SORTING WITH PREDICTIONS

XINGJIAN BAI, CHRISTIAN COESTER

UNIVERSITY OF OXFORD

Motivation

Traditional Algorithms

Worst-case guarantees

Pessimistic?

Machine Learning Models

Often very powerful

No guarantee

Real life $\neq$ worst case, often predictable

(e.g., solve similar instances repeatedly)

Algorithms with predictions

Goal: Good predictions $\Longrightarrow$ much better performance

Bad predictions $\Longrightarrow$ same worst-case guarantee

Sorting with Predictions

Task: Sort an array of items, $a_1, a_2, \ldots, a_n$, wrt. <

Positional Prediction Setting:

Receive prediction of the ranking of each item

Dirty Comparison Setting:

Access to quick-and-dirty comparisons between each pair of items, besides slow-and-clean comparisons.

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

$p(i)$: true ranking of a_i in the sorted list

$\hat{p}(i)$: predicted ranking of a_i in the sorted list

Error: $\eta_i = |\hat{p}(i) - p(i)|$

Displacement Sort: $O\left(\sum_{i=1}^{n} \log(\eta_i + 2)\right)$

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

true ranking of a_i in the sorted list

prediction $\hat{p}(i)$ of a_i's ranking in the sorted list

Error: $\eta_i = |\hat{p}(i) - p(i)|$, equals to the absolute difference between

$$\eta_i^l := \left| \{j \in [n] : \hat{p}(j) \le \hat{p}(i) \wedge p(j) > p(i)\} \right| \text{ and}$$

$$\eta_i^r := \left| \{j \in [n] : \hat{p}(j) \ge \hat{p}(i) \wedge p(j) < p(i)\} \right|$$

Double-Hoover Sort: $O\left(\sum_{i=1}^{n} \log\left(\min\{\eta_i^l, \eta_i^r\} + 2\right)\right)$

Sorting with Dirty Comparisons

Input: $a_1, a_2, \ldots, a_n$

slow-and-clean comparator $<$

quick-and-dirty comparator $\hat{<}$

Error: $\eta_i := \#\{j : (a_j < a_i) \neq (a_j \mathrel{\hat{<}} a_i)\}$

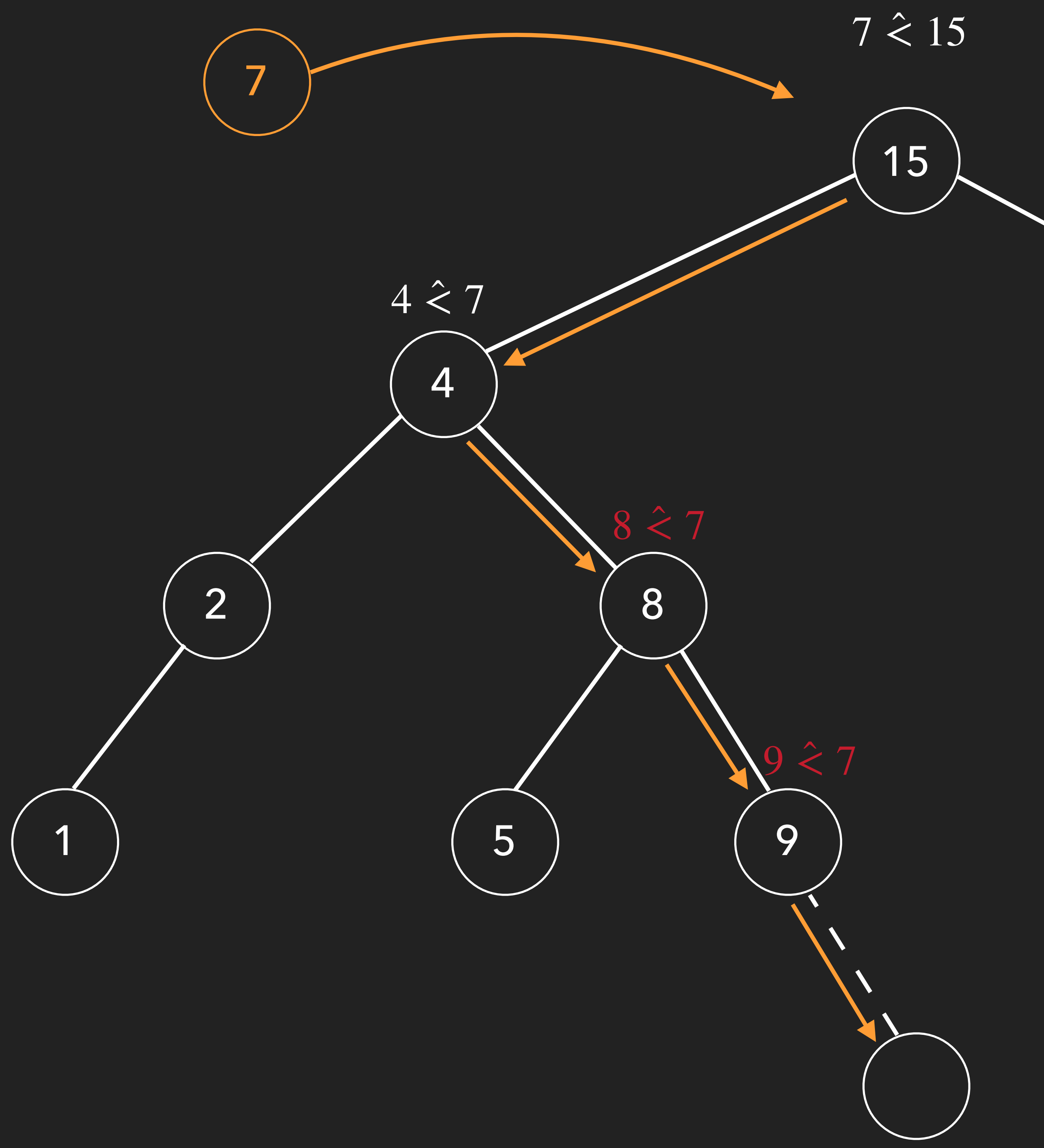
Dirty-Clean Sort: $O(n \log n)$ dirty comparisons

