FIBONACCI HEAPS

partially based on notes by Jeff Erickson

(not CLRS)

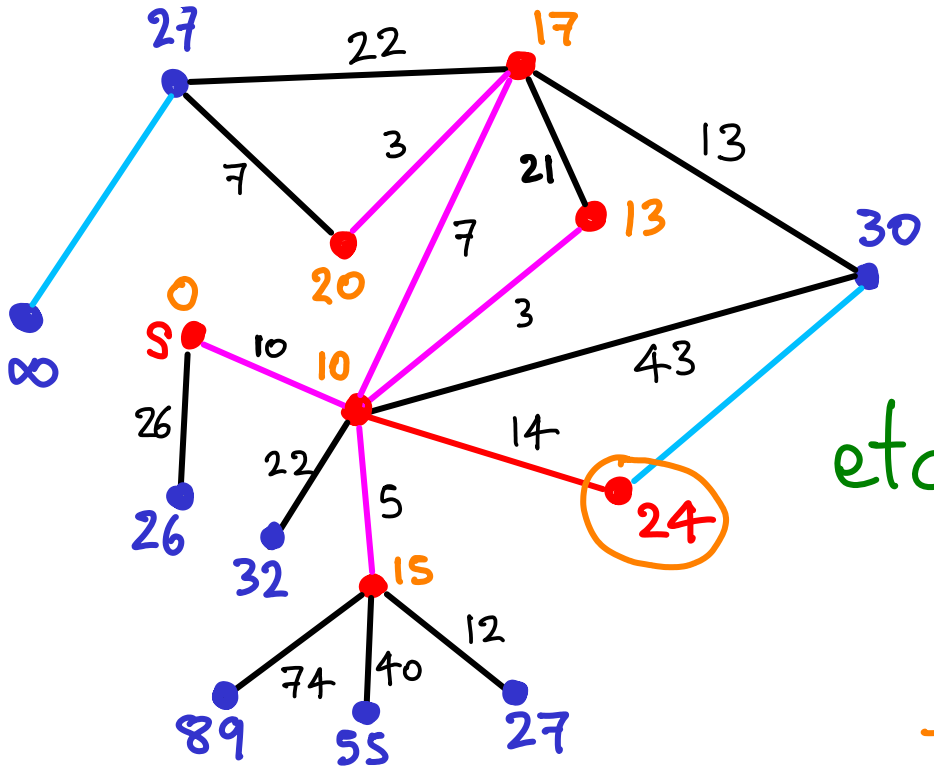
HEAPS:	REGULAR	BINOMIAL
REPORT MIN:	$O(1)$	←
EXTRACT MIN:	$O(\log n)$	←
INSERT:	$O(\log n)$ $O(1)$ amortized	←
DECREASE KEY:	$O(\log n)$	←
DELETE:	$O(\log n)$	←
MERGE/UNION:	$O(n)$	$O(\log n)$

HEAPS:	REGULAR	BINOMIAL	FIBONACCI
REPORT MIN:	O(1)	←	←
EXTRACT MIN:	O(logn)	←	O(n) O(logn) amortized
INSERT:	O(logn) O(1) amortized	←	O(1)
DECREASE KEY:	O(logn)	←	O(n) O(1) amortized
DELETE:	O(logn)	←	O(n) O(logn) amortized
MERGE/UNION:	O(n)	O(logn)	O(1)

HEAPS:

	REGULAR	BINOMIAL	FIBONACCI
REPORT MIN:	O(1)	←	←
EXTRACT MIN:	O(logn)	←	O(n) O(logn) amortized
INSERT:	O(logn) O(1) amortized	←	O(1)
DECREASE KEY:	O(logn)	←	* O(n) O(1) amortized
DELETE:	O(logn)	←	O(n) O(logn) amortized
MERGE/UNION:	O(n)	O(logn)	* O(1)

DIJKSTRA'S ALGORITHM for SSSP



(after initializing)

while pr.queue not empty

- x: extract-min & add edge to T
- mark x → in T.
- for each neighbor v of x
  - RELAX(x,v)

time : $O(V^2)$ or $O(E \log V)$

Fibonacci heap: E * decrease-key + V * extract = E * O(1) + V * logV [amortized]

HEAPS:	REGULAR	BINOMIAL	FIBONACCI
REPORT MIN:	O(1)	←	←
EXTRACT MIN:	O(logn)	←	O(n) O(logn) amortized
INSERT:	O(logn) O(1) amortized	←	O(1)
DECREASE KEY:	O(logn)	←	O(n) O(1) amortized
DELETE:	O(logn)	←	O(n) O(logn) amortized
MERGE/UNION:	O(n)	O(logn)	O(1)

HEAPS:	REGULAR	BINOMIAL	FIBONACCI
REPORT MIN:	O(1)	←	←
EXTRACT MIN:	O(logn)	←	O(n) O(logn) amortized
INSERT:	O(logn) O(1) amortized	←	O(1)
DECREASE KEY:	O(logn)	←	O(n) O(1) amortized
MERGE/UNION:	O(n)	O(logn)	O(1)

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

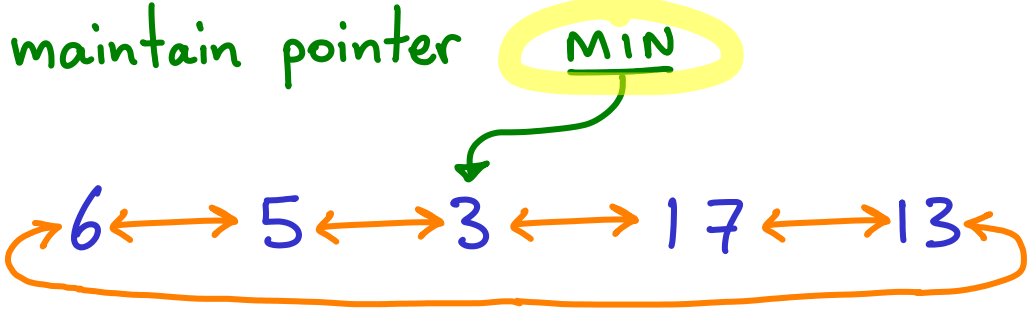
} What if we only needed these?

FIBONACCI HEAPS

• MIN • INSERT • MERGE	O(1)
DECREASE	O(n) O(1) am.
EXTRACT MIN	O(n) O(log n) am.

What if we only needed these?

maintain pointer MIN



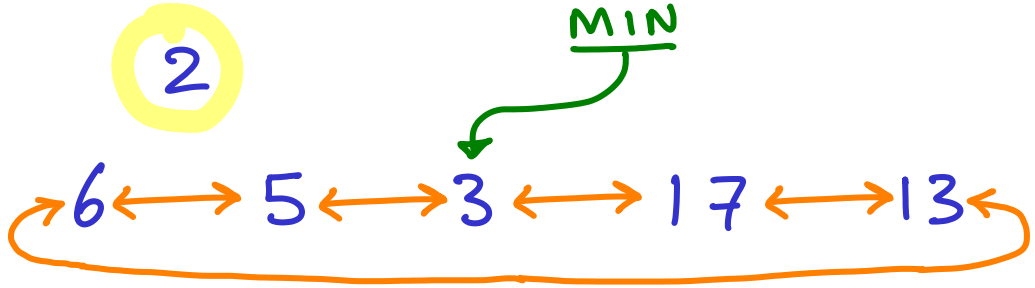
FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE	O(n) O(1) am.
EXTRACT MIN	O(n) O(log n) am.

What if we only needed these?



FIBONACCI HEAPS

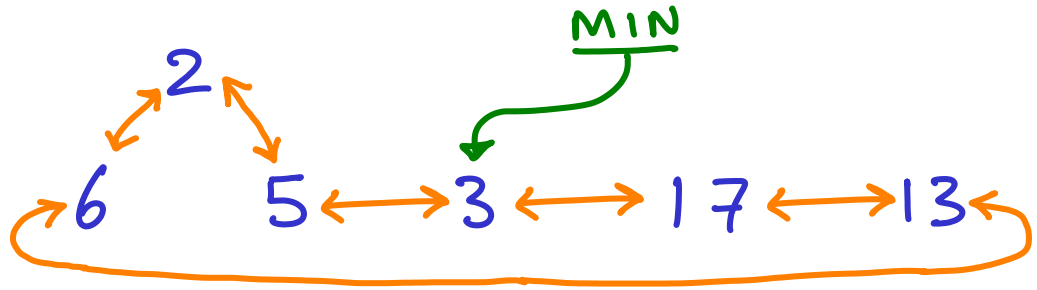
- MIN
- INSERT
- MERGE

} O(1)

DECREASE: O(n), O(1) am.

EXTRACT MIN: O(n), O(log n) am.

What if we only needed these?



FIBONACCI HEAPS

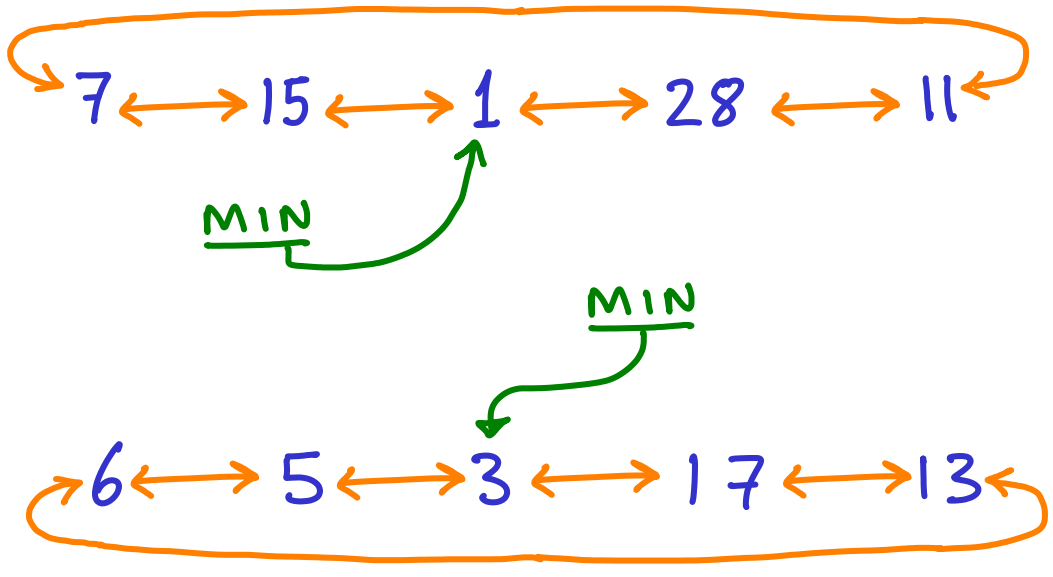
- MIN
- INSERT
- MERGE

} O(1)

DECREASE: O(n), O(1) am.

EXTRACT MIN: O(n), O(log n) am.

What if we only needed these?



FIBONACCI HEAPS

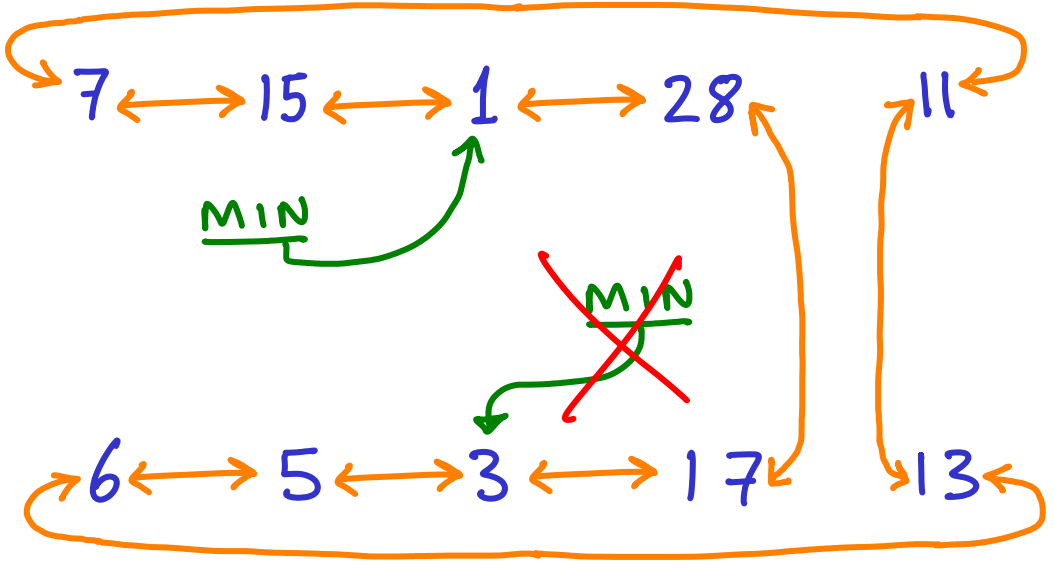
- MIN
- INSERT
- MERGE

} O(1)

DECREASE: O(n), O(1) am.

EXTRACT MIN: O(n), O(log n) am.

What if we only needed these?



