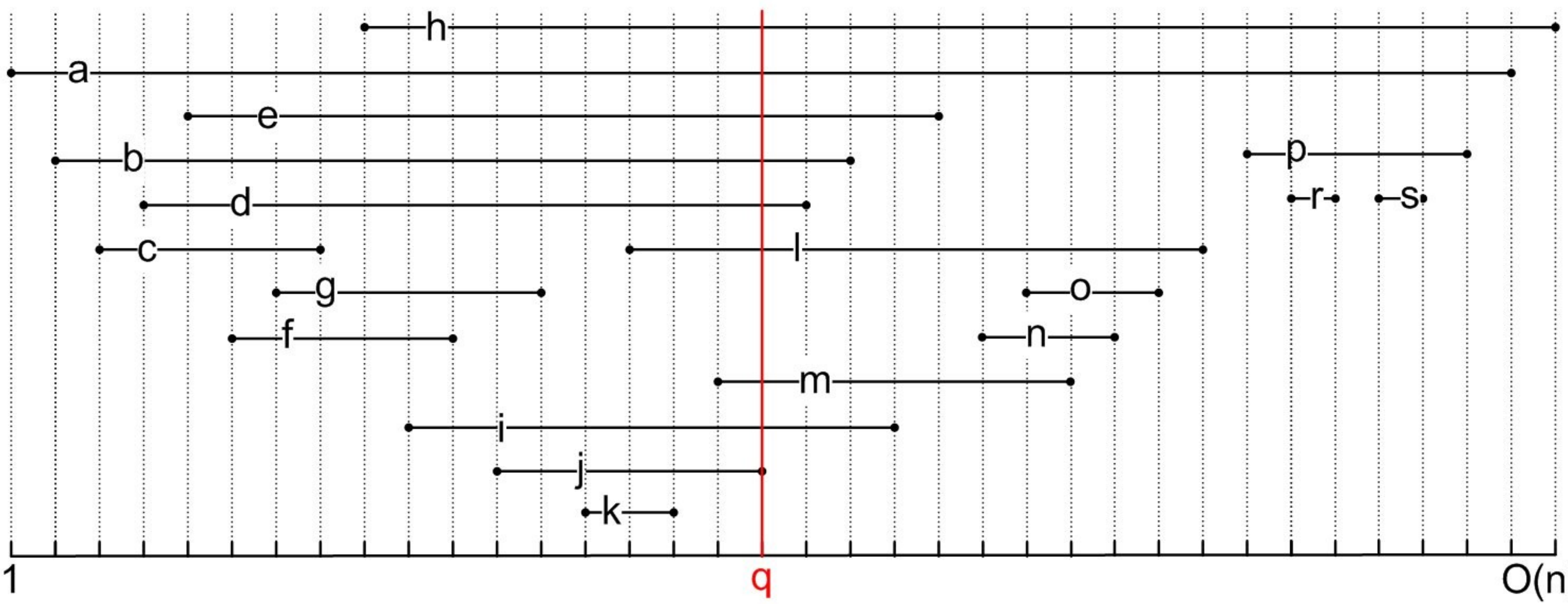
<u>Thm (Beame and Fitch 1999):</u>
For arbitrary *N*, <u>every data structure</u> using $n^{O(1)}$ memory cells needs $\Omega(\sqrt{\log(n)/\log(\log(n))})$ time for a stabbing query.



Why Small Integer Ranges?

To achieve constant time we have to impose a restriction:
⇒ We assume that all endpoints and q are in {1,...,O(n)}.

- W.l.o.g. all endpoints are pairwise distinct.



Overview

Wanted: Data structure for

- Interval Stabbing Problem
- Interval Intersection Problem
- Interval Cover Problem
- Multiple Query Problems

with

- *O(n)* preprocessing
- Stabbing queries in optimal time *O(1+k)*, output-sens.
- Output sorted by left endpoints

Overview

1986 Chazelle: Filtering Search

Wanted: Data structure for

- Interval Stabbing Problem
- Interval Intersection Problem
- Interval Cover Problem
- Multiple Query Problems

with

- *O(n)* preprocessing
- Stabbing queries in optimal time *O(1+k)*, output-sens.
- Output sorted by left endpoints

Overview

Wanted: Data structure for

- Interval Stabbing Problem
- Interval Intersection Problem
- Interval Cover Problem
- Multiple Query Problems

1986 Chazelle: Filtering Search

2000 Alstrup et al.: 3-sided range queries, O(1+k), but involved

with

- *O(n)* preprocessing
- Stabbing queries in optimal time *O(1+k)*, output-sens.
- Output sorted by left endpoints

Overview

Wanted: Data structure for

- Interval Stabbing Problem
- Interval Intersection Problem
- Interval Cover Problem
- Multiple Query Problems

1986 Chazelle: Filtering Search

2000 Alstrup et al.: 3-sided range queries, O(1+k), but involved

with

- *O(n)* preprocessing
- Stabbing queries in optimal time *O(1+k)*, output-sens.
- Output sorted by left endpoints

We will focus on the first problem.