and $O\left(\sum_{i=1}^{n} \log(\eta_i + 2)\right)$ clean comparisons

Dirty-Clean Sort Algorithm

Idea: Build BST wrt. <

Guide insertions via $\hat{<}$ and <

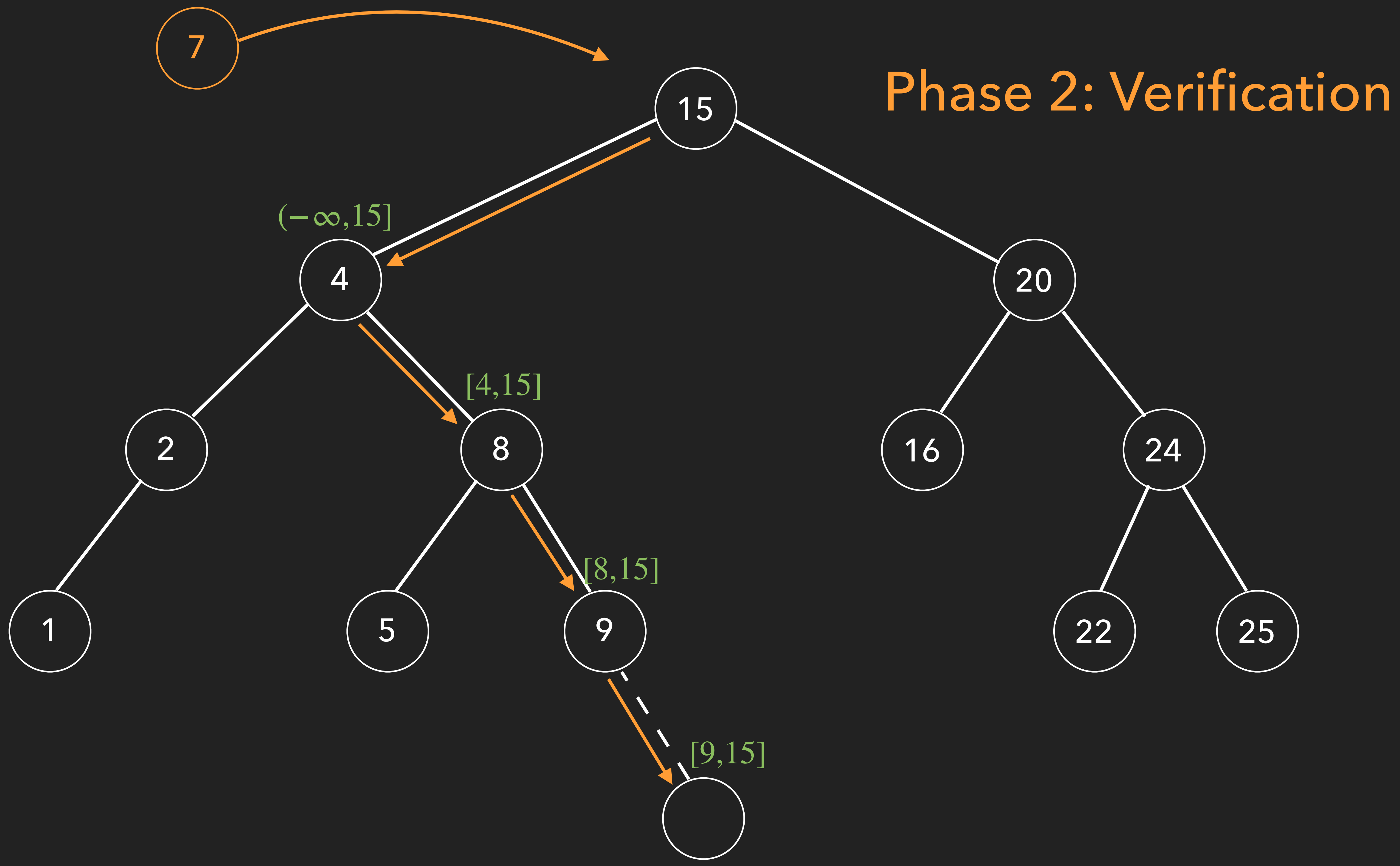


Phase 1: Dirty Probing

Dirty-Clean Sort Algorithm

Idea: Build BST wrt. <

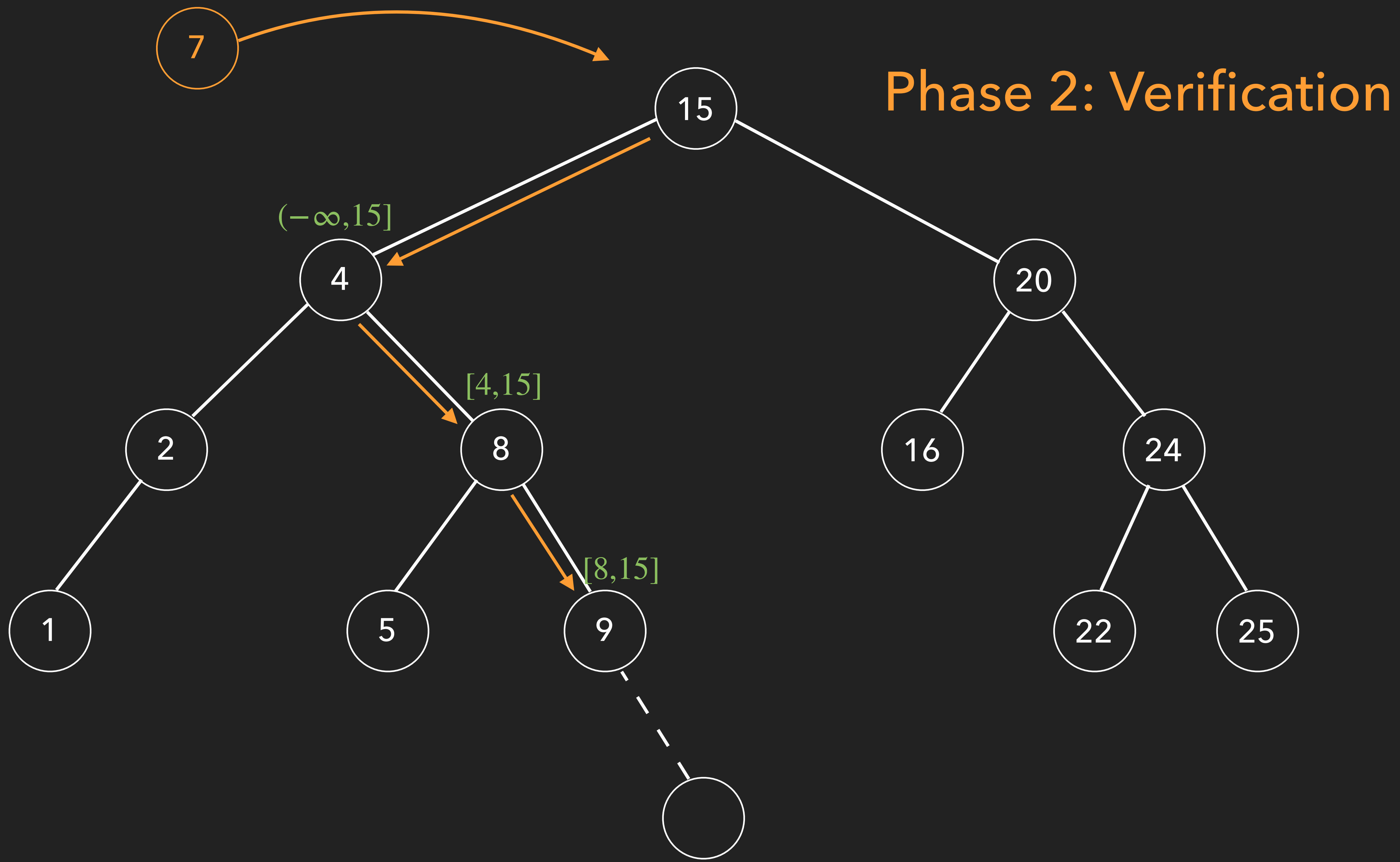
Guide insertions via $\hat{<}$ and <



Dirty-Clean Sort Algorithm

Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$