FIBONACCI HEAPS

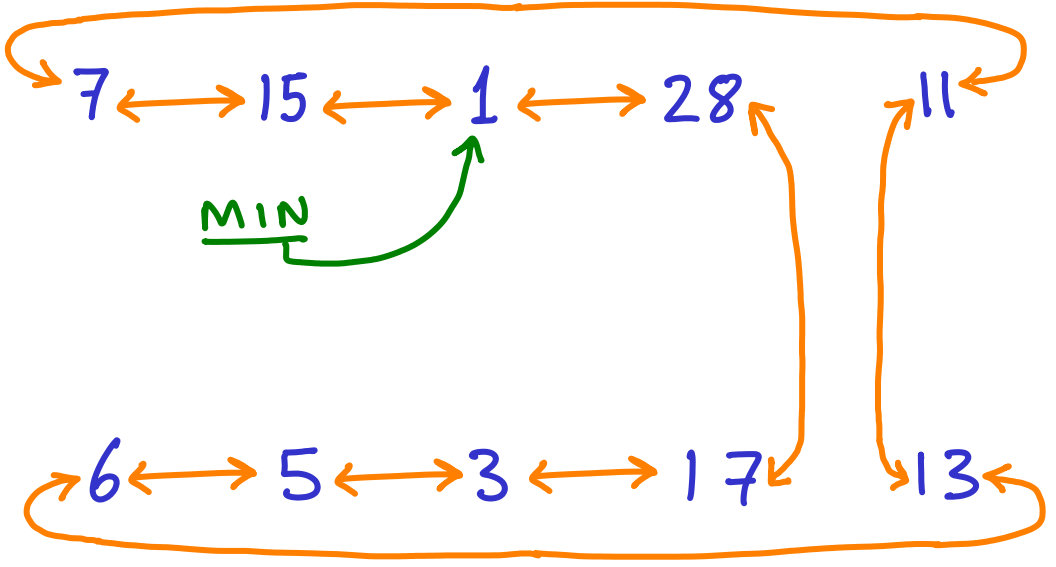
- MIN
- INSERT
- MERGE

} O(1)

DECREASE	O(n) O(1) am.
EXTRACT MIN	O(n) O(log n) am.

What if we only needed these?

In fact we get DECREASE in O(1)
and EXTRACT MIN in O(n)



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

What if we only needed MIN, INSERT, MERGE, EXTRACT MIN

$O(1)$ $O(1)$ $O(1)$ $O(\log n)$ am.

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

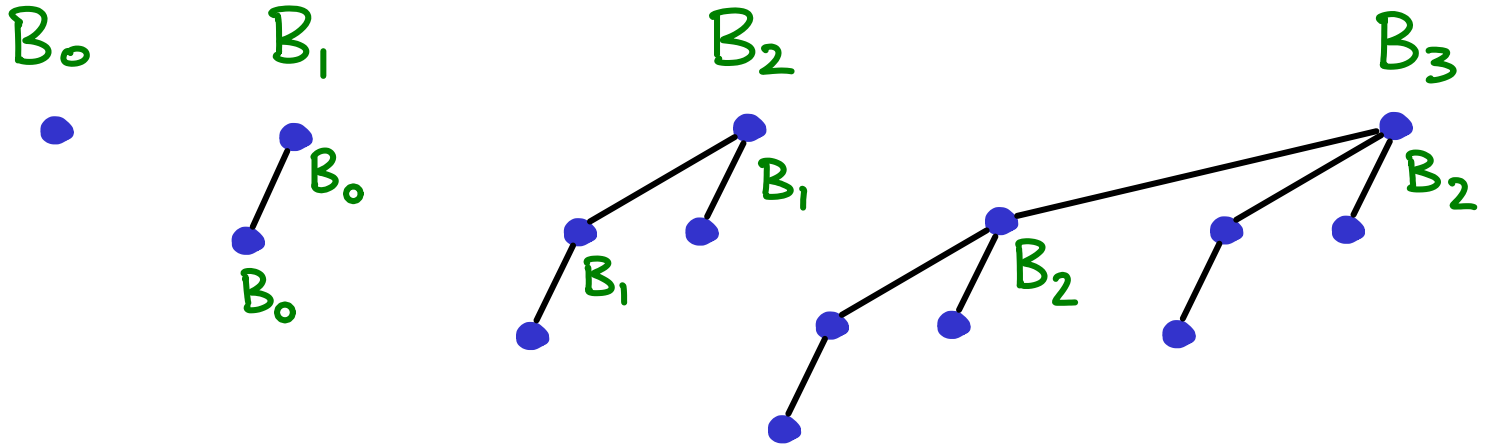
EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

rigid structure, maintained asap

binomial trees:



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

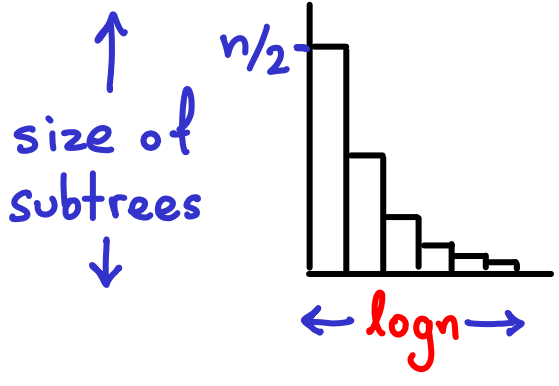
DECREASE O(n)
O(1) am.

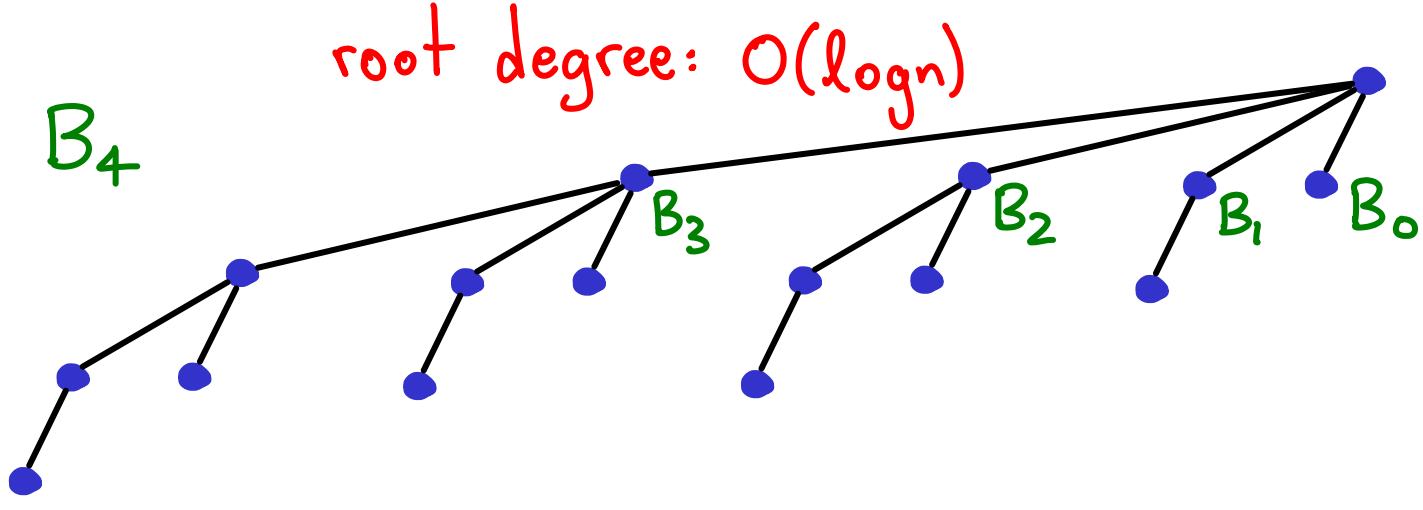
EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

rigid structure, maintained asap





FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

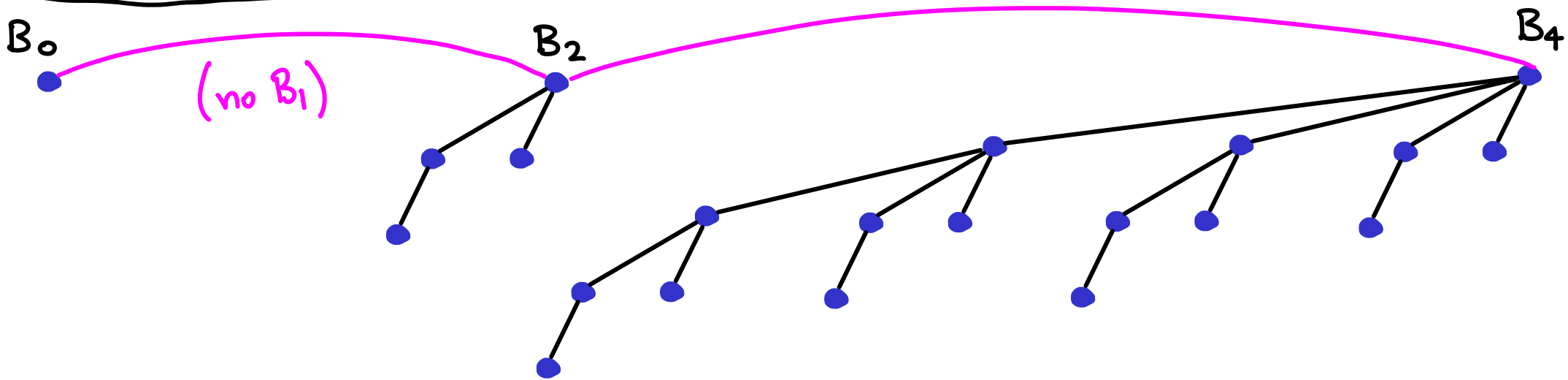
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

rigid structure, maintained asap



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

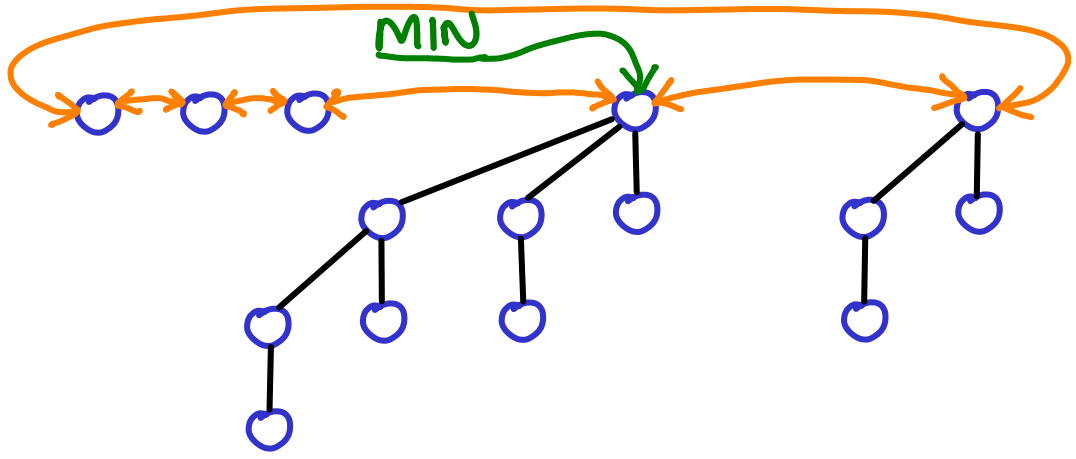
EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

~~rigid structure, maintained asap~~

Instead:



- list of binomial trees
- relaxed structure
  (allow > 1 of each)
  (not sorted by size)

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

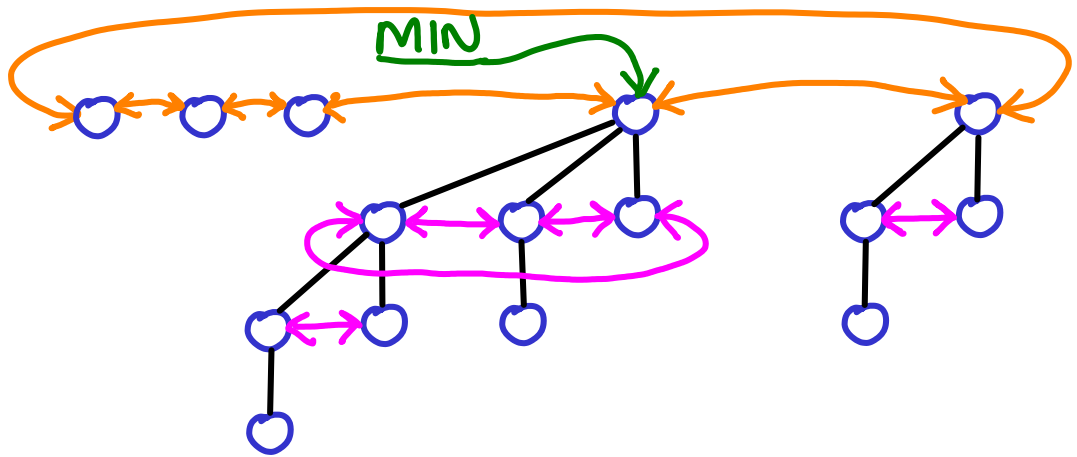
DECREASE O(n) / O(1) am.