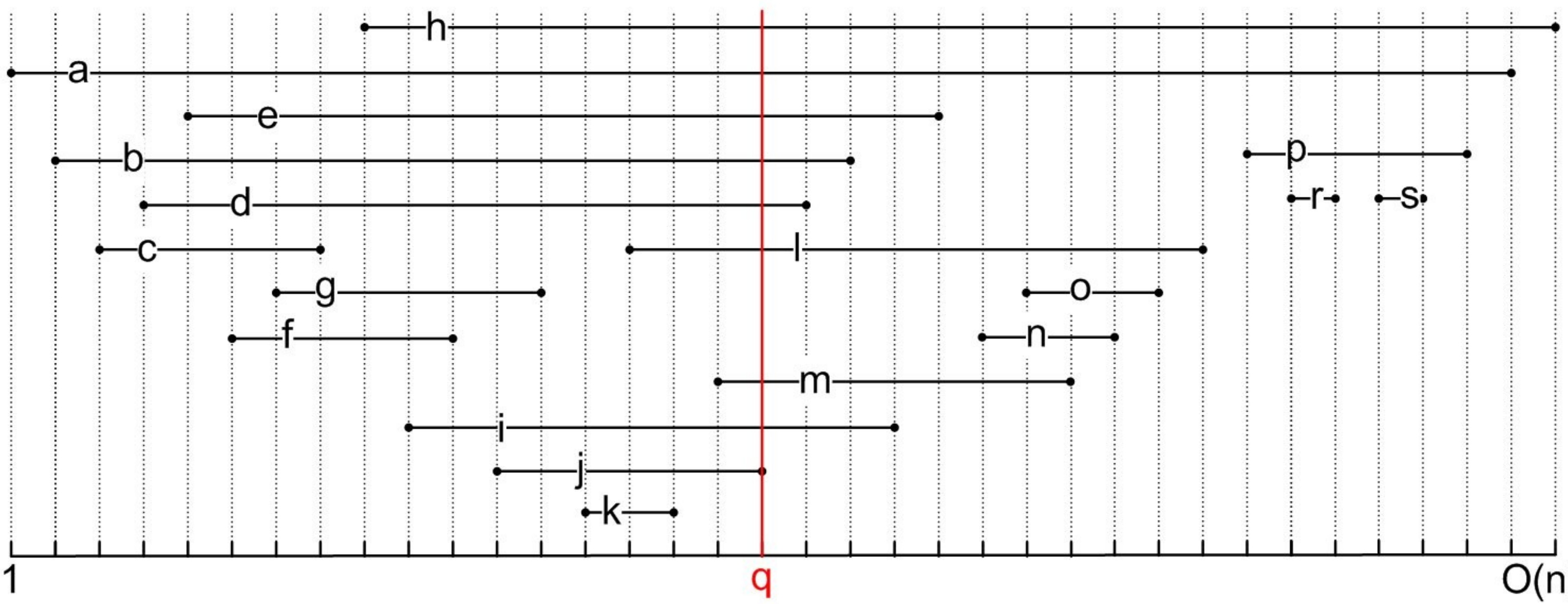
Outline

1. Problem Definitions

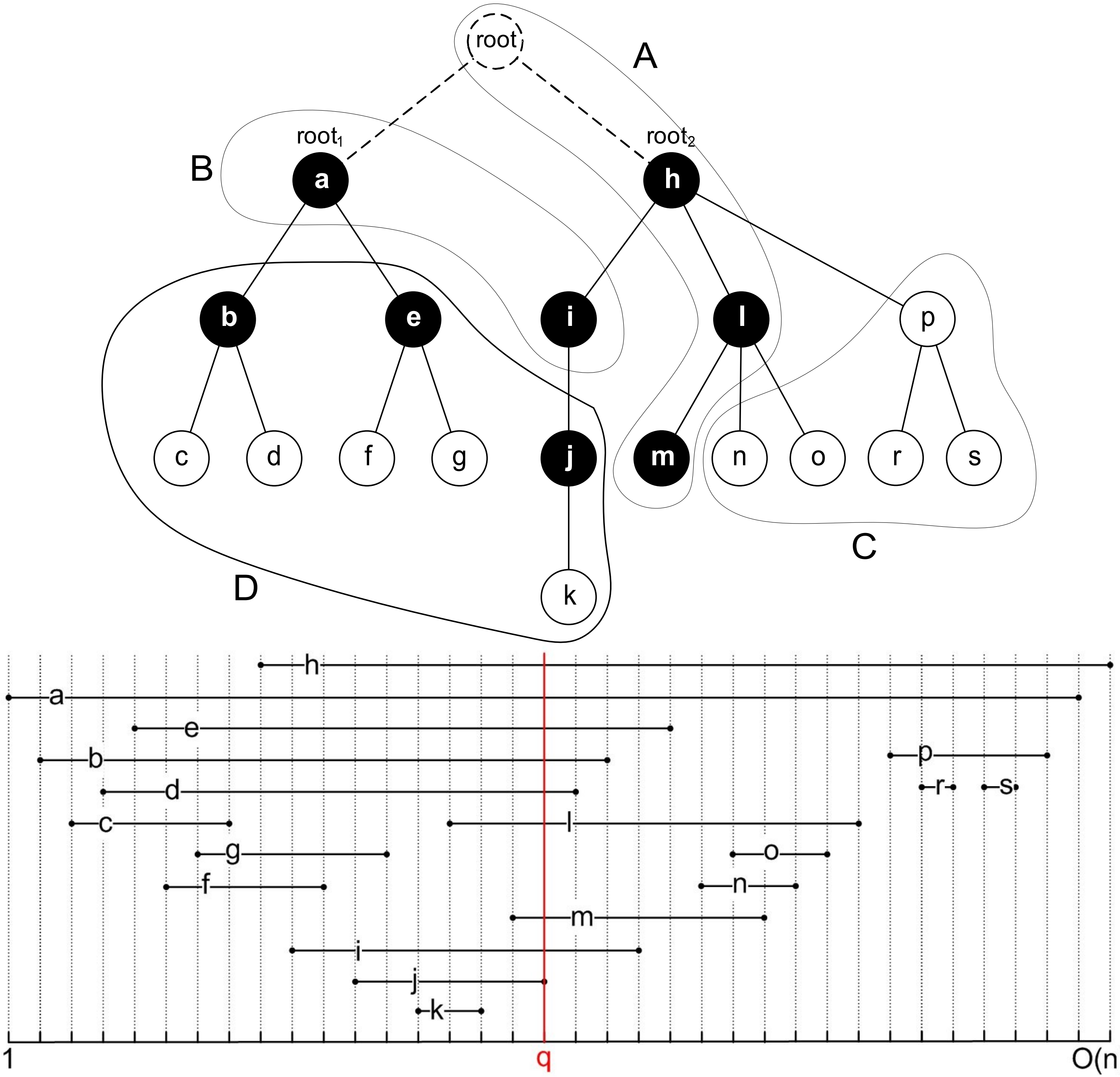
2. Data Structure

Data Structure

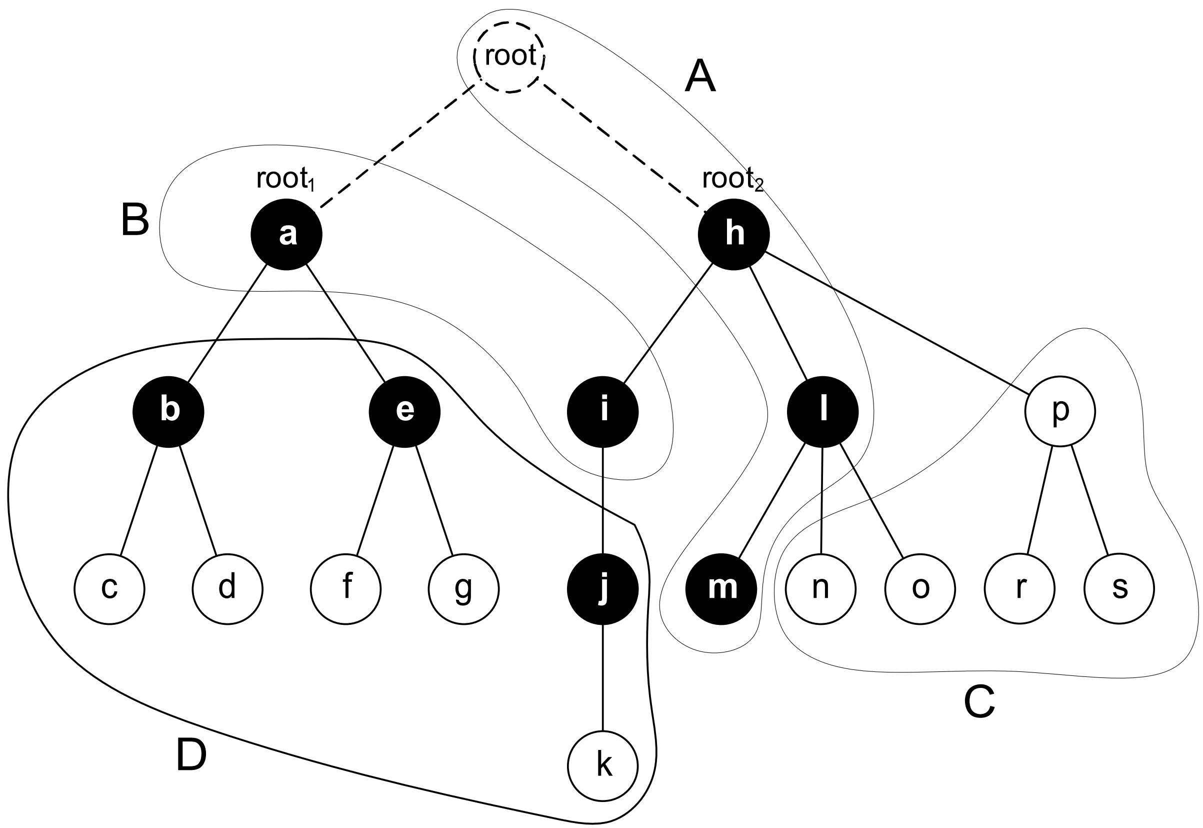
- An interval in a subset of *I* is *rightmost* if it is the one with maximum left endpoint.
- For an interval i:
  Parent(i) := rightmost interval that contains i