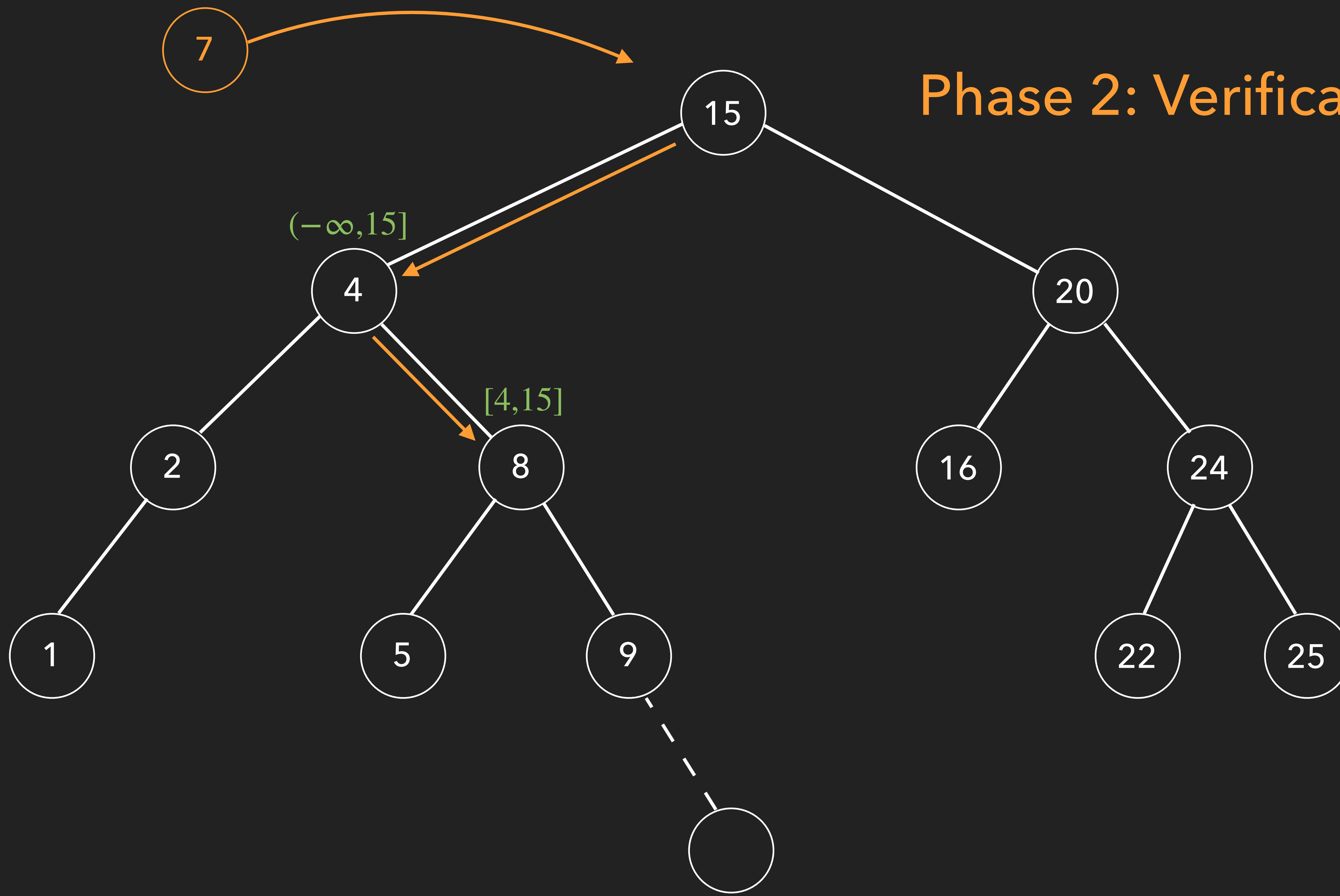


Dirty-Clean Sort Algorithm

Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$

Phase 2: Verification

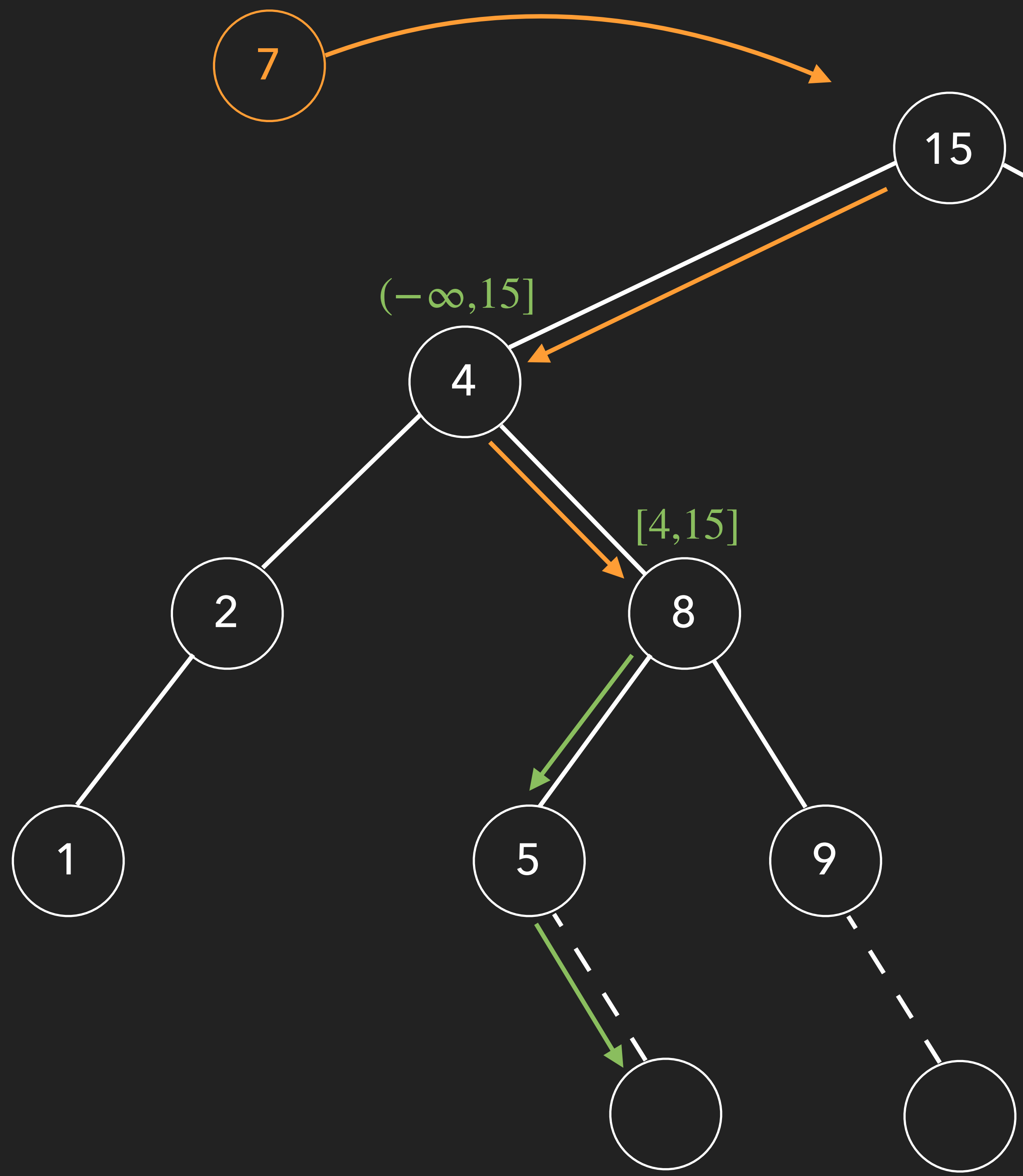


Dirty-Clean Sort Algorithm

Idea: Build BST wrt. <

Guide insertions via $\hat{<}$ and <

Phase 3: Clean Insert

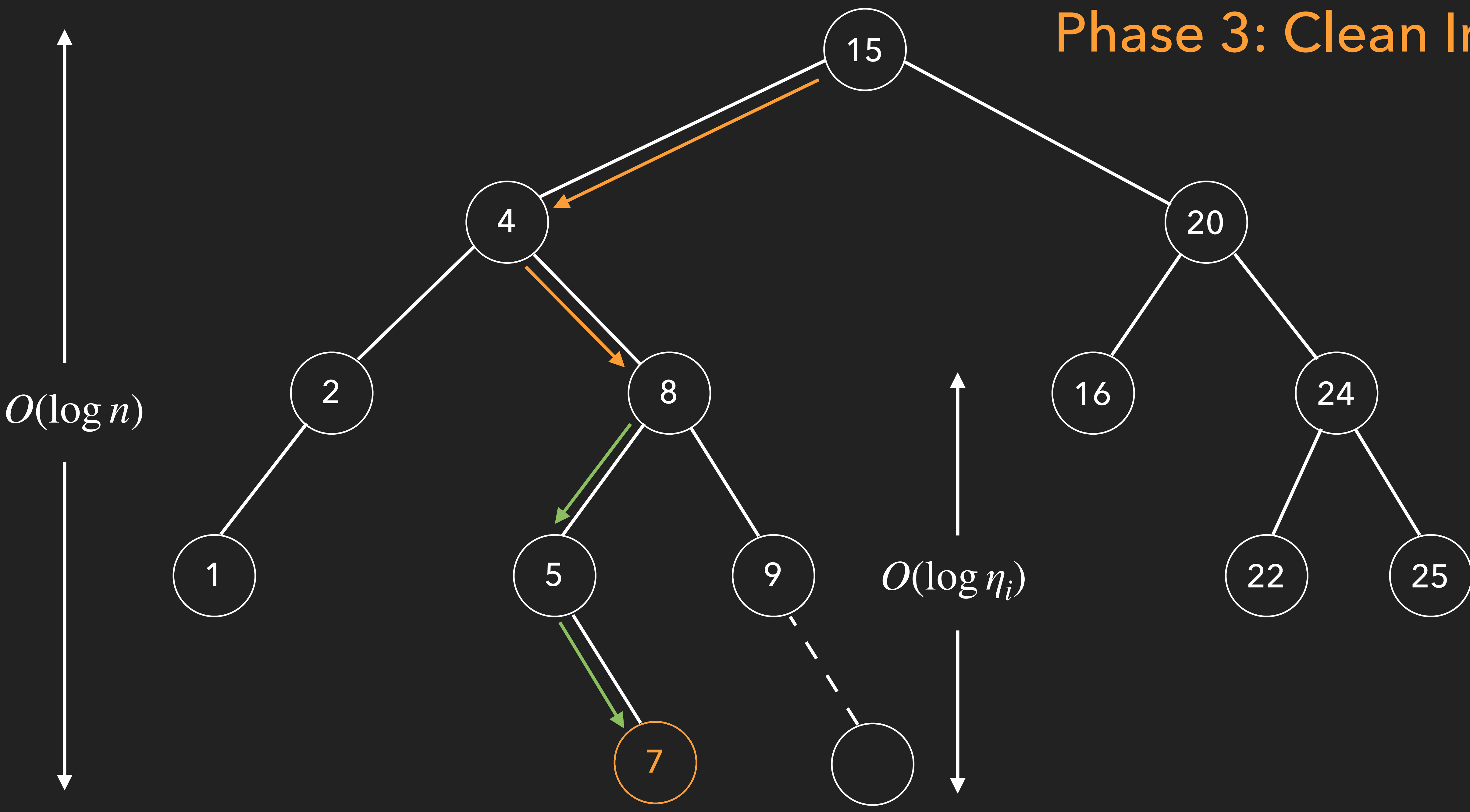


Dirty-Clean Sort Algorithm

Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$

Phase 3: Clean Insert



Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

a_i	69	28	82	67	49	71	64	38	9	81

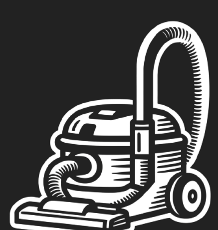
Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

In round i, each Hoover sucks in items that costs i comparisons to be inserted.

Round 1



a_i	69	28	82	67	49	71	64	38	9	81

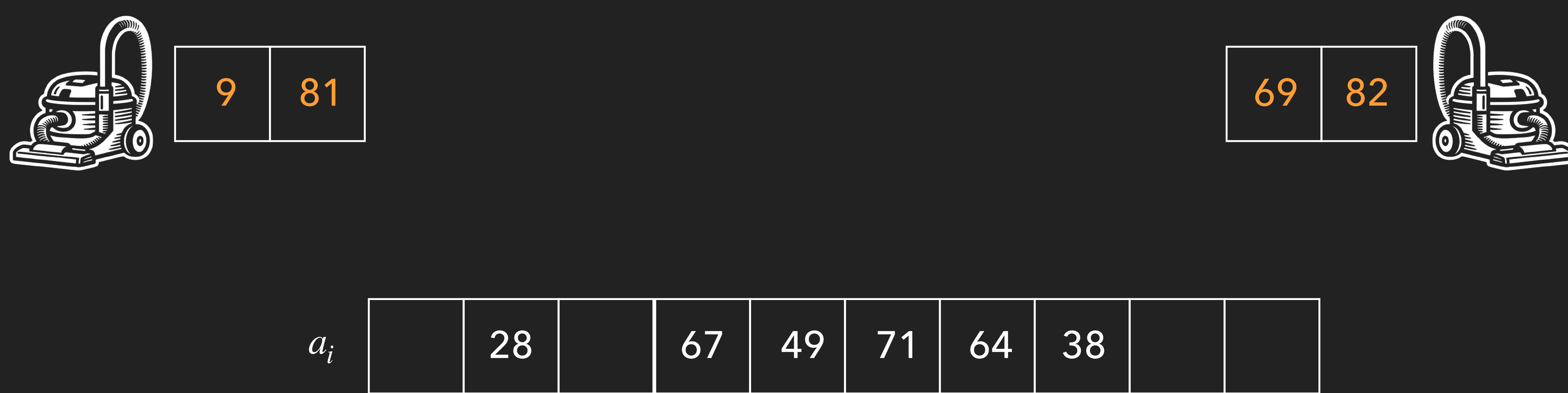
Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

In round i, each Hoover sucks in items that costs i comparisons to be inserted.