EXTRACT MIN O(n) / O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

~~rigid structure, maintained asap~~



- list of binomial trees
- relaxed structure
  (allow > 1 of each)
  (not sorted by size)
- extra pointers

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

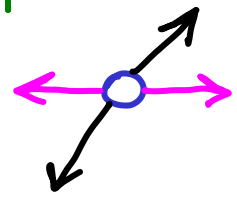
What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

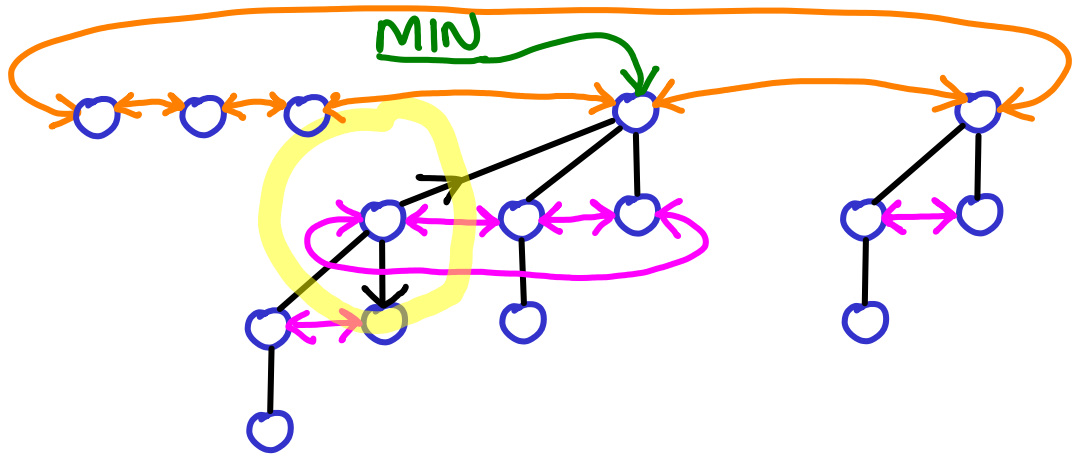
Recall, Binomial heap: O(1), O(1) am., O(log n), O(log n)

~~rigid structure, maintained asap~~

- list of binomial trees
- relaxed structure
  (allow > 1 of each)
  (not sorted by size)
- extra pointers

4 pointers per node





FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

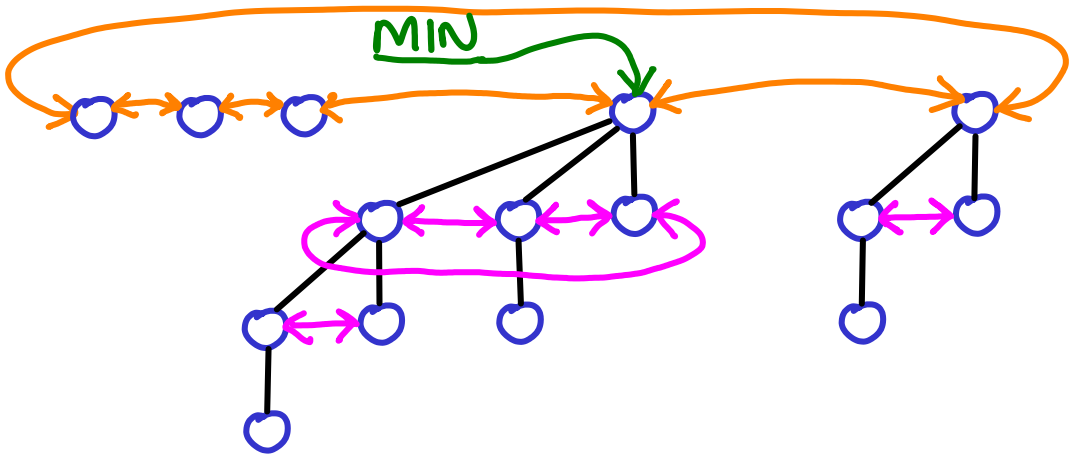
} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

* MIN, INSERT, MERGE
  - ↳ like linked list solution
  - ↳ be lazy: allow relaxed structure



- list of binomial trees
- relaxed structure
  (allow > 1 of each)
  (not sorted by size)
- extra pointers

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

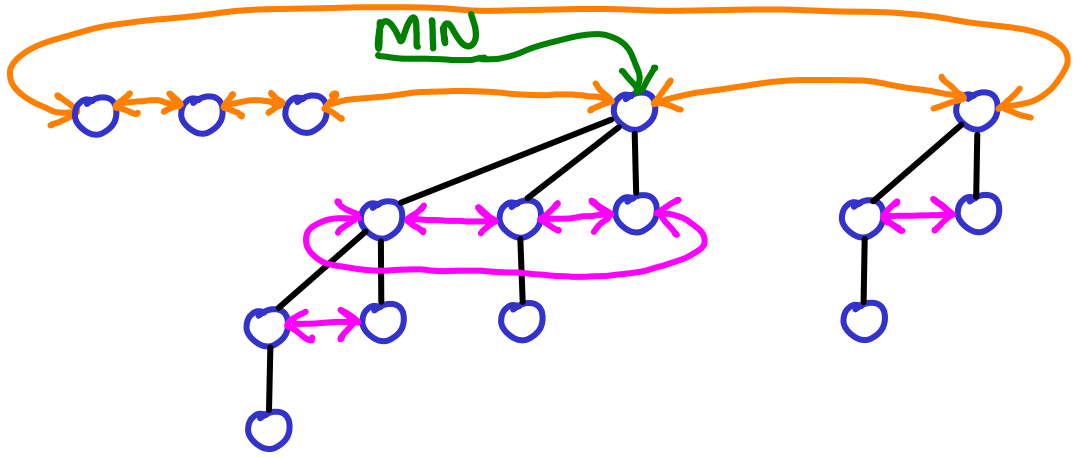
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

What if we only needed MIN, INSERT, MERGE, EXTRACT
O(1) O(1) O(1) O(log n) am.

* MIN, INSERT, MERGE
  - ↳ like linked list solution
  - ↳ be lazy: allow relaxed structure

* EXTRACT → pay for laziness: unrelax
  - ↳ ensure ≤1 of each binomial tree size



- list of binomial trees
- relaxed structure
  (allow >1 of each)
  (not sorted by size)
- extra pointers

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

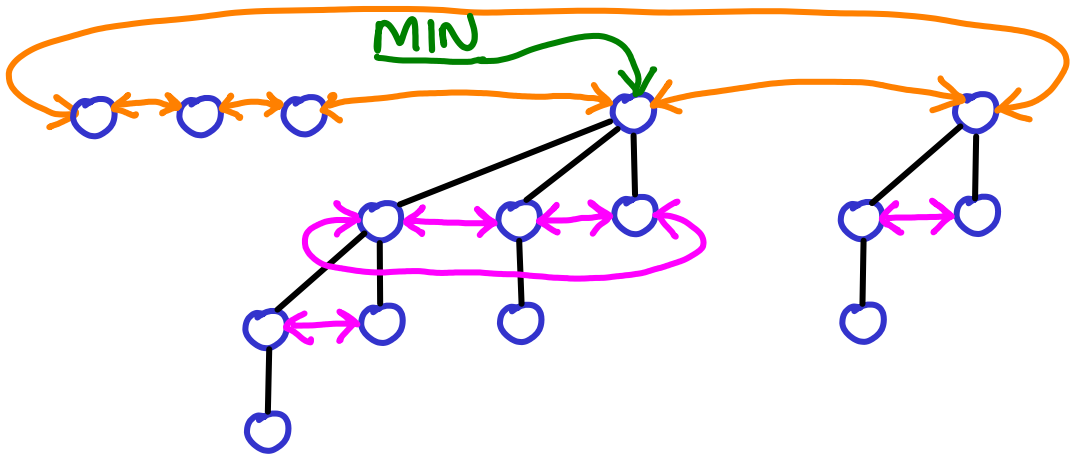
} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

EXTRACT MIN

1) Delete MIN node
2) Set children's parent pointers to NULL



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

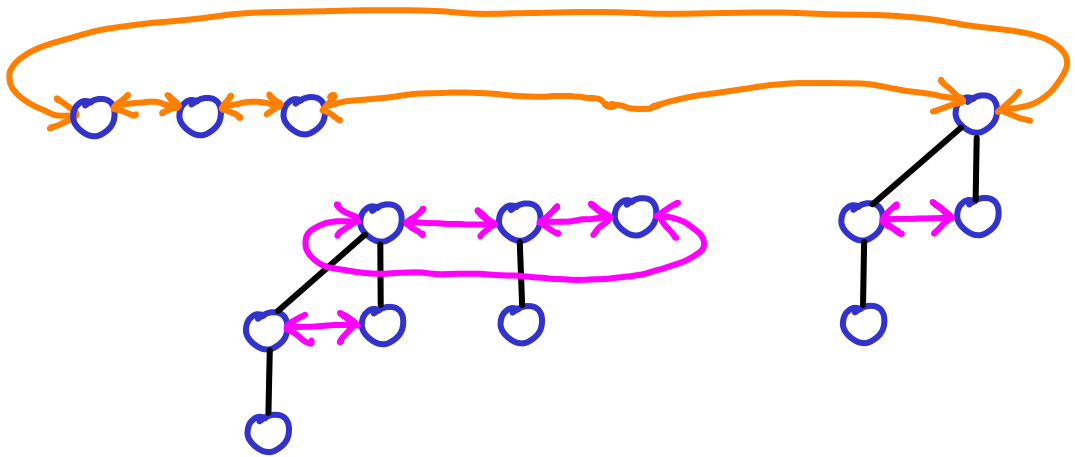
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

EXTRACT MIN

1) Delete MIN node
2) Set children's parent pointers to NULL

O(deg(MIN))
= ?



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE: O(n), O(1) am.

EXTRACT MIN: O(n), O(logn) am.

EXTRACT MIN

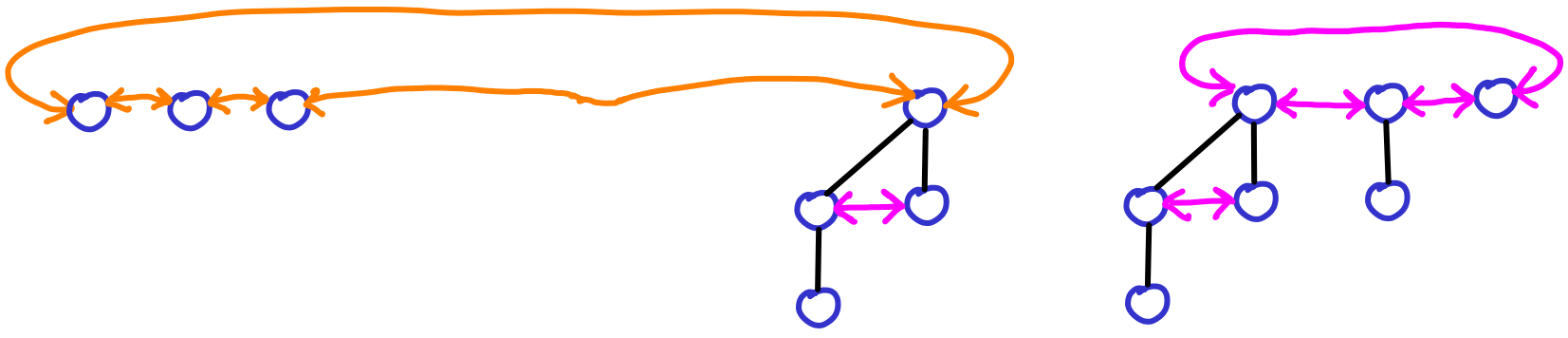
1) Delete MIN node
2) Set children's parent pointers to NULL

O(logn)

↓

binomial heap

↓



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

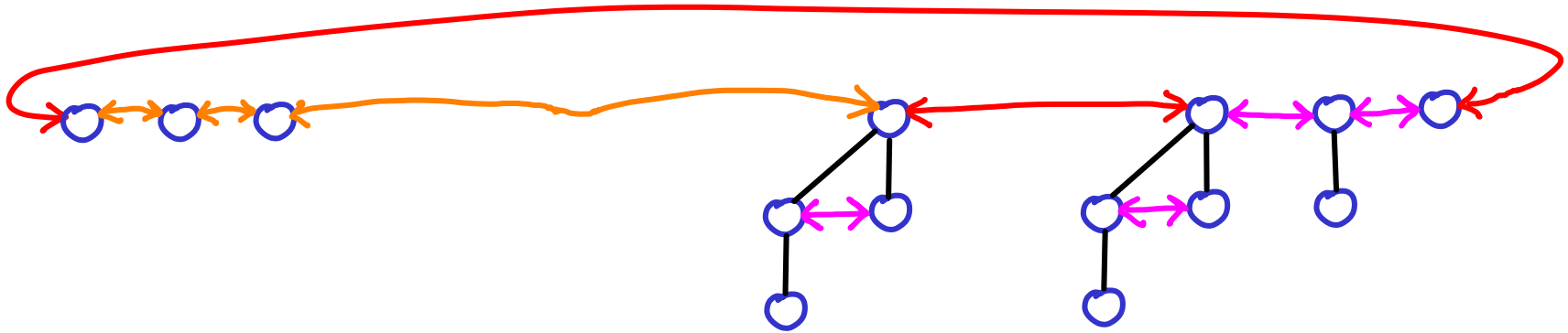
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

EXTRACT MIN

1) Delete MIN node
2) Set children's parent pointers to NULL
3) Link root lists: O(1)

O(logn)



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

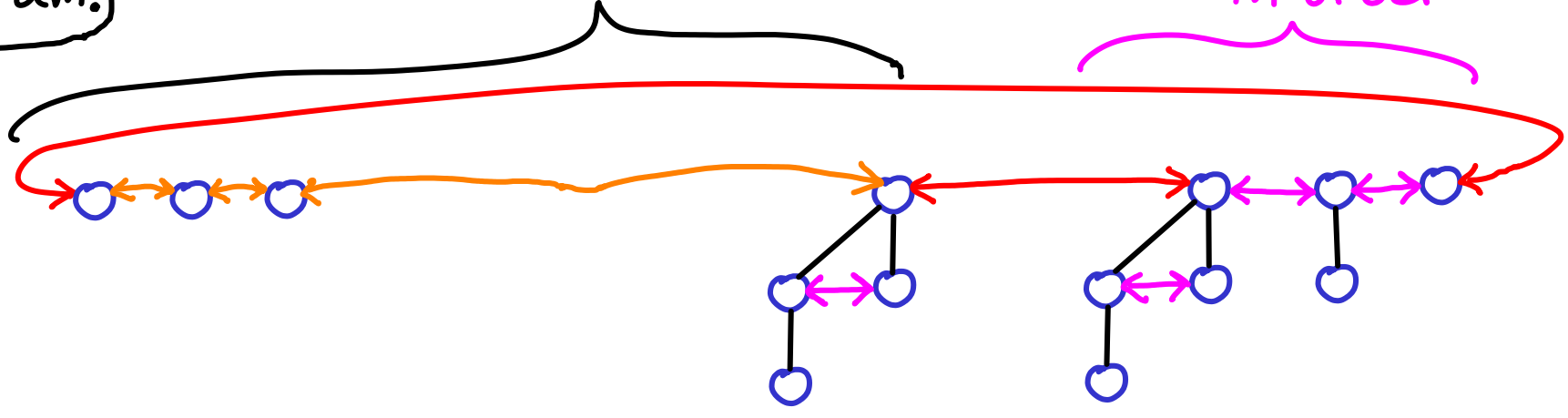
EXTRACT MIN O(n)
O(logn) am.