Data Structure



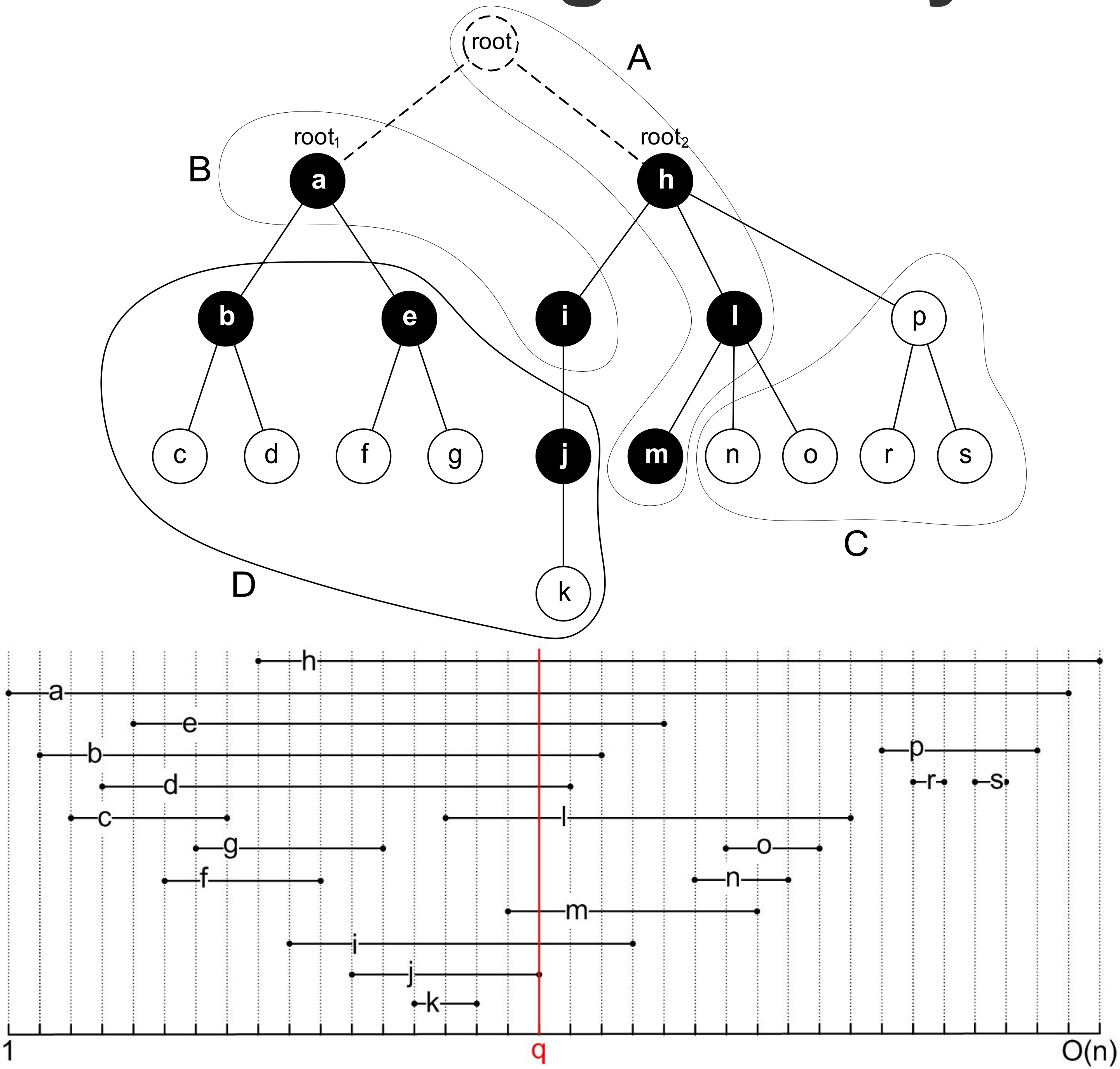
Data Structure



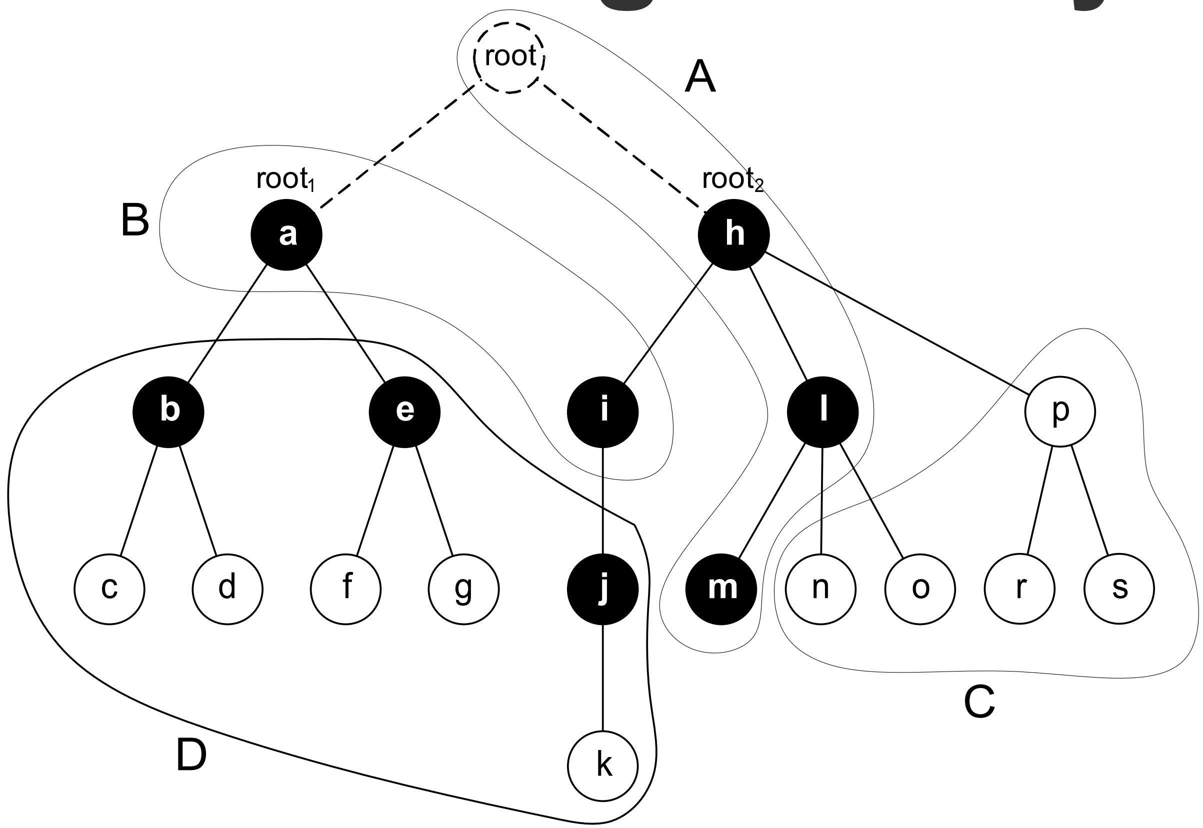
- All Parents can be computed in *O(n)* by a sweep line alg.
- Parents build a forest
- Data Structure: The forest + virtual root (trees ordered)

- We handle a query on *q* by traversing the forest from the (precomputed) rightmost interval containing *q*.

Processing a Query

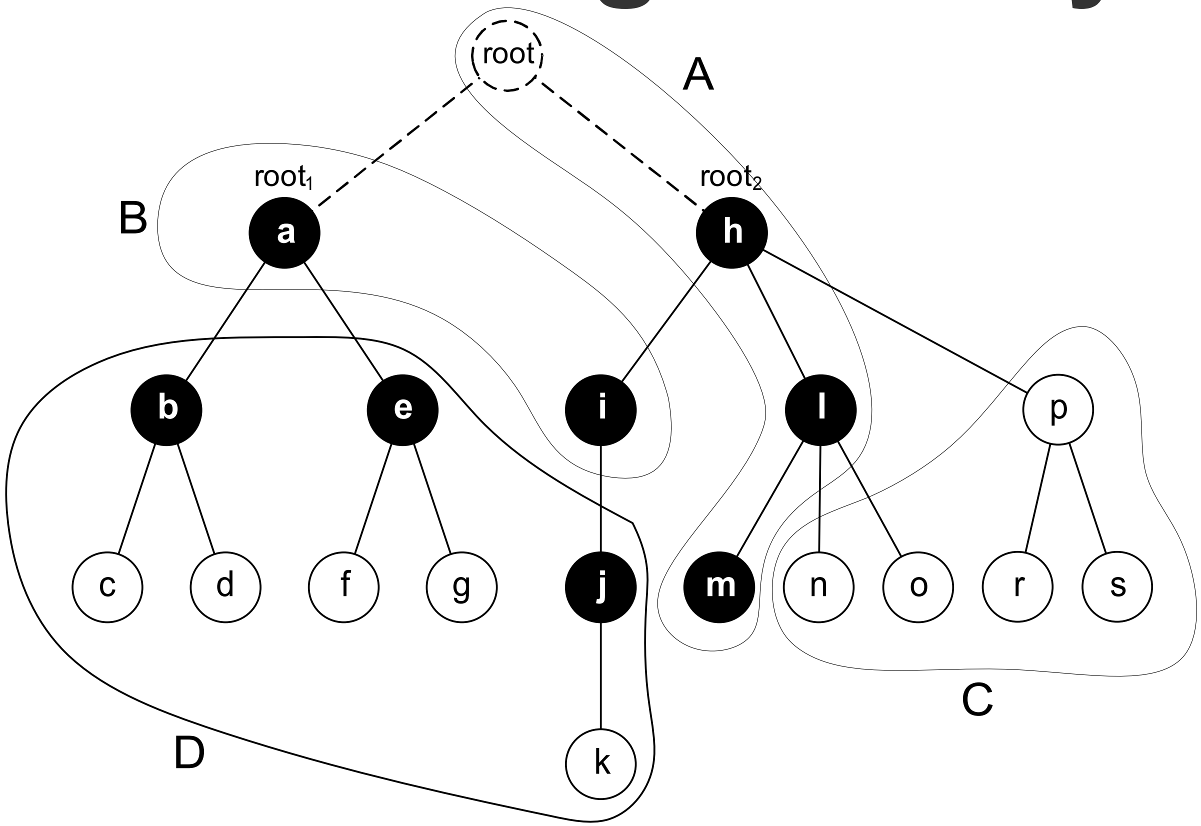


Processing a Query



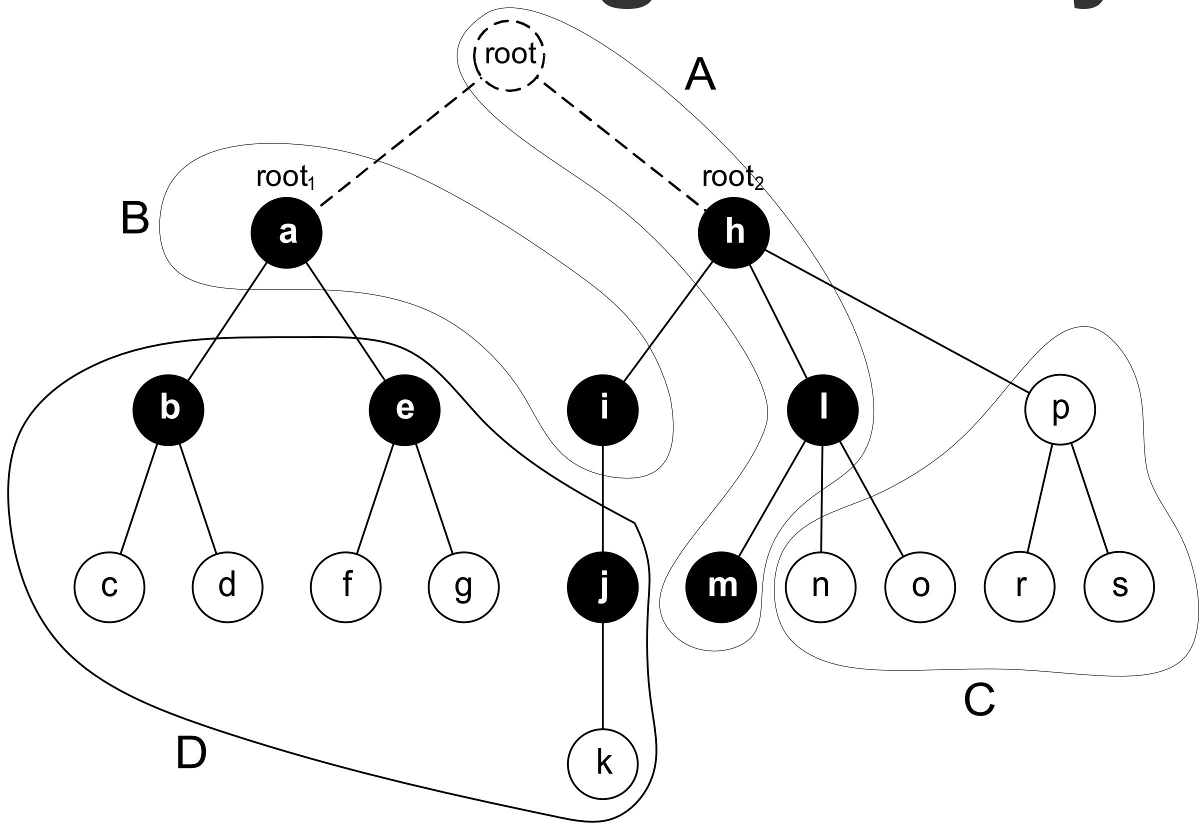
- All intervals in A are stabbed.
- No interval in C is stabbed.
- Stabbed siblings are adjacent.
  ⇒ Stabbed intervals in B can be computed efficiently.
- Only D remains.

Processing a Query



- Lemma 1: Every stabbed vertex $v \in D$ has a (stabbed) ancestor in *B*.

Processing a Query



- Lemma 2: The sibling *w* to the right of a stabbed vertex *v*∈*D* is stabbed as well, if it exists.