Round 1



Double-Hoover Sort Algorithm

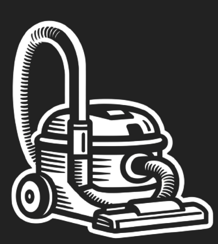
Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

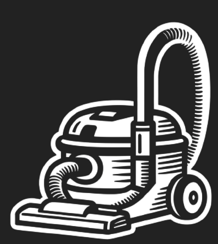
Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

In round i, each Hoover sucks in items that costs i comparisons to be inserted.

Round 2

69	82





9	81

a_i		28		67	49	71	64	38		

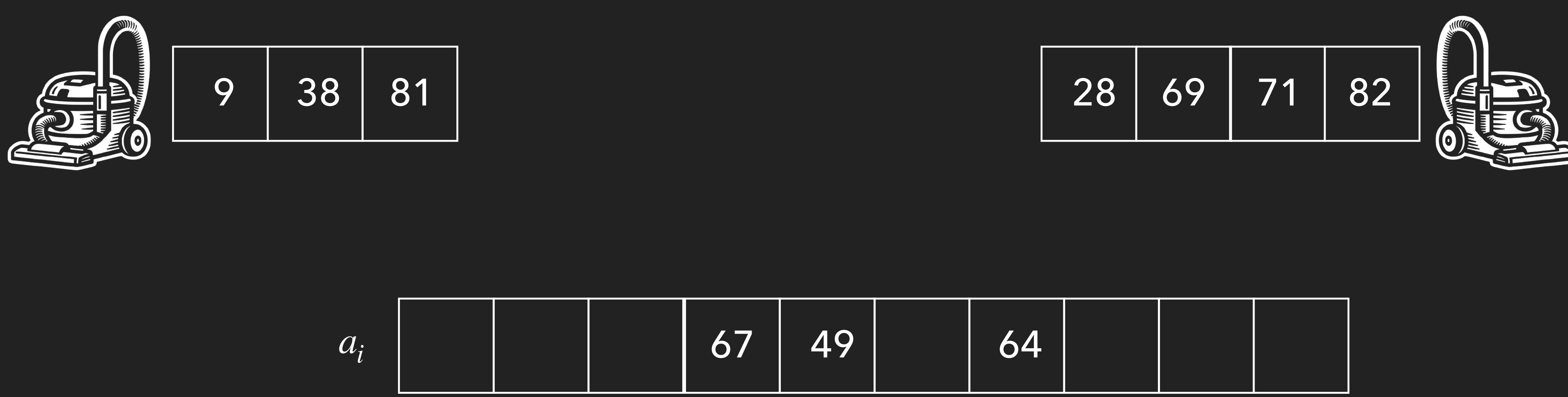
Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