EXTRACT MIN

1) Delete MIN node
2) Set children's parent pointers to NULL O(logn)
3) Link root lists: O(1)

possibly unordered binomial trees
& > 1 of each size

in order



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

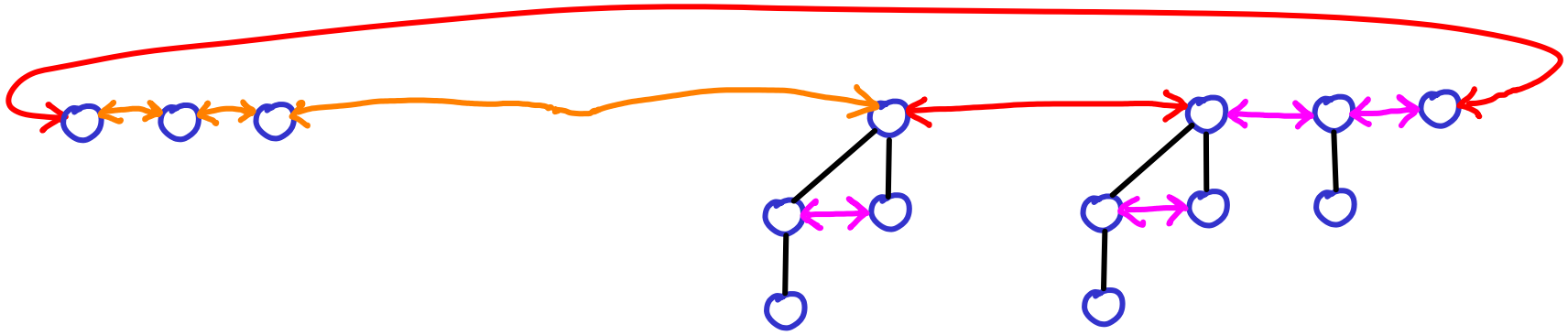
} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

EXTRACT MIN

1) Delete MIN node
2) Set children's parent pointers to NULL
3) Link root lists: O(1) O(log n)
4) Make sure ≤1 binomial tree of each size



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

4) Make sure ≤ 1 binomial tree of each size.

This is like merging two binomial heaps except we have less structure

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

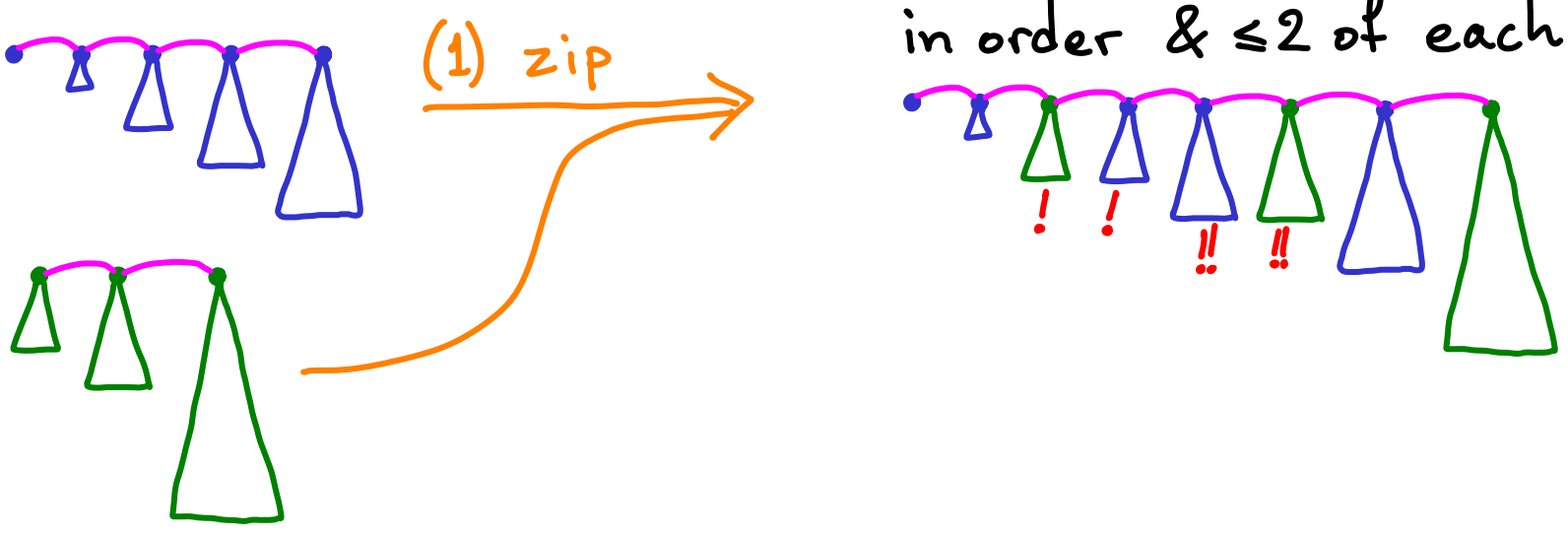
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.

This is like merging two binomial heaps except we have less structure

Recall binomial heap merge:



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

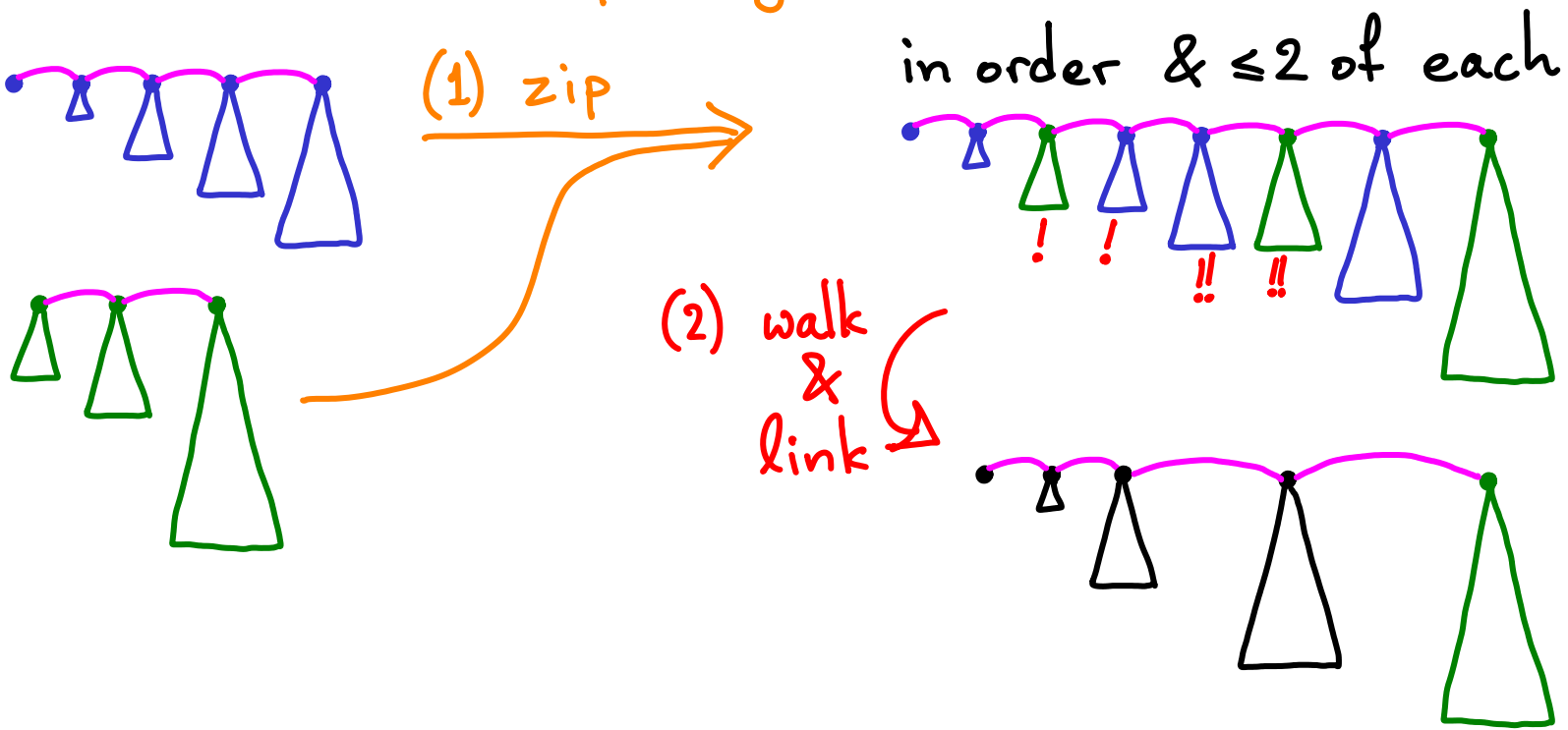
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.

This is like merging two binomial heaps except we have less structure

Recall binomial heap merge:



FIBONACCI HEAPS

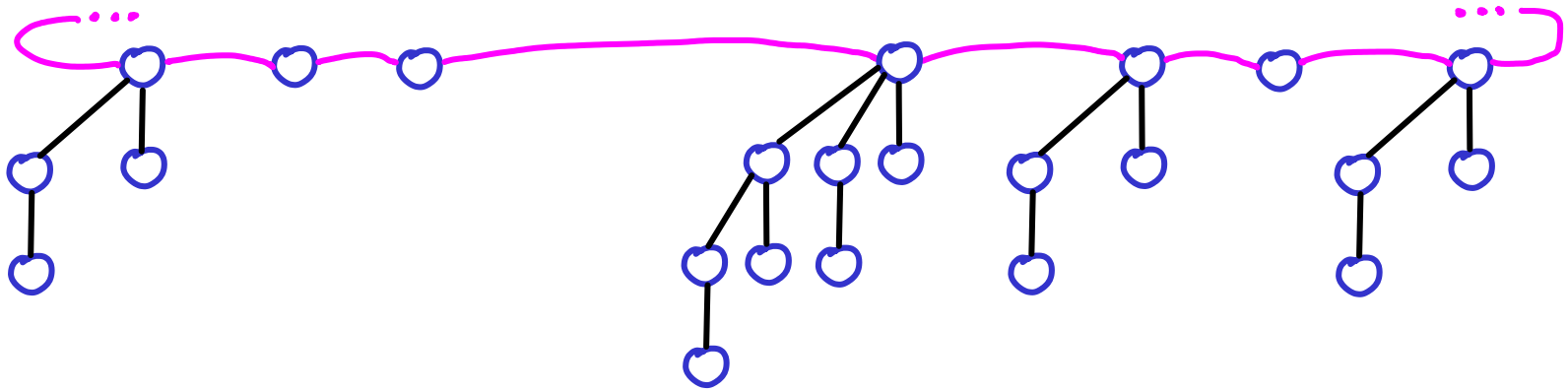
- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

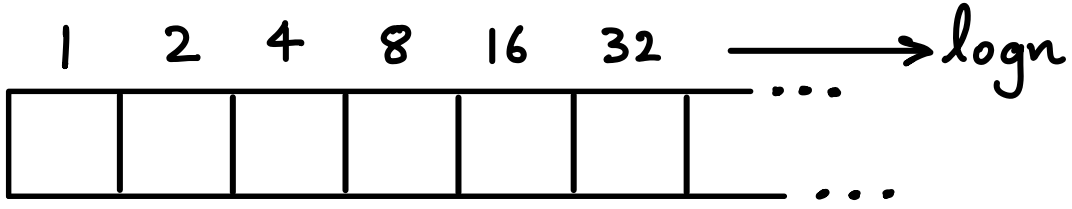
- MIN
- INSERT
- MERGE

} $O(1)$

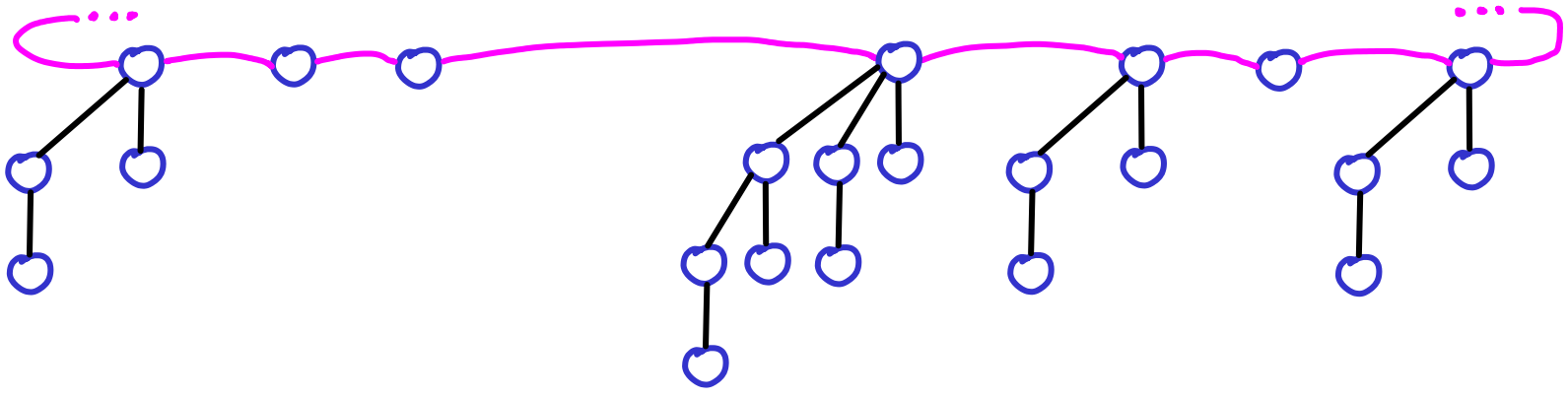
DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

4) Make sure ≤ 1 binomial tree of each size.