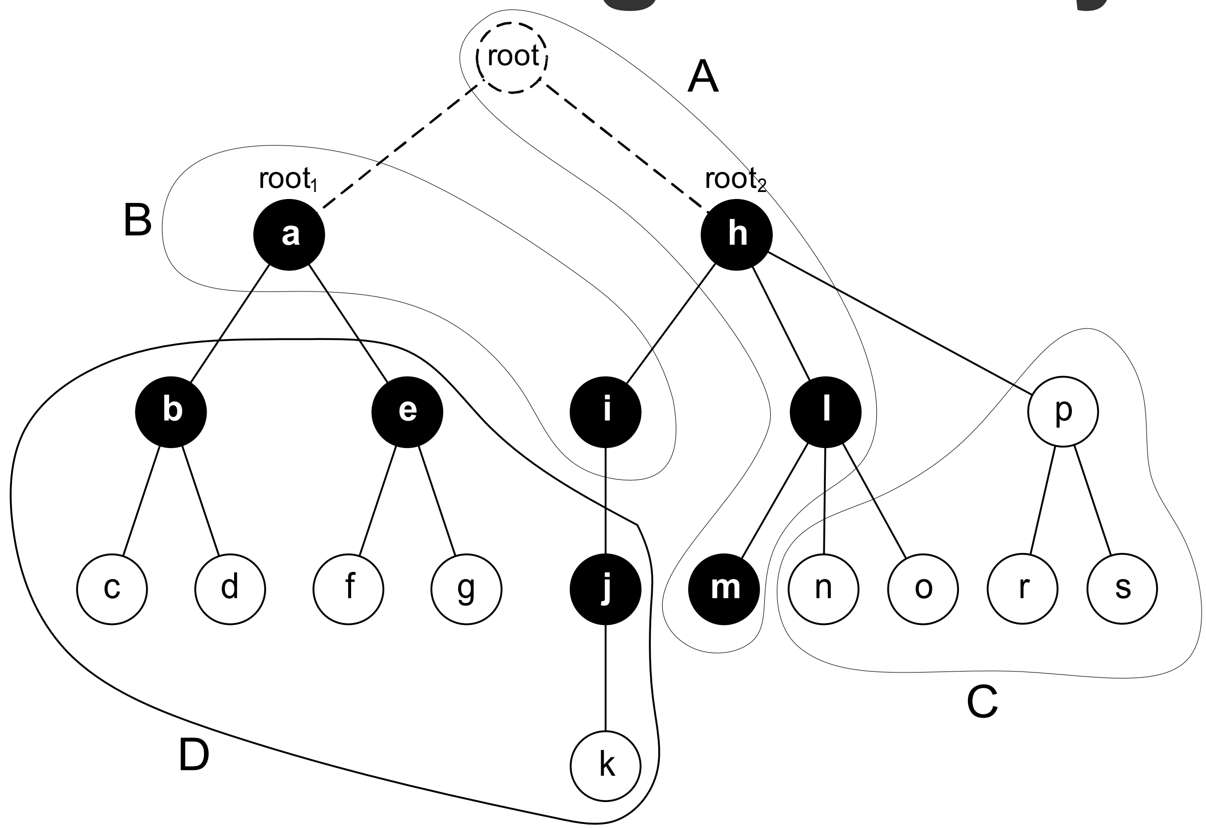
z'∈B∪A stabbed

z∈B stabbed

w∈D

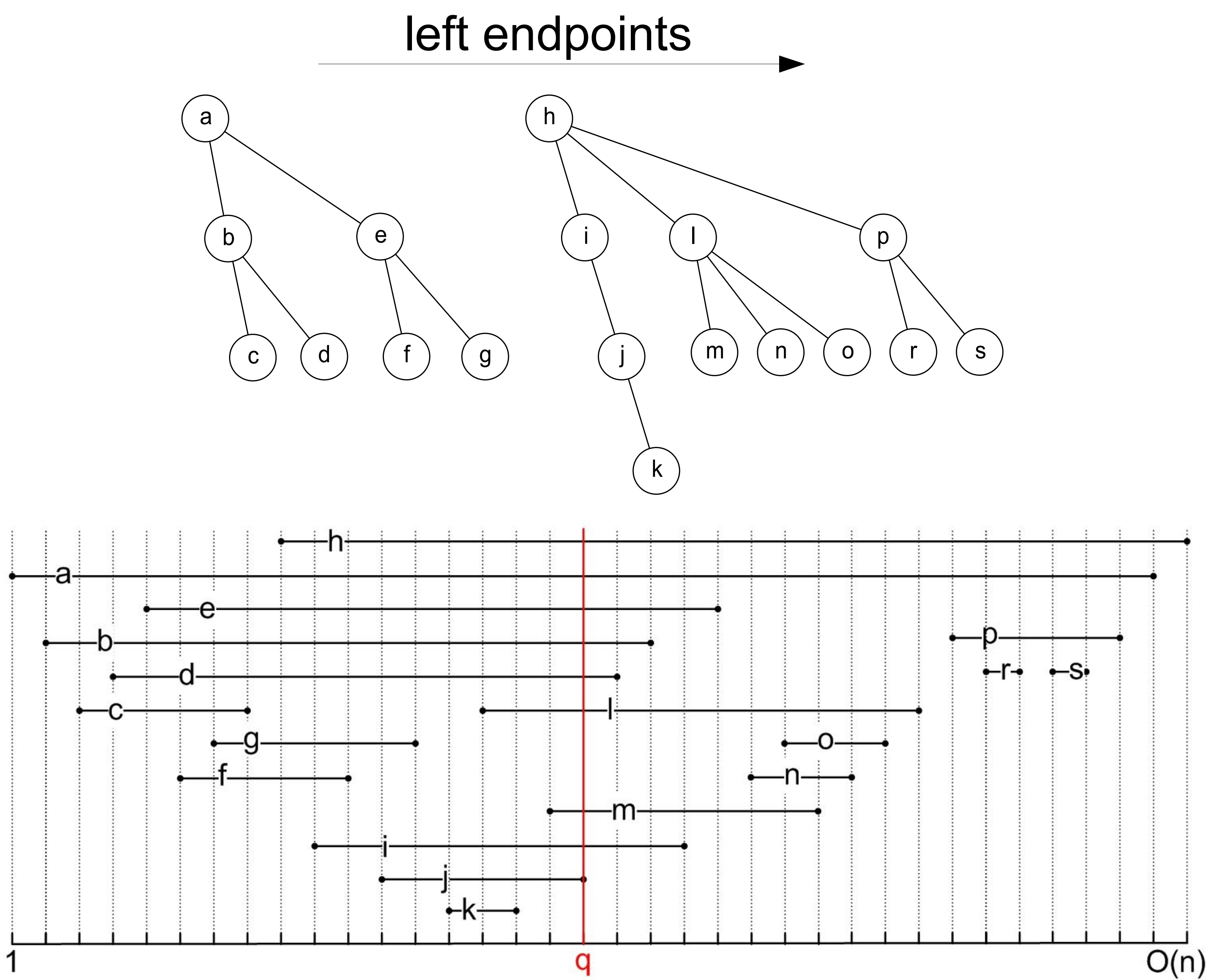
v∈D stabbed

Processing a Query



- It follows that every stabbed vertex $v \in D$ can be reached from a stabbed vertex in B by a zig-zag-path consisting of stabbed vertices.
- Only 3 directions to check on being stabbed:
  to the rightmost child, to the left, and up (only in A).

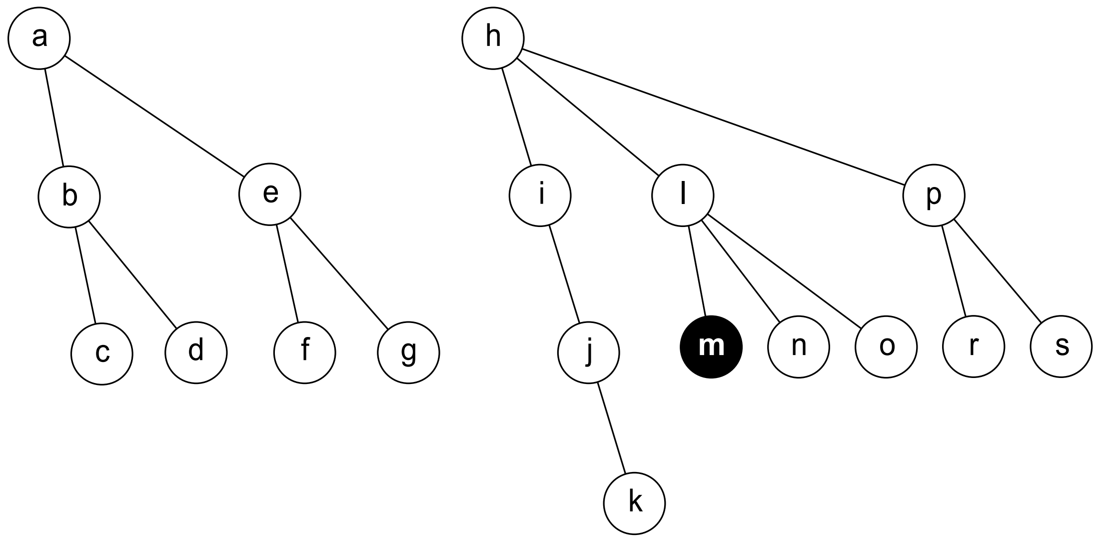
Example

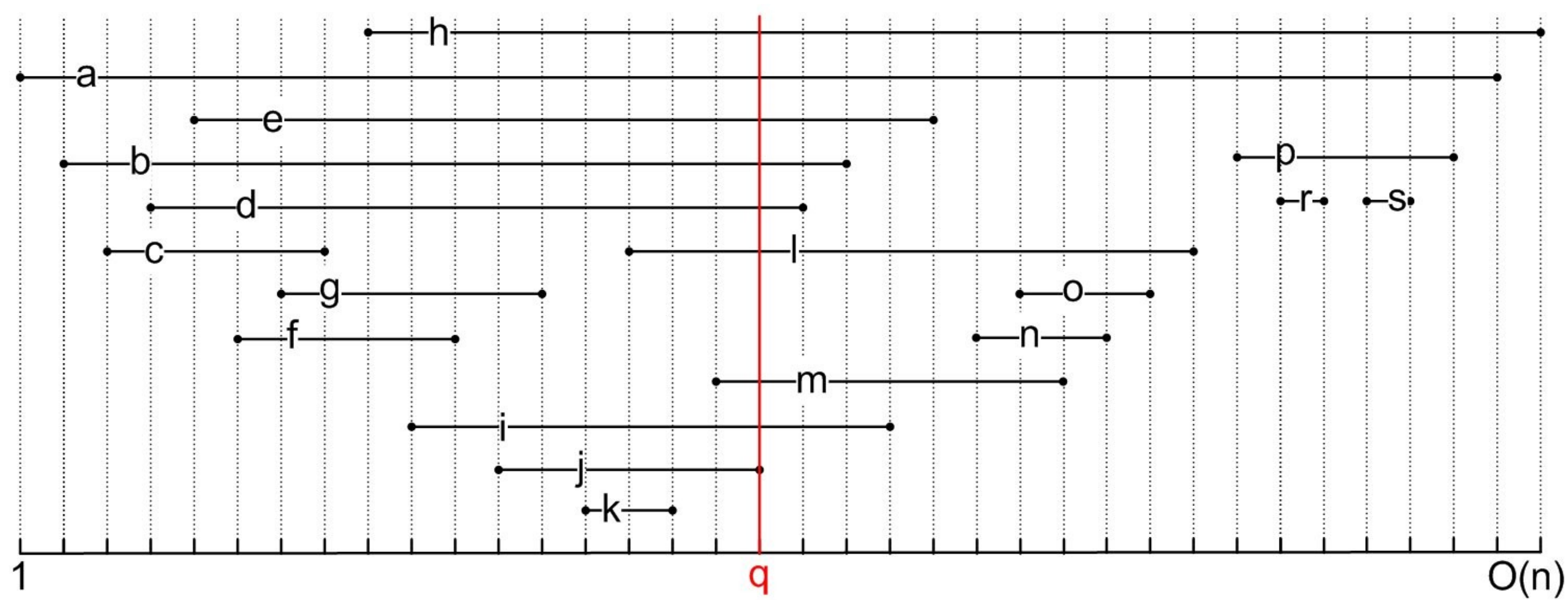


left endpoints
a
b
e
c
d
f
g
h
i
l
p
j
m
n
o
r
s
k
h
a
e
b
p
d
r
s
c
l
g
o
f
n
m
i
j
k
1
q
O(n)