In round i, each Hoover sucks in items that costs i comparisons to be inserted.

Round 2



Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

Round 3

28	69	71	82

9	38	81

a_i

			67	49		64			

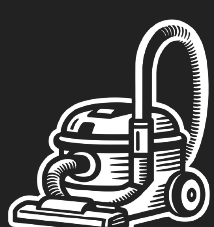
Double-Hoover Sort Algorithm

Idea: Bucket Sort the items w.r.t. $\hat{p}(i)$

Two "Hoovers", L and R, scan through the array repeatedly in $\log(n)$ rounds

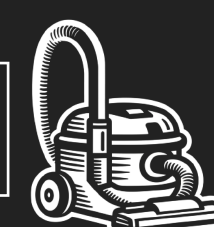
In round i, each Hoover sucks in items that costs i comparisons to be inserted.

Finally, combine items in both Hoovers



9	38	49	64	81

28	67	69	71	82

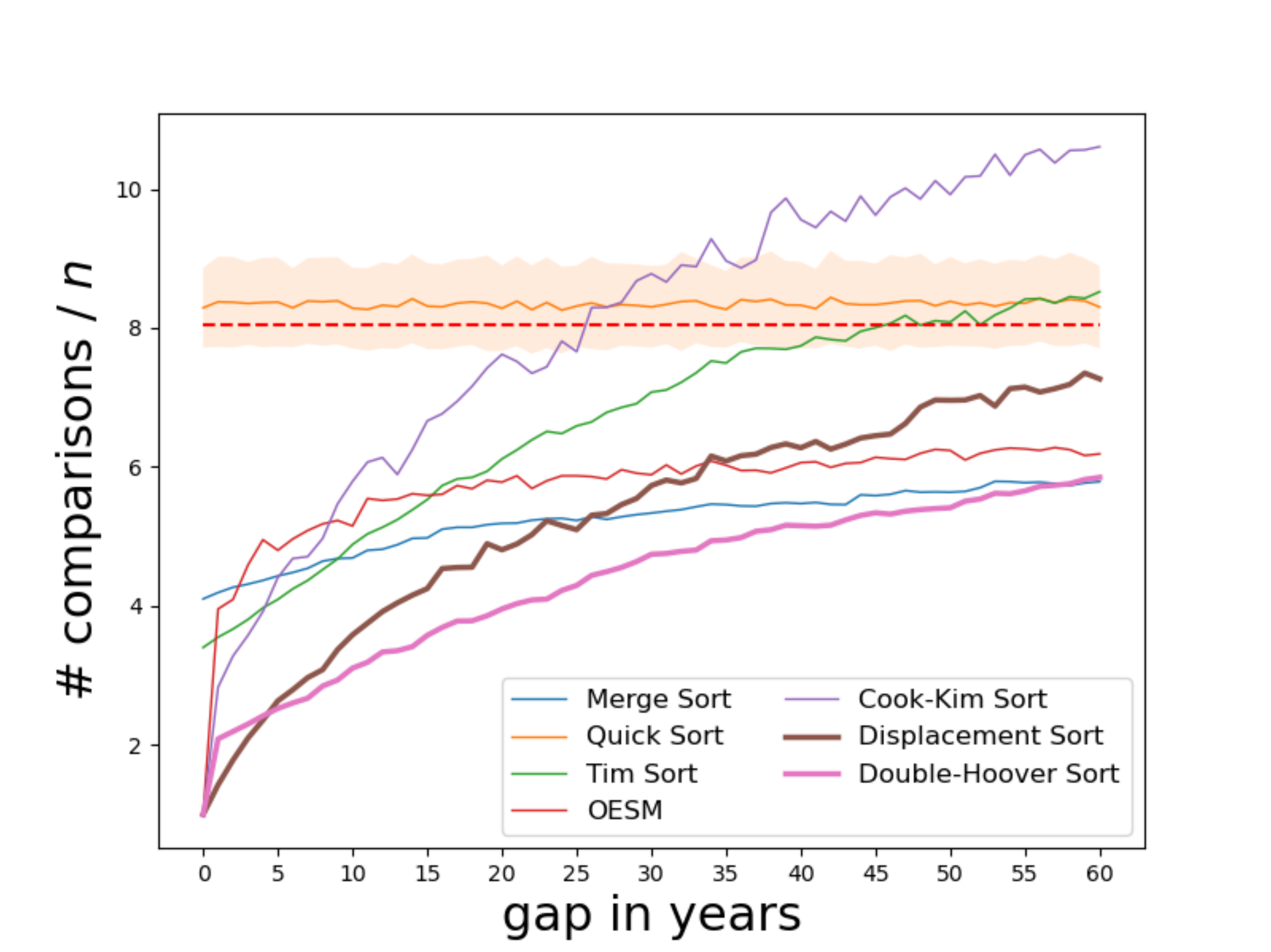


Each a_i is sucked into the Hoovers before round $\log(\min\{\eta_i^l, \eta_i^r\})$

Experiments: Country Population Ranking

Sorting countries by population (n=261)