Make array of size $\log n$ = max degree



FIBONACCI HEAPS

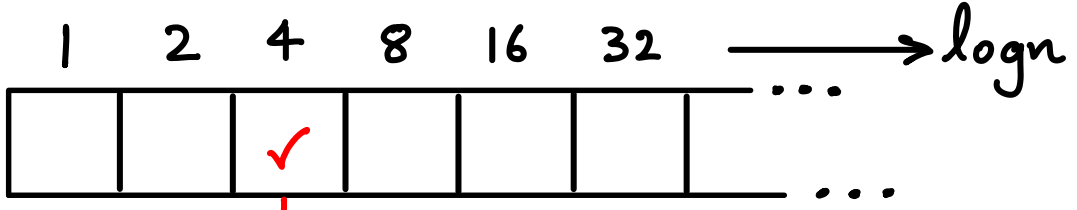
- MIN
- INSERT
- MERGE

} O(1)

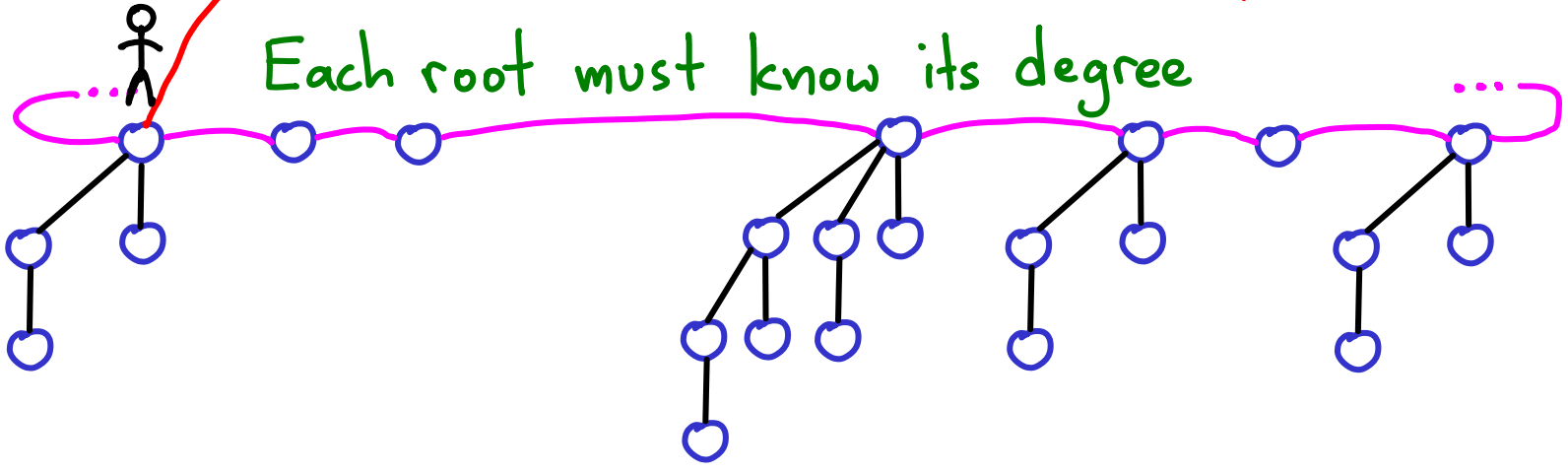
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



Traverse root list & link to array.
Also update MIN.



FIBONACCI HEAPS

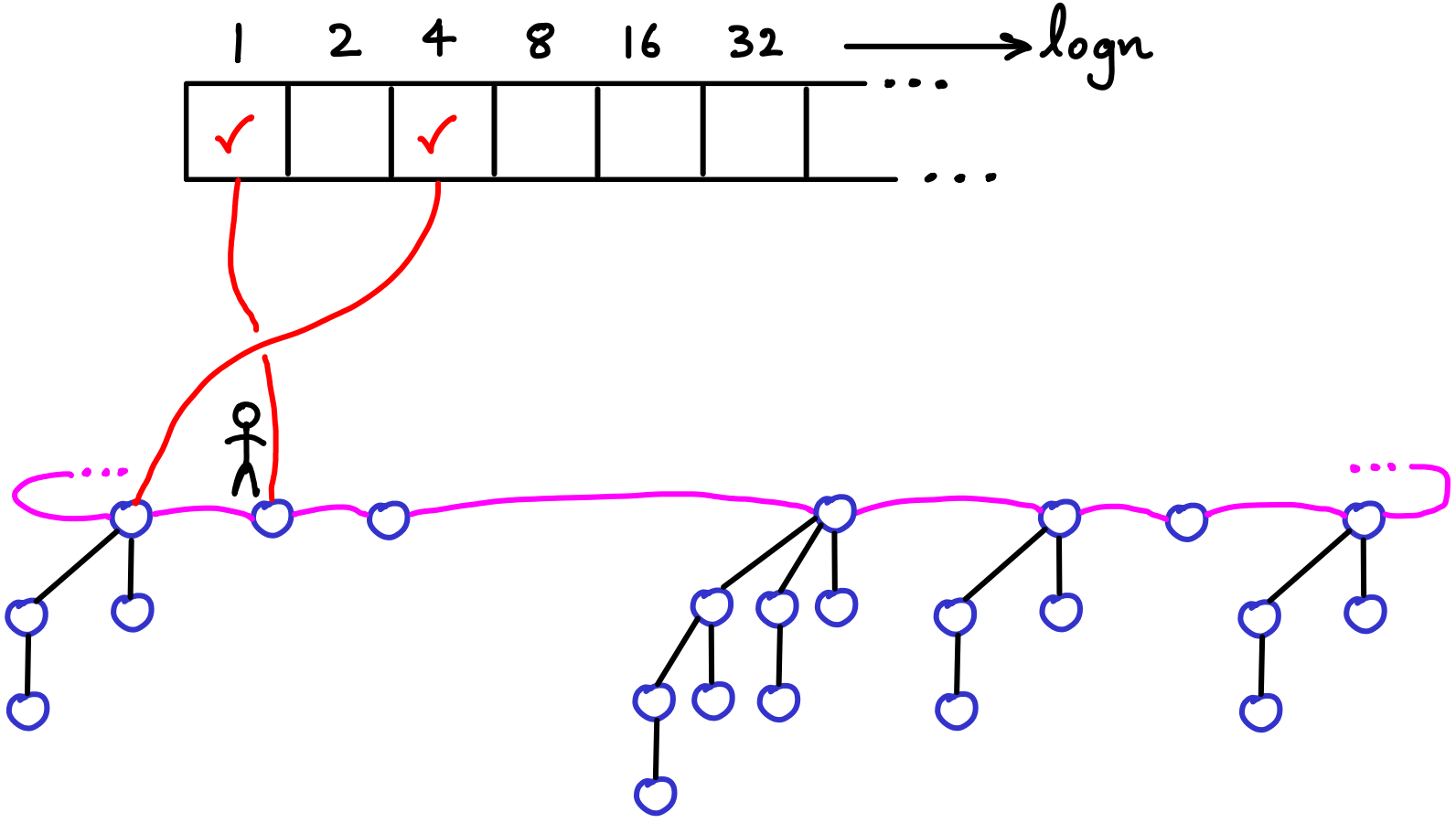
- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

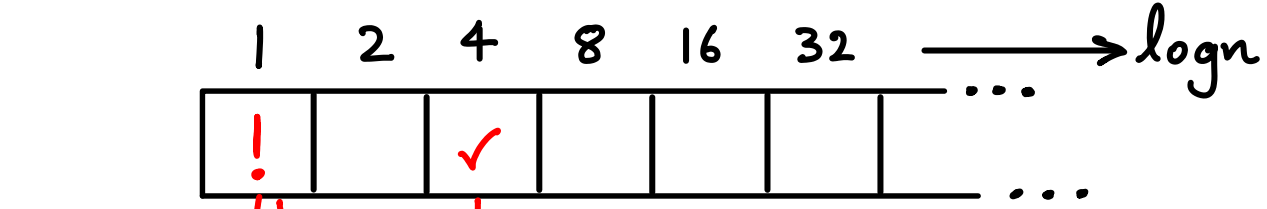
- MIN
- INSERT
- MERGE

} O(1)

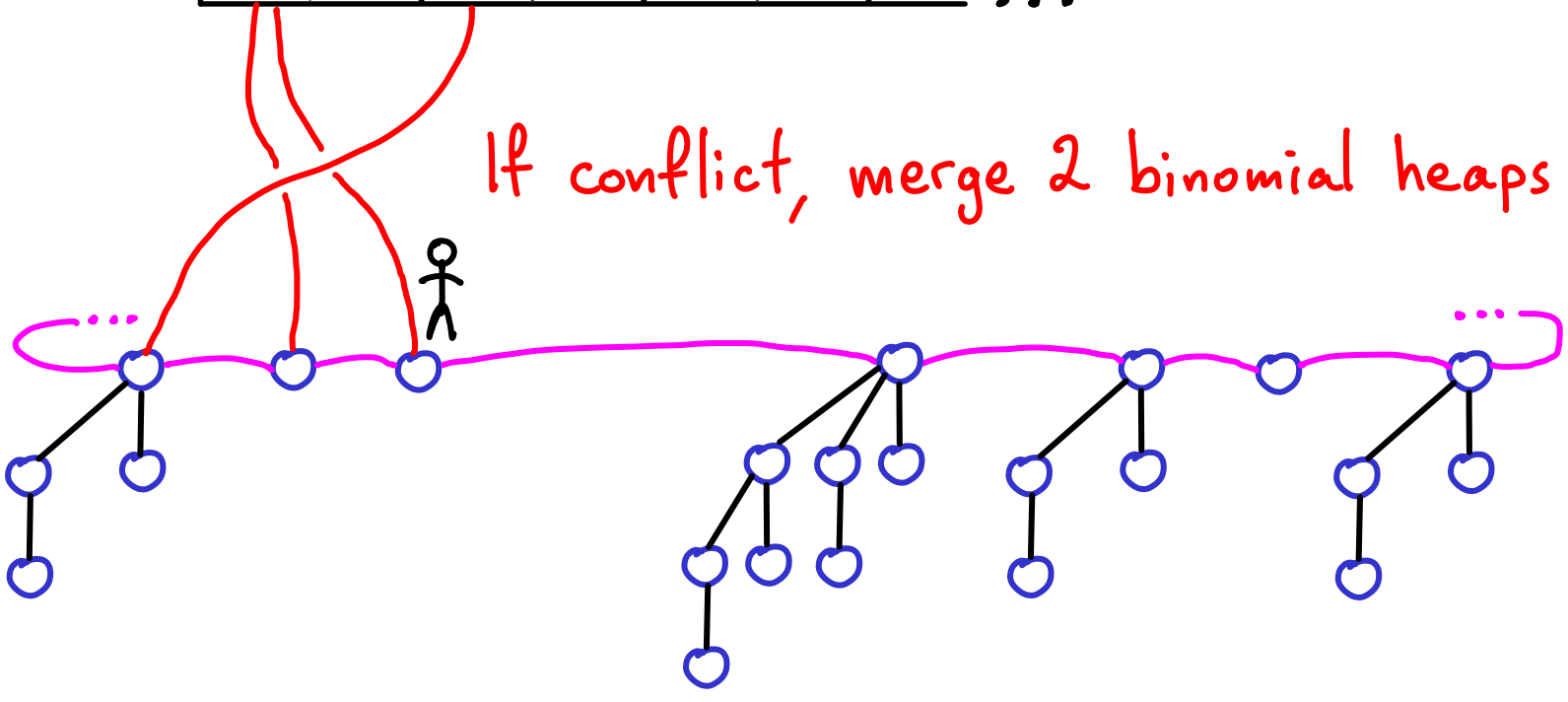
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



If conflict, merge 2 binomial heaps



FIBONACCI HEAPS

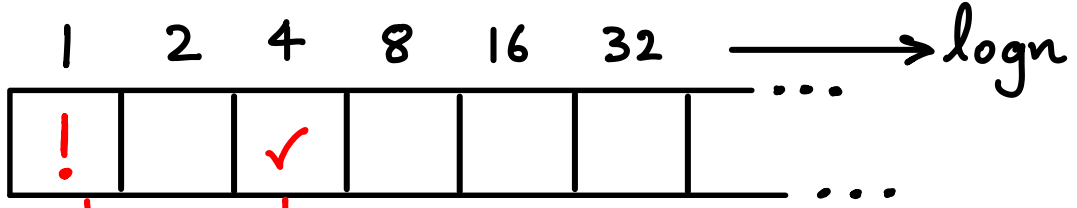
- MIN
- INSERT
- MERGE

} O(1)