Example

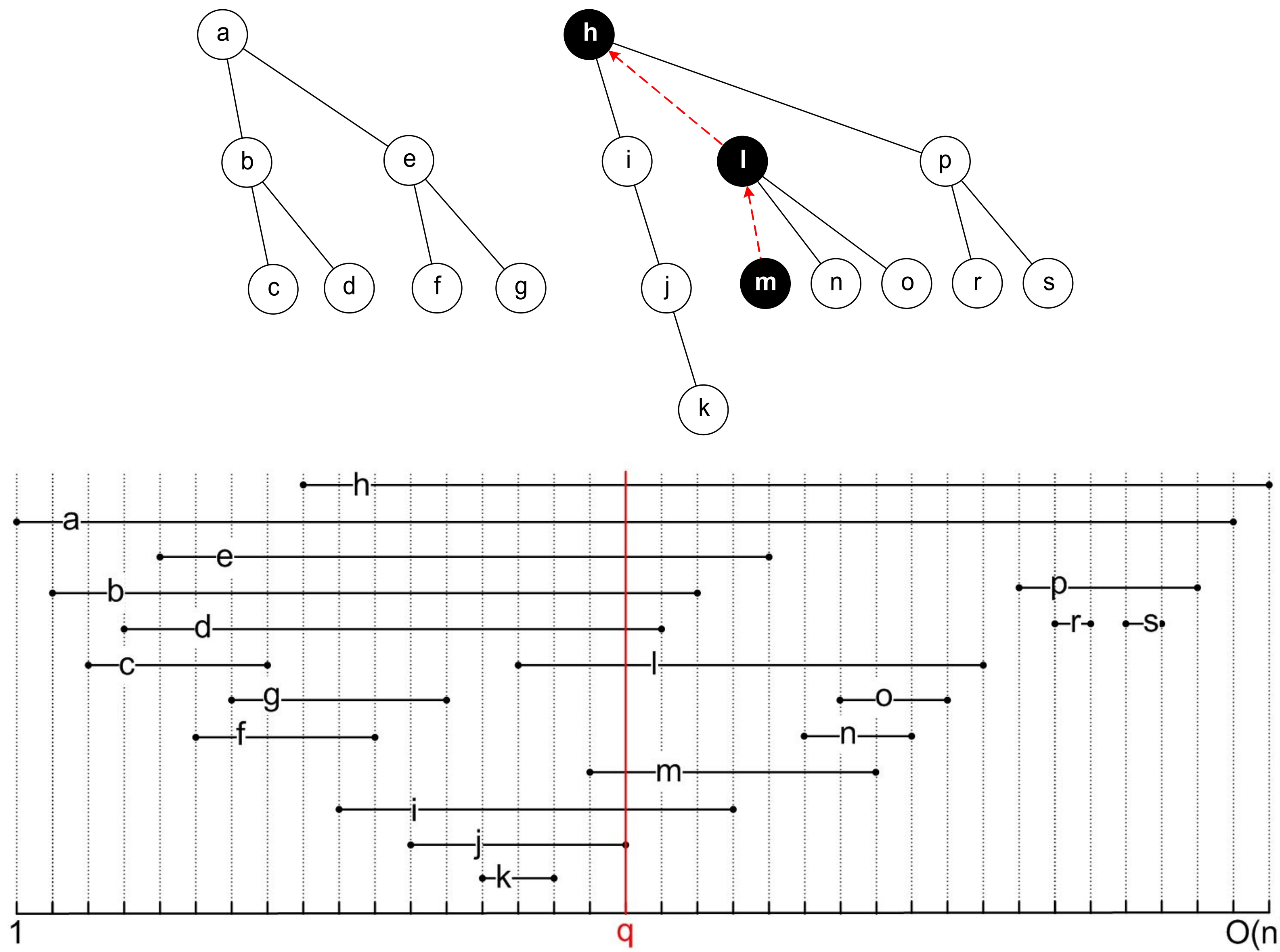
start traversal at the rightmost interval containing *q*