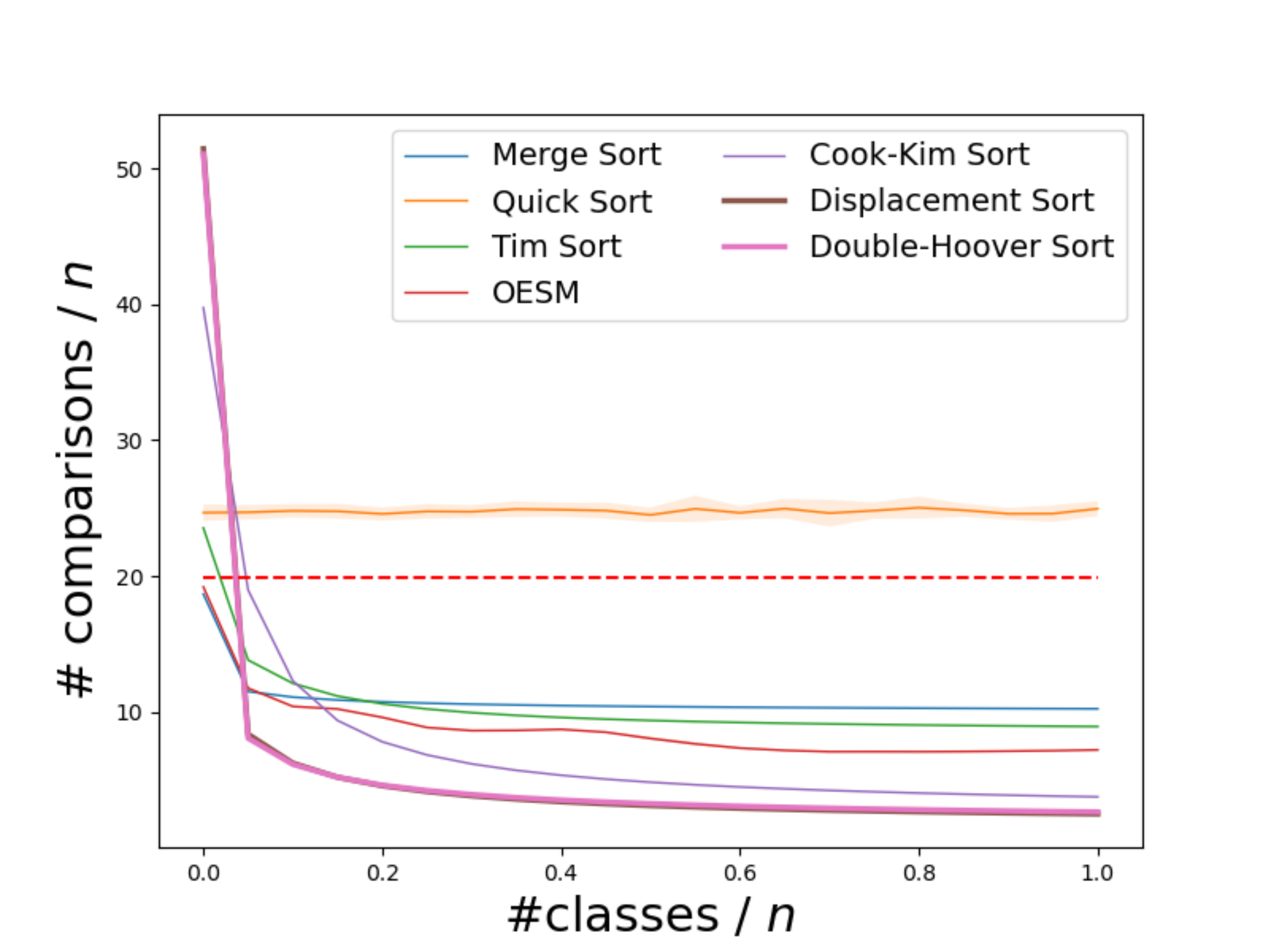
Predictions: ranking x years ago



Experiments: Class Setting

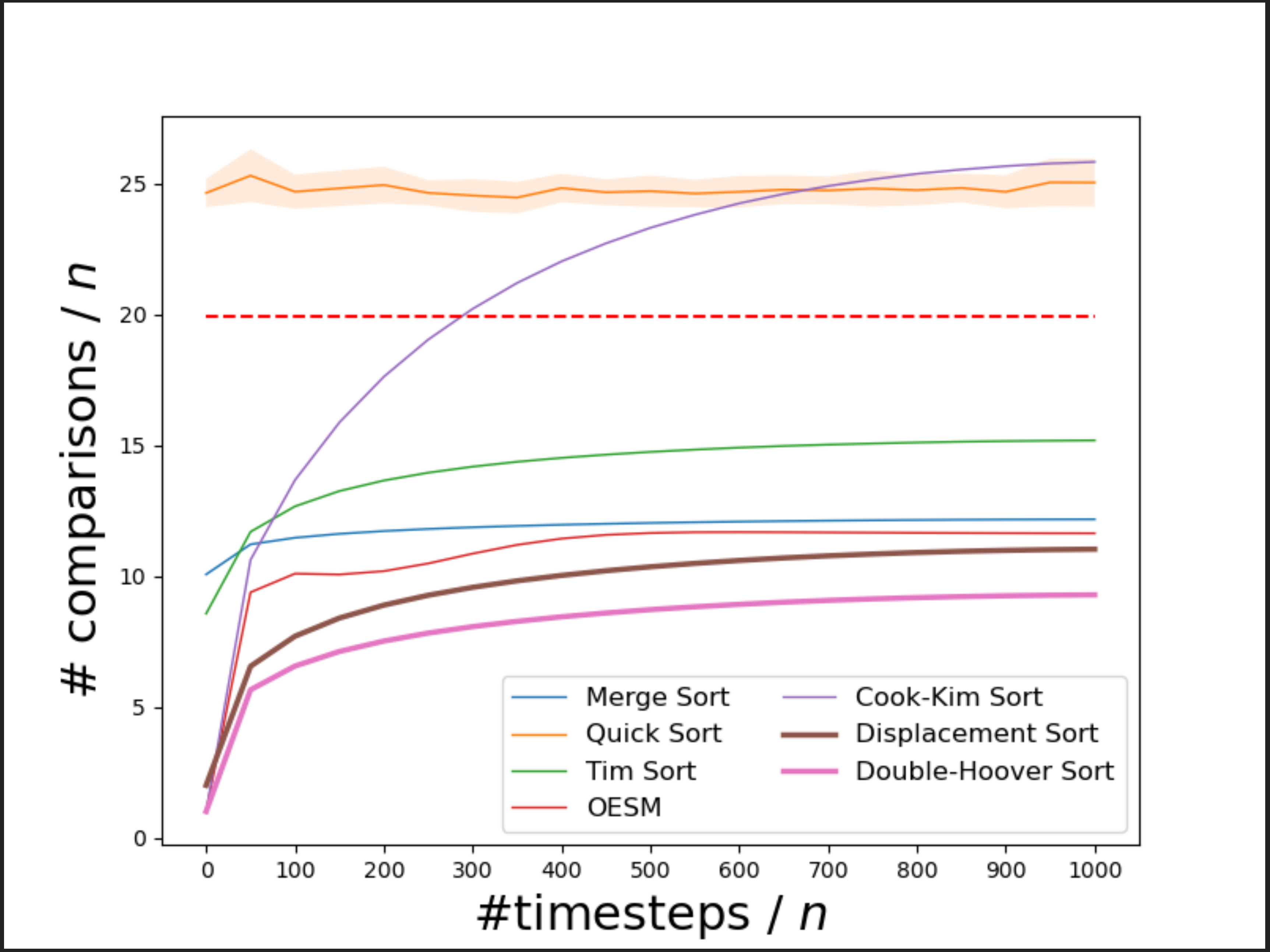
Classes of consecutive items (n=1,000,000)