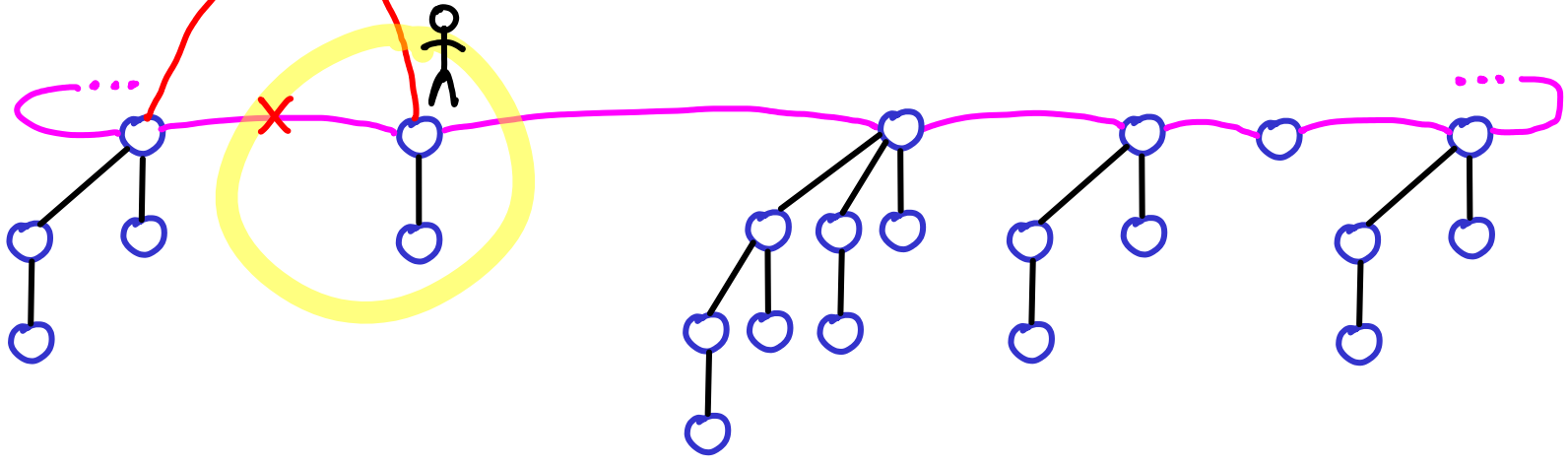
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



If conflict, merge 2 binomial heaps



FIBONACCI HEAPS

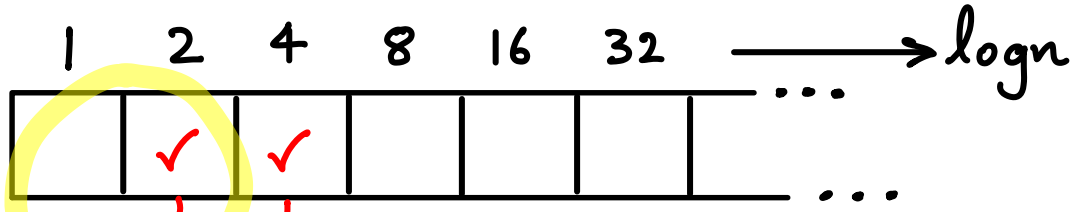
- MIN
- INSERT
- MERGE

} O(1)

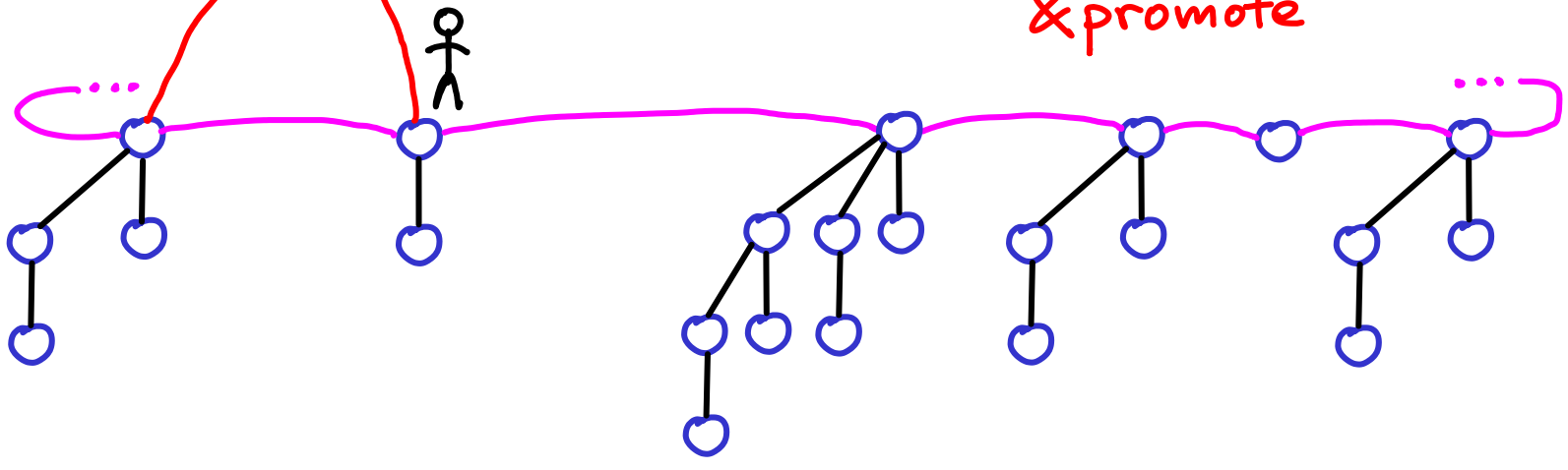
DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(log n) am.

4) Make sure ≤1 binomial tree of each size.



If conflict, merge 2 binomial heaps & promote



FIBONACCI HEAPS

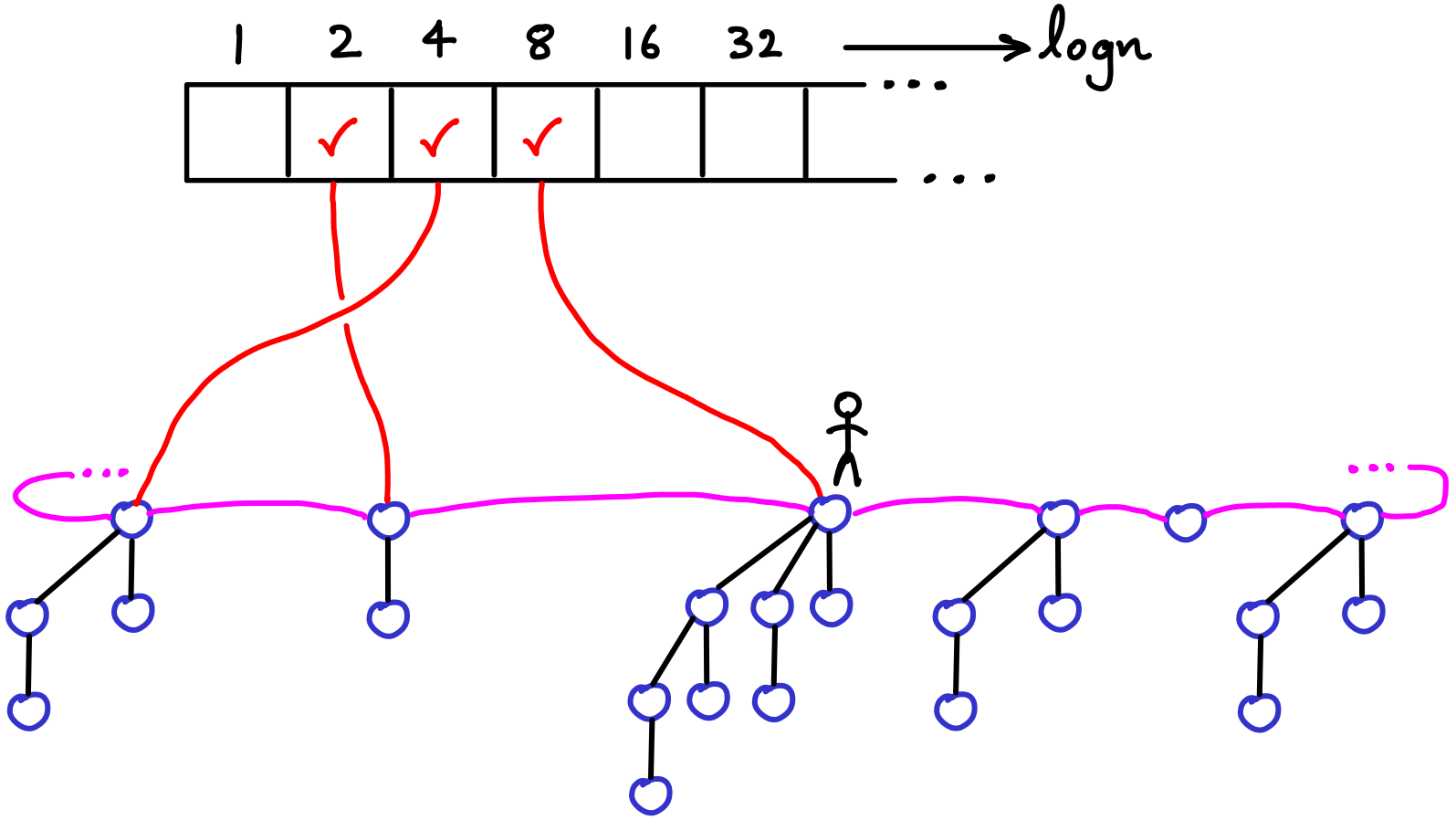
- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

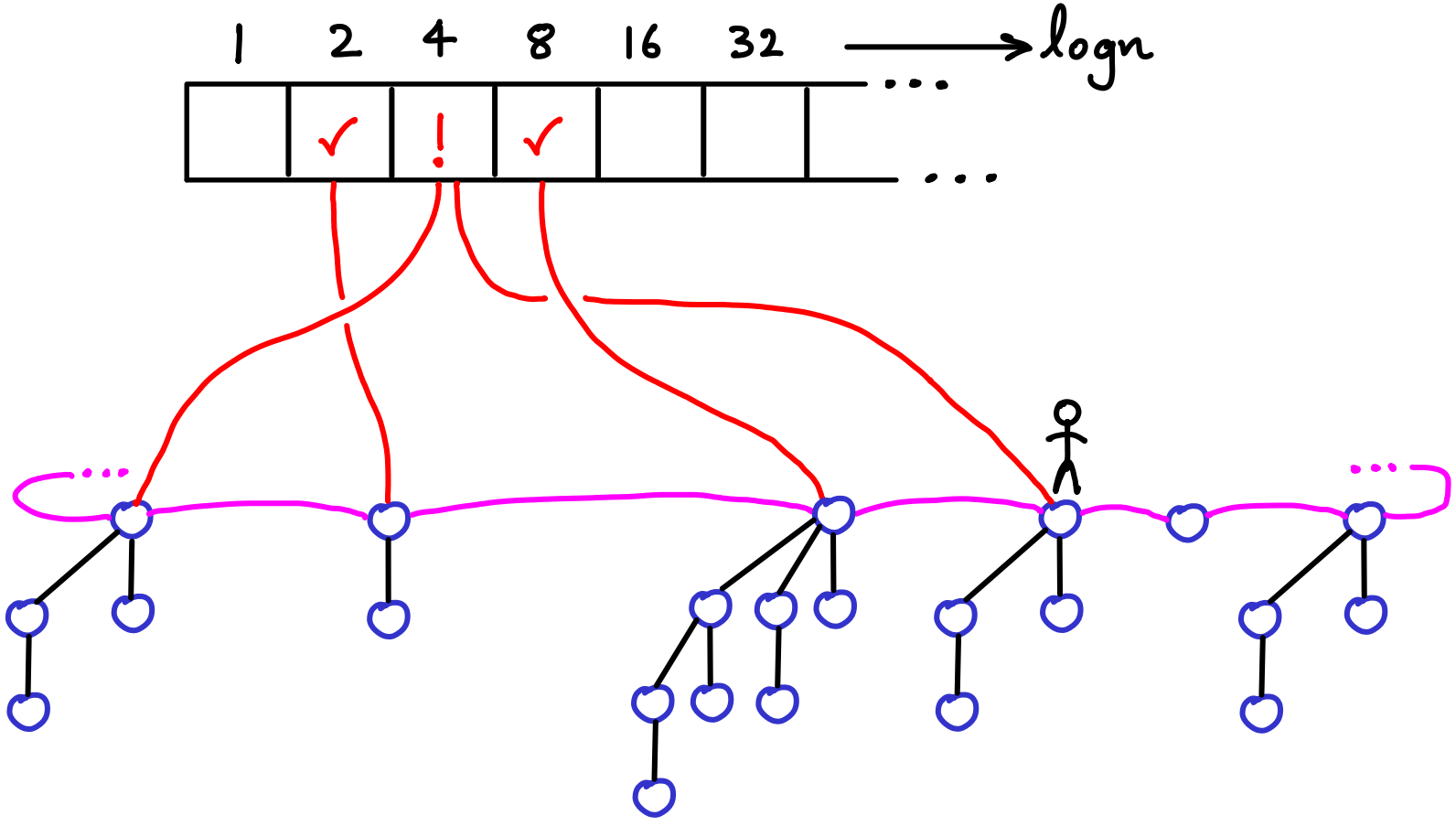
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(logn) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