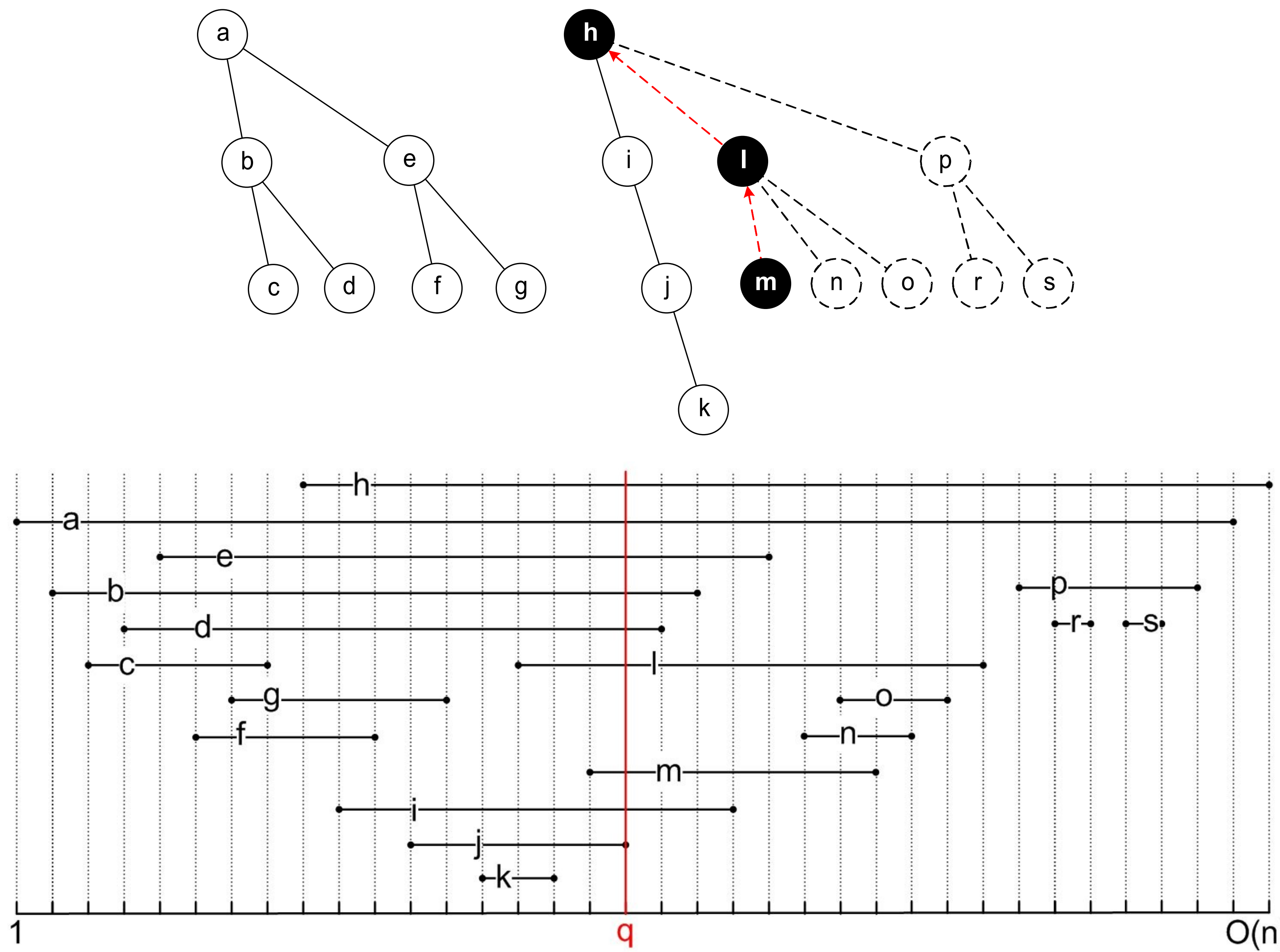
Example

ancestors of stabbed intervals are stabbed



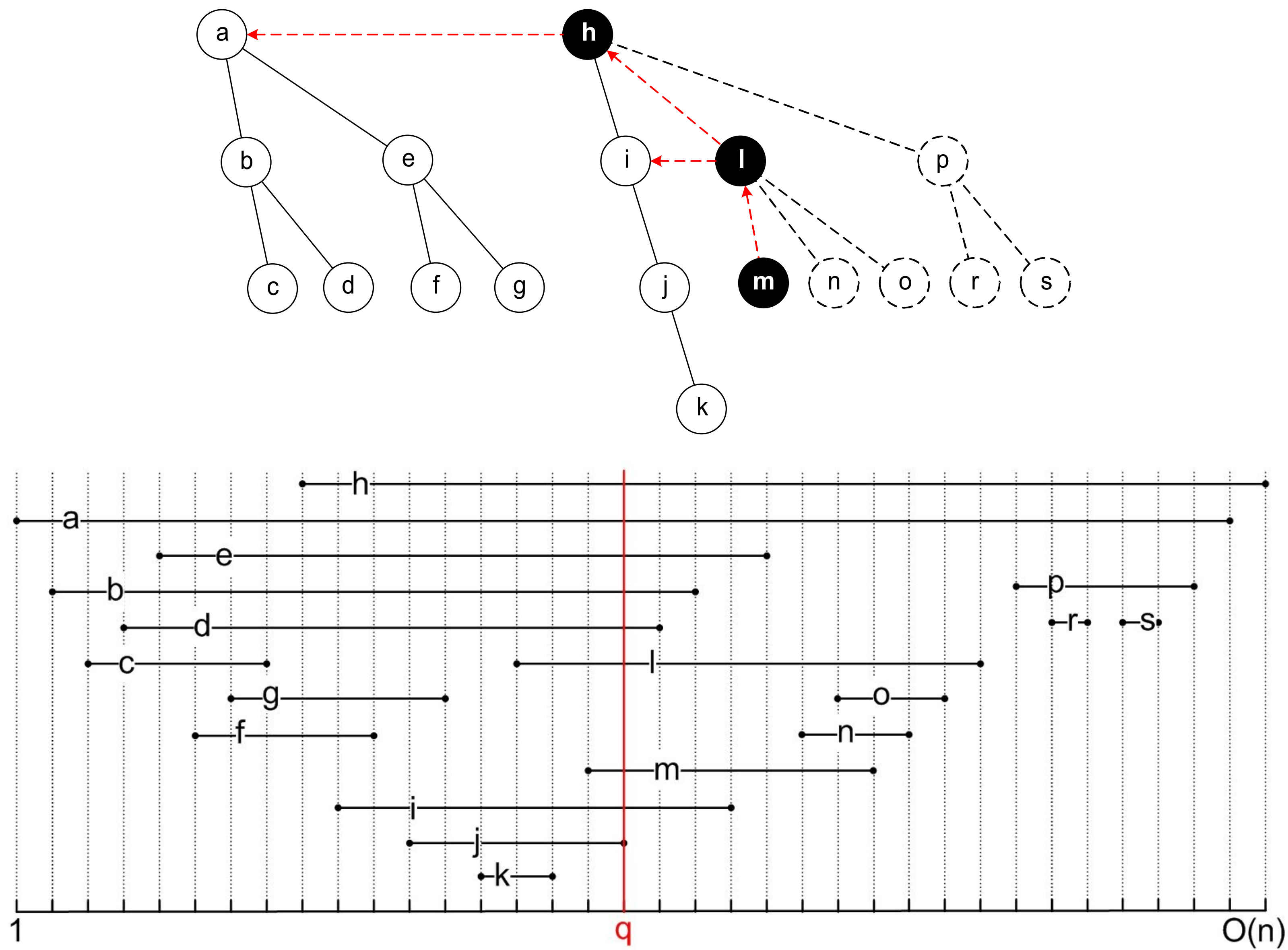
Example

C does not contain stabbed intervals



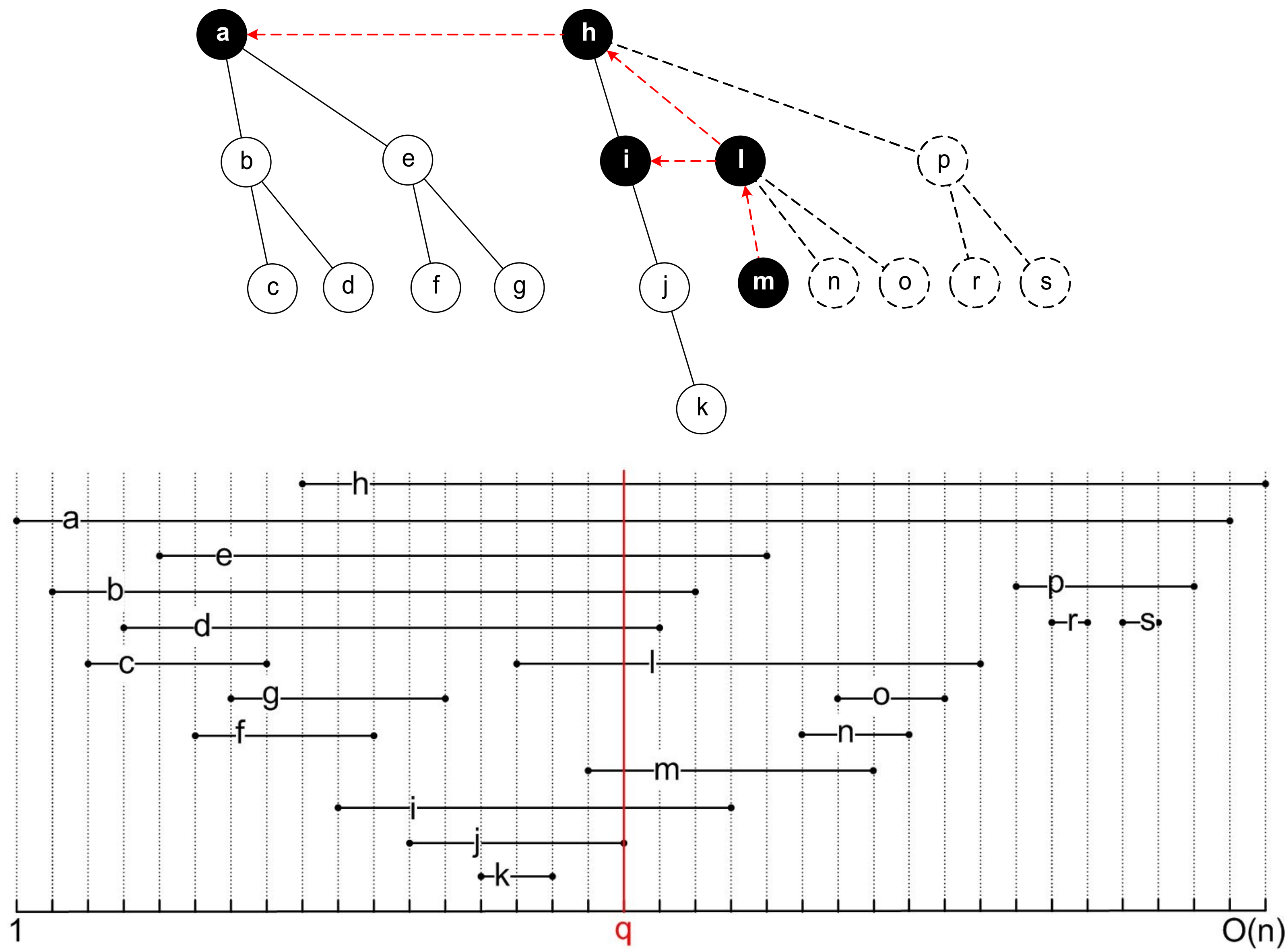
Example

check left siblings successively in *O(1)*...