Predictions: random position within class



Experiments: Decay Setting

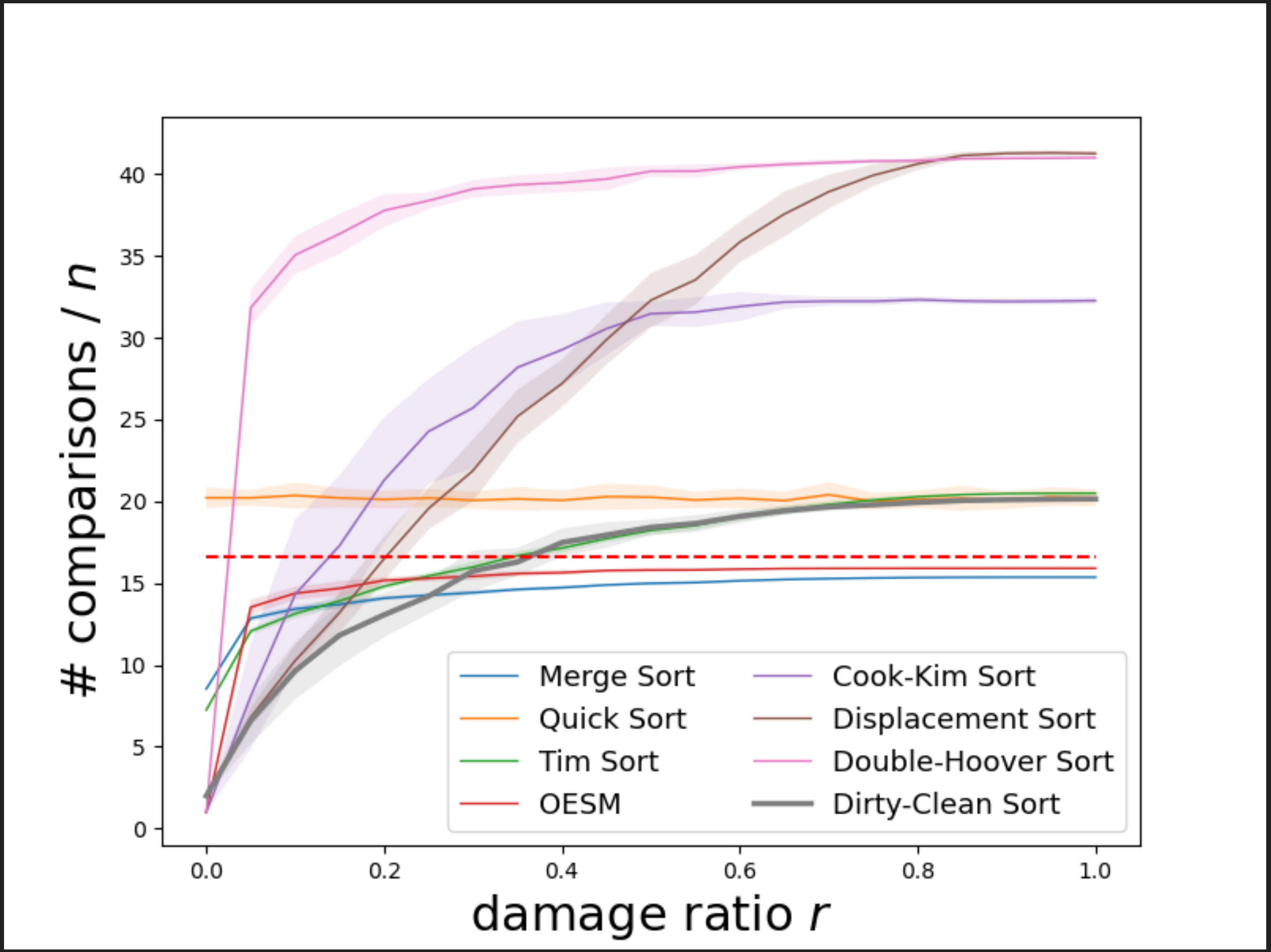
Repeatedly add ± 1 to $\hat{p}(i)$, for i random (n=1,000,000)



Experiments: Bad-Dominating Setting

Fraction r of items damaged (n=100,000)

$\hat{<}$ random if an item damaged, otherwise correct



Experiments: Good-Dominating Setting

Fraction r of items damaged (n=100,000)

$\hat{<}$ random if both items damaged, otherwise correct