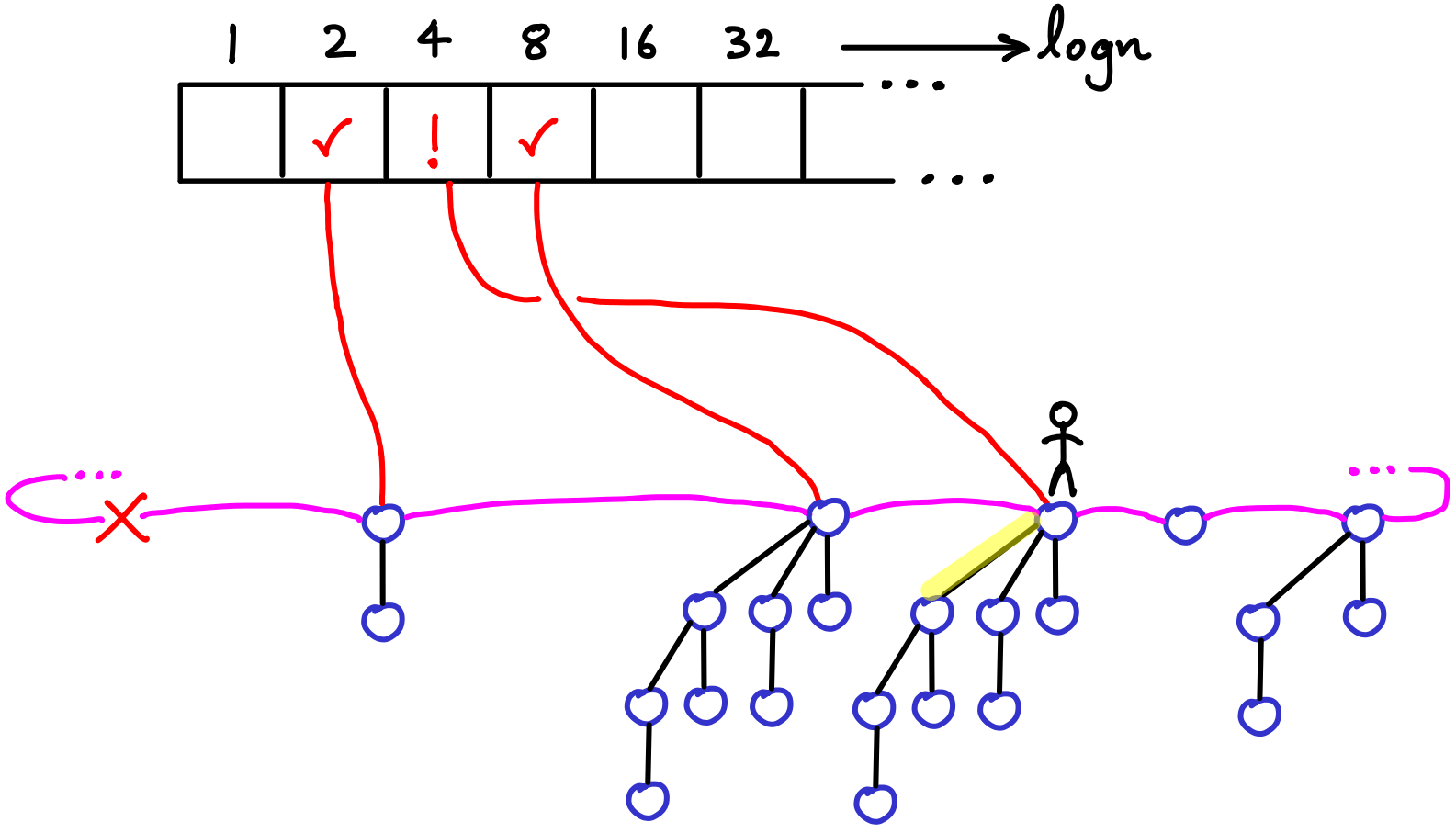
- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

4) Make sure ≤ 1 binomial tree of each size.



FIBONACCI HEAPS

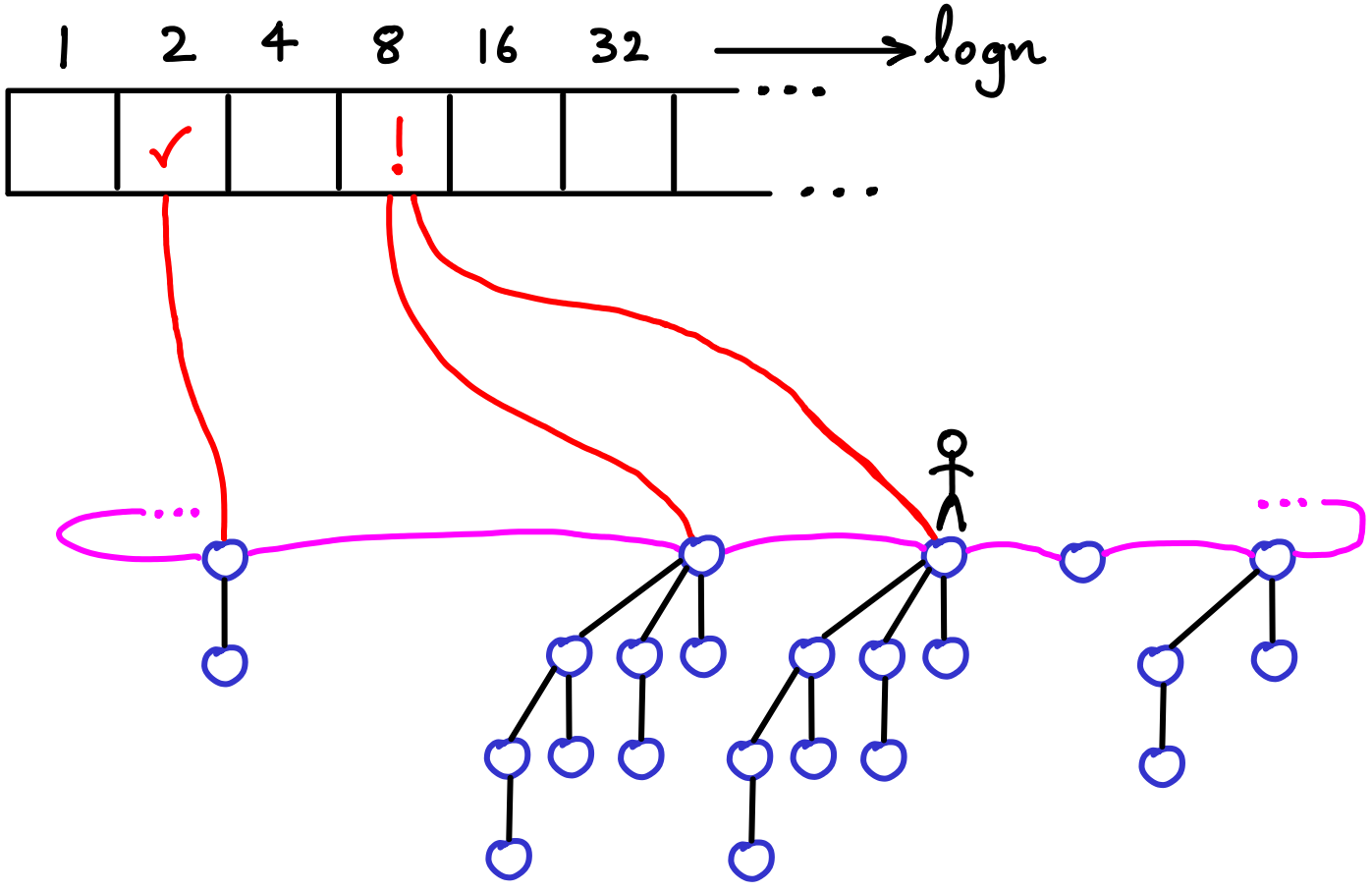
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

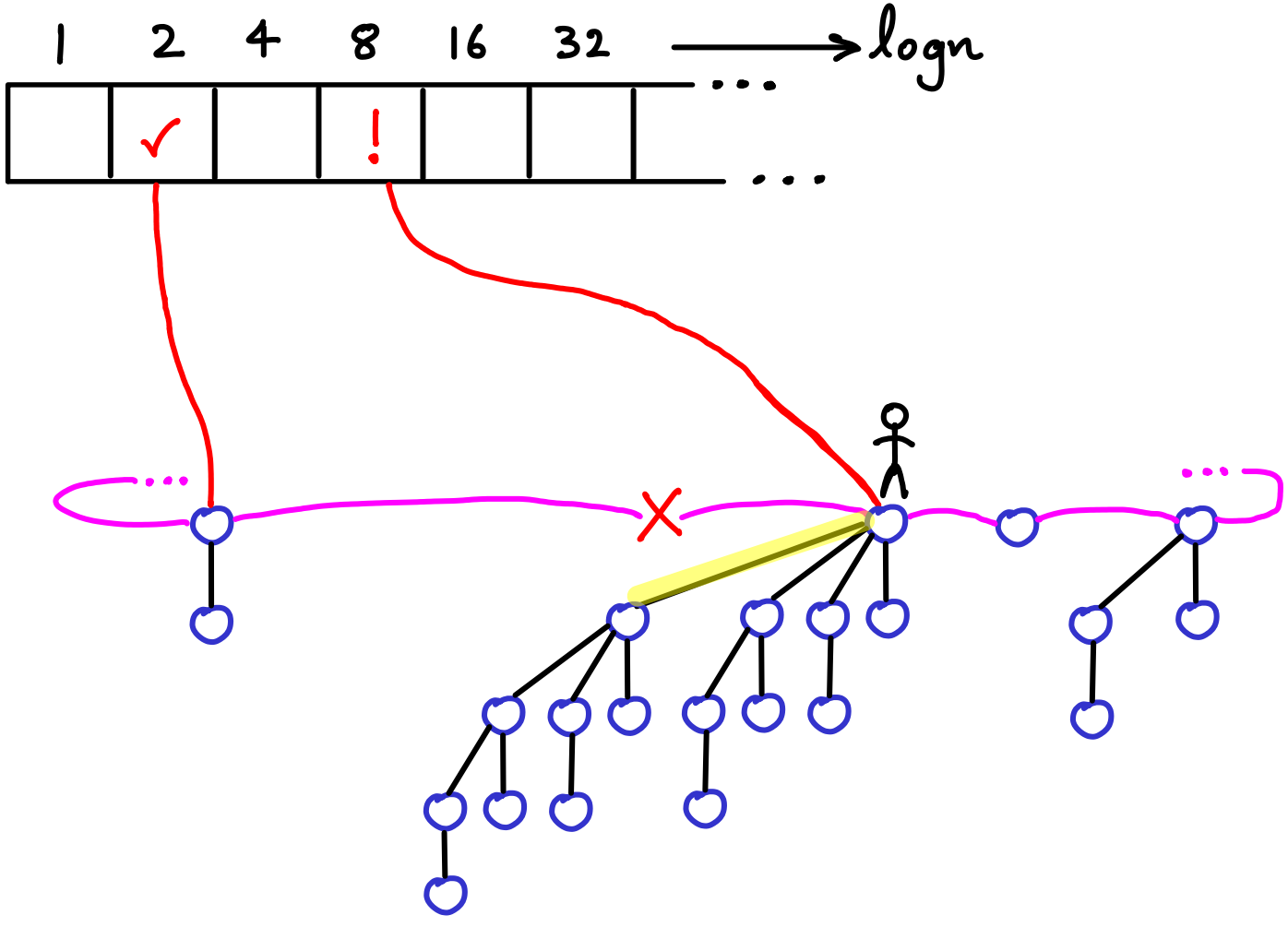
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

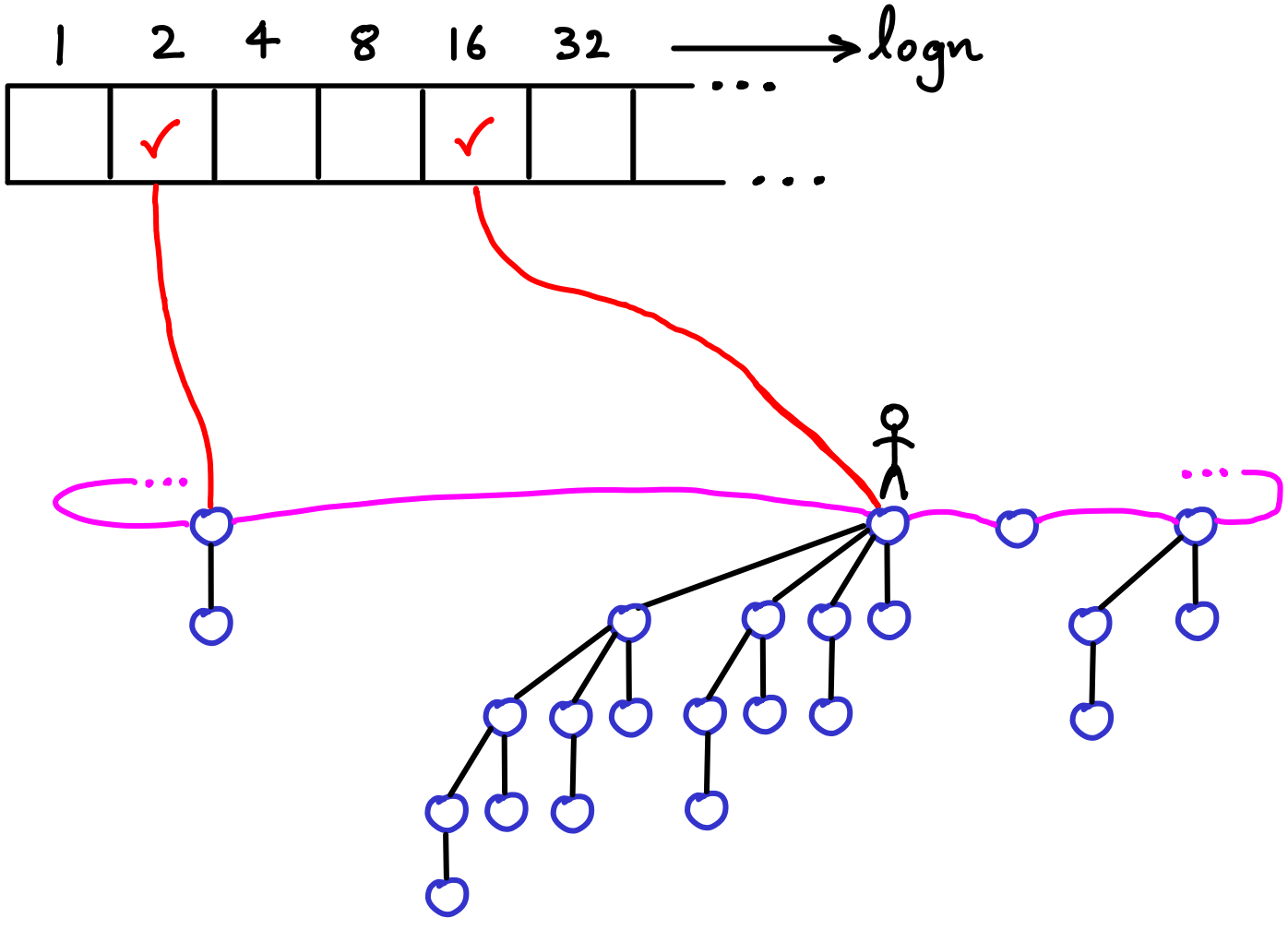
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(log n) am.