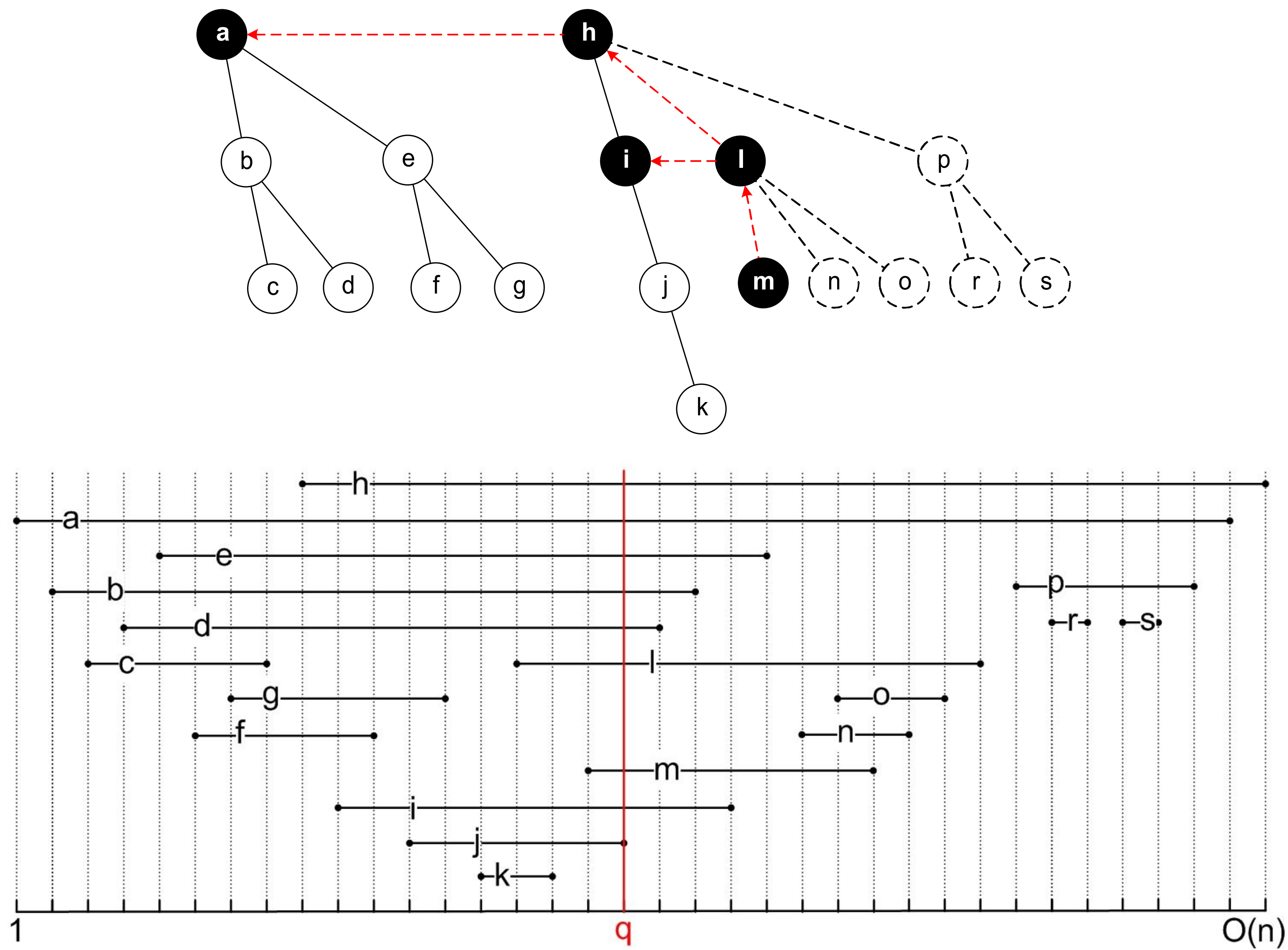
Example

...and get *B*



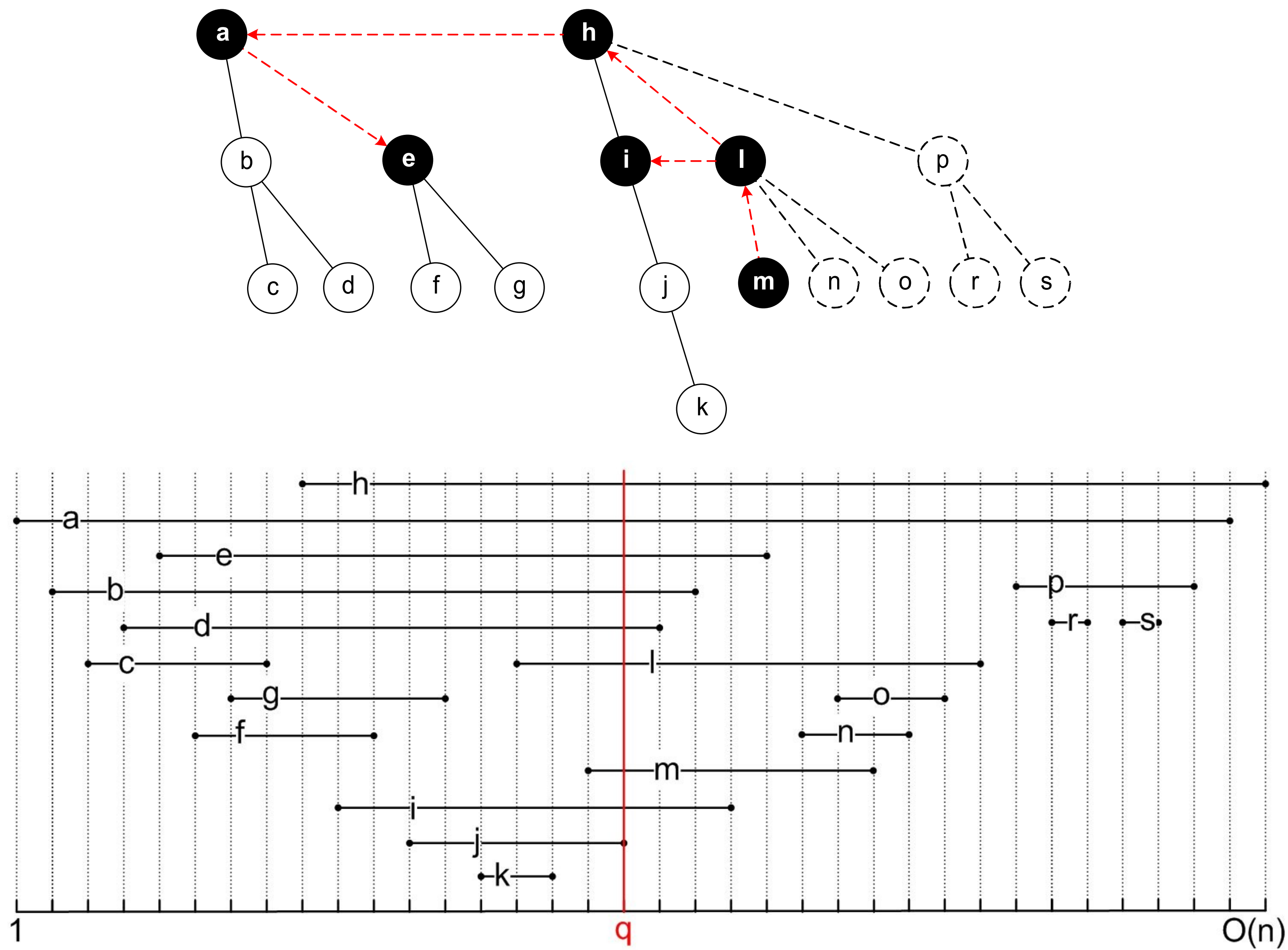
Example

try first to go to the rightmost child, then to go to the left



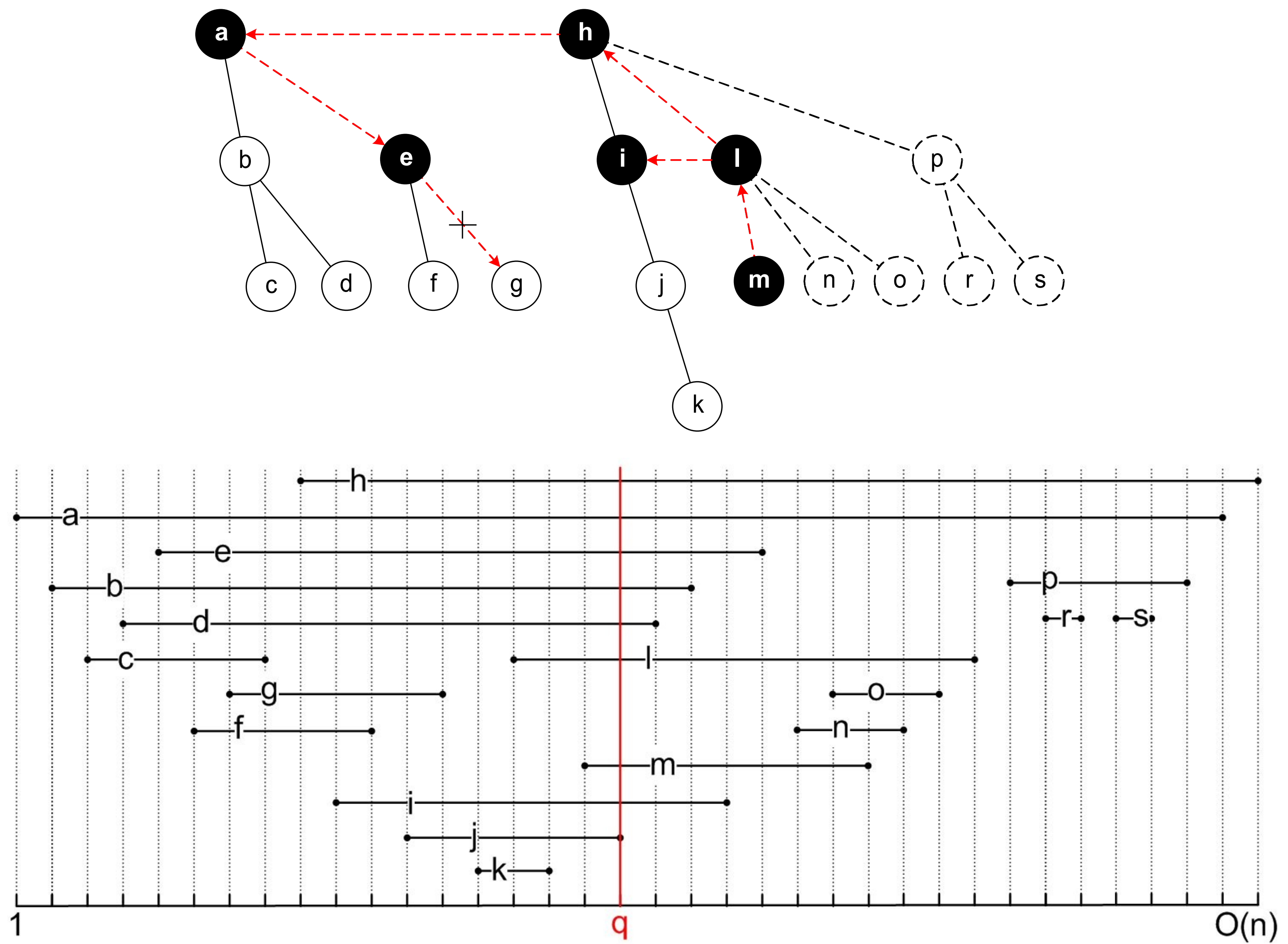
Example

try first to go to the rightmost child, then to go to the left



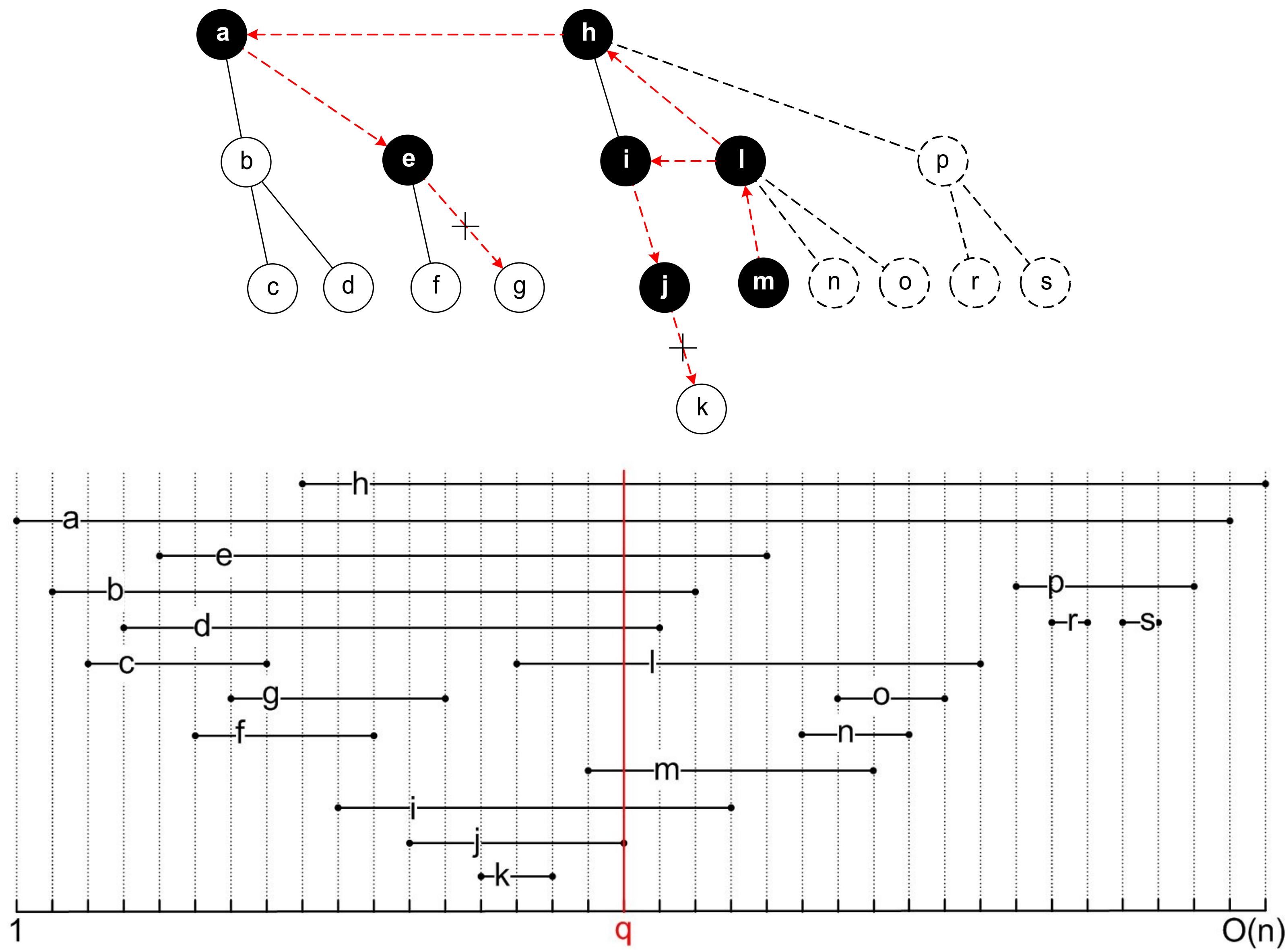
Example

try first to go to the rightmost child, then to go to the left



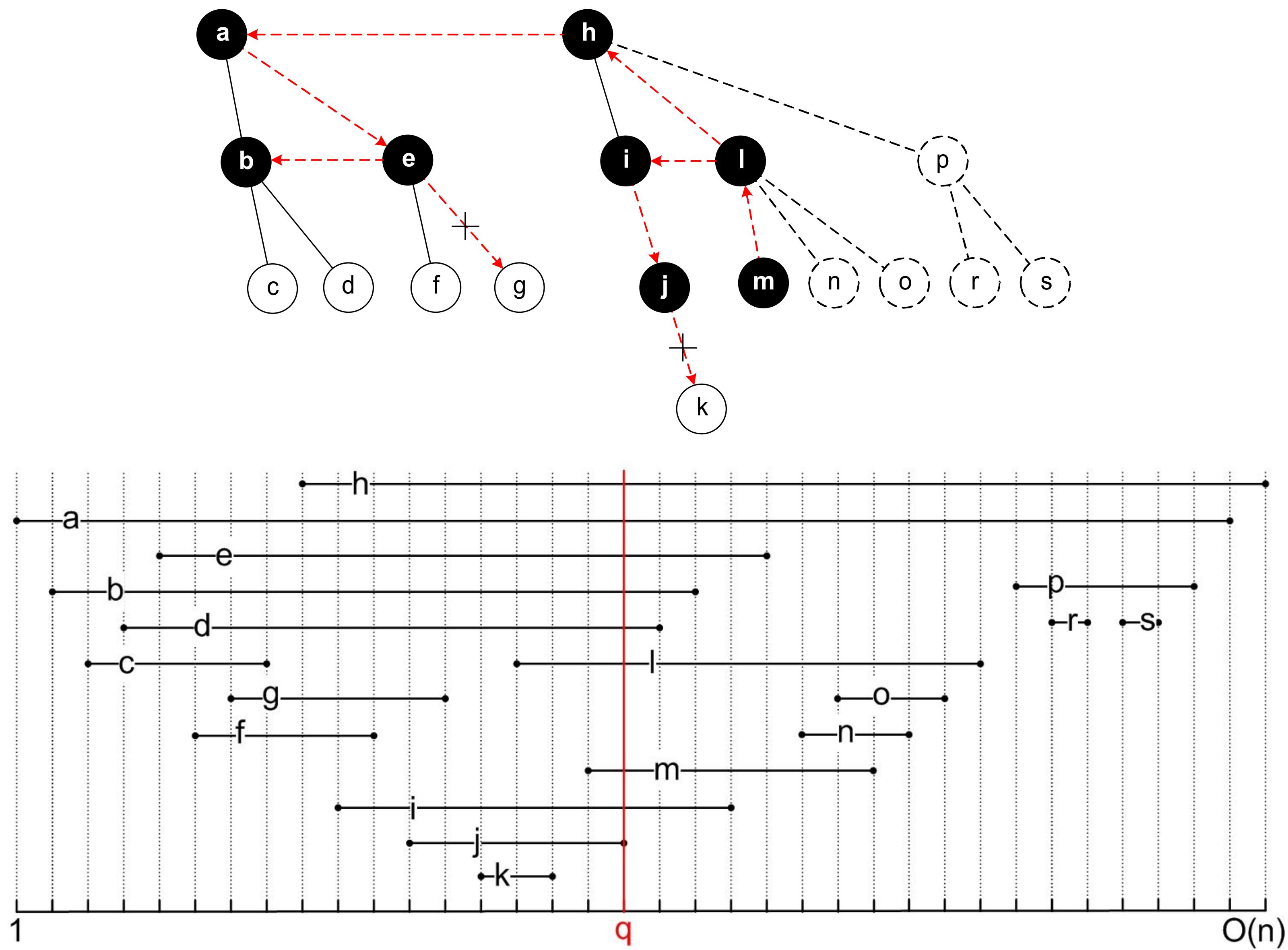
Example

try first to go to the rightmost child, then to go to the left