4) Make sure ≤1 binomial tree of each size.



FIBONACCI HEAPS

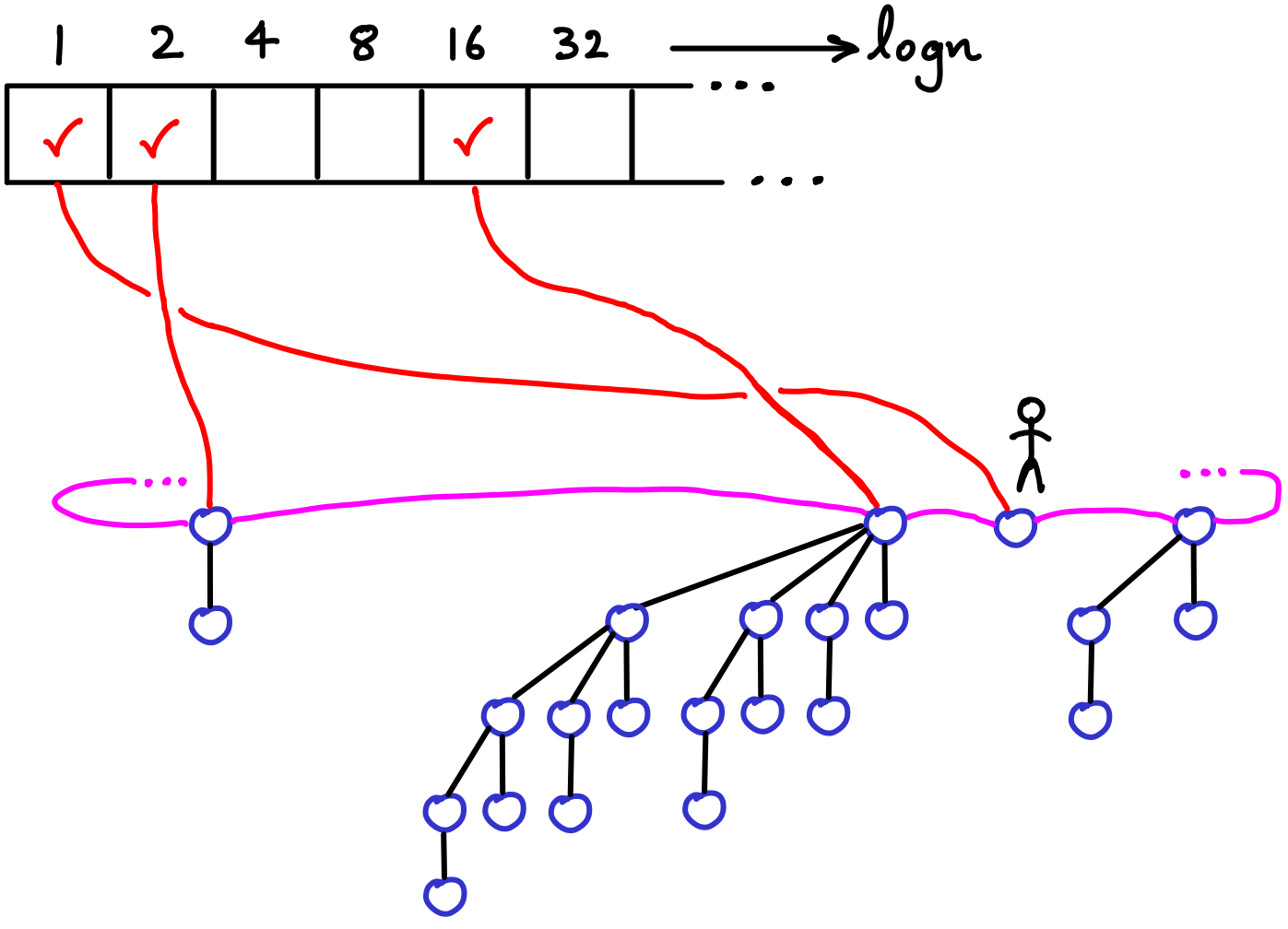
- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

4) Make sure ≤ 1 binomial tree of each size.



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

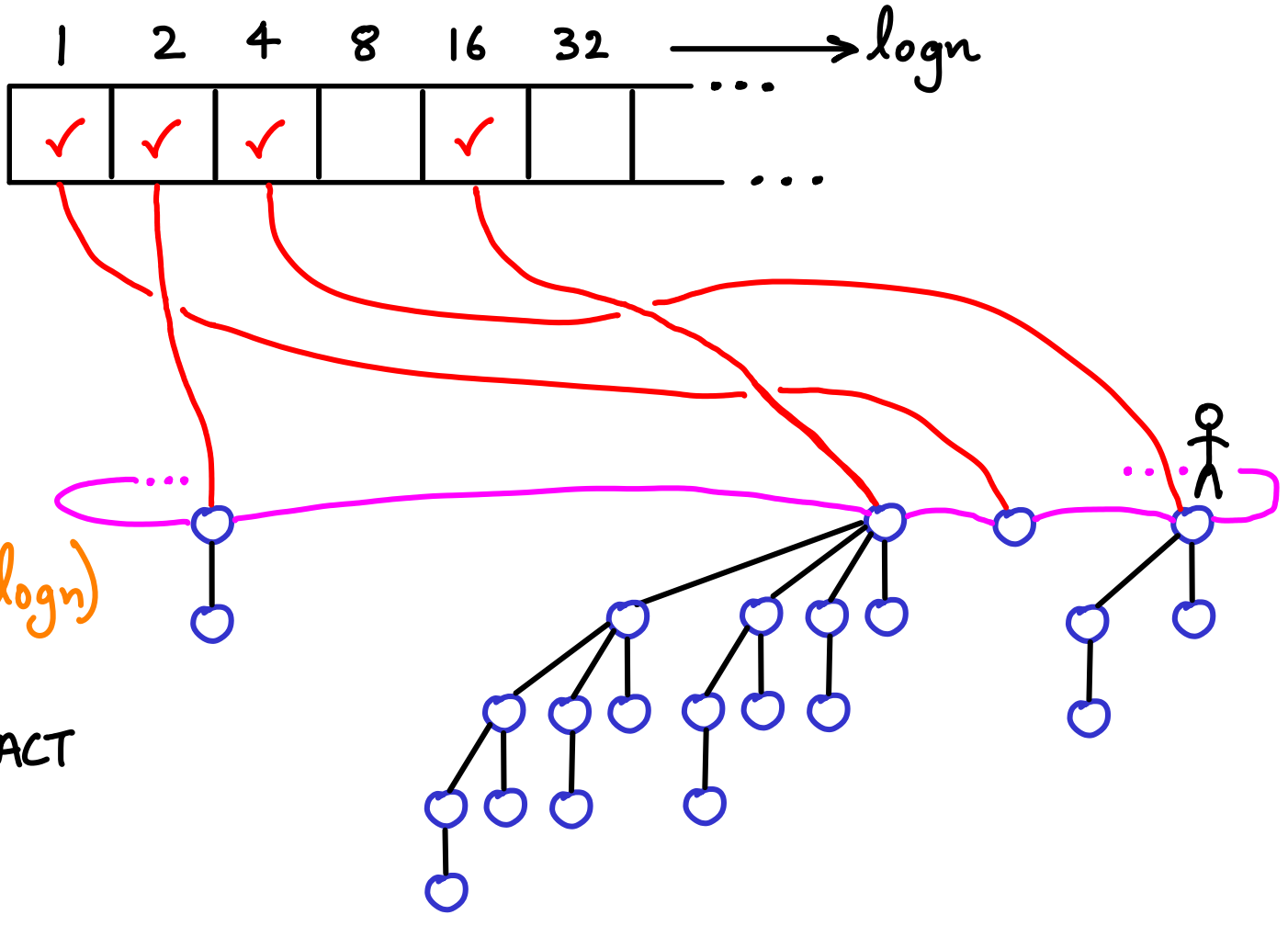
DECREASE ~~$O(n)$~~ $O(1)$ am.

EXTRACT MIN $O(n)$ $O(\log n)$ am.

Traversal: Θ(root list size + $\log n$)

before EXTRACT

4) Make sure ≤ 1 binomial tree of each size



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

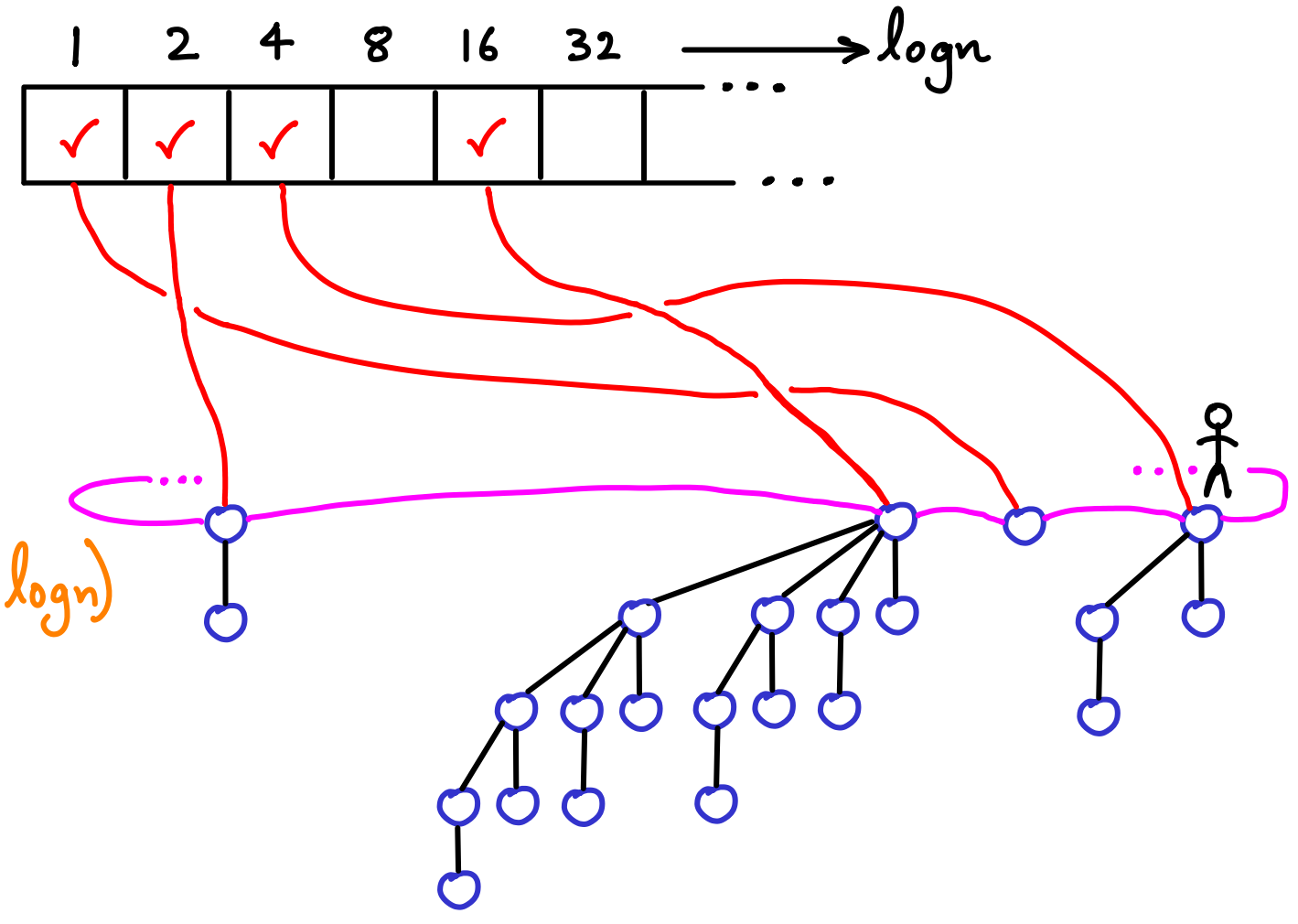
EXTRACT MIN O(n)
O(log n) am.

Traversal: Θ(root list size + log n)

Conflict

└ link 2 trees, O(1)

4) Make sure ≤1 binomial tree of each size



FIBONACCI HEAPS