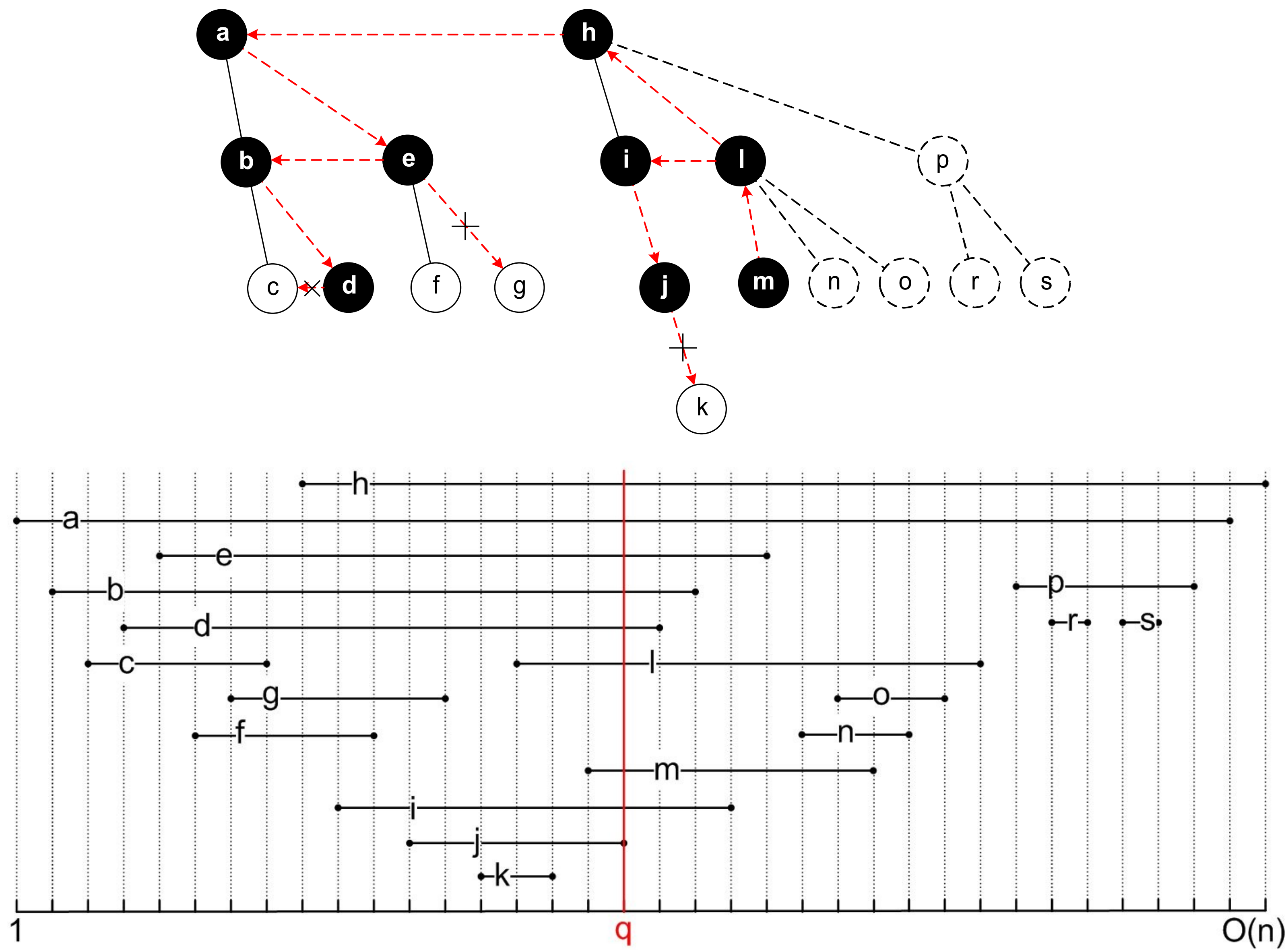
Example

try first to go to the rightmost child, then to go to the left



Example

try first to go to the rightmost child, then to go to the left



Problem Variants

- **Interval Intersection Problem:**
  - Perform query on the right endpoint of query interval, but change the stopping condition of the transversal.
- **Interval Cover Problem:**
  - Lemmas 1 and 2 still hold when *q* is an interval.
- **Multiple Query Problems:**
  - Start with the rightmost query and choose adaptively the next lower query value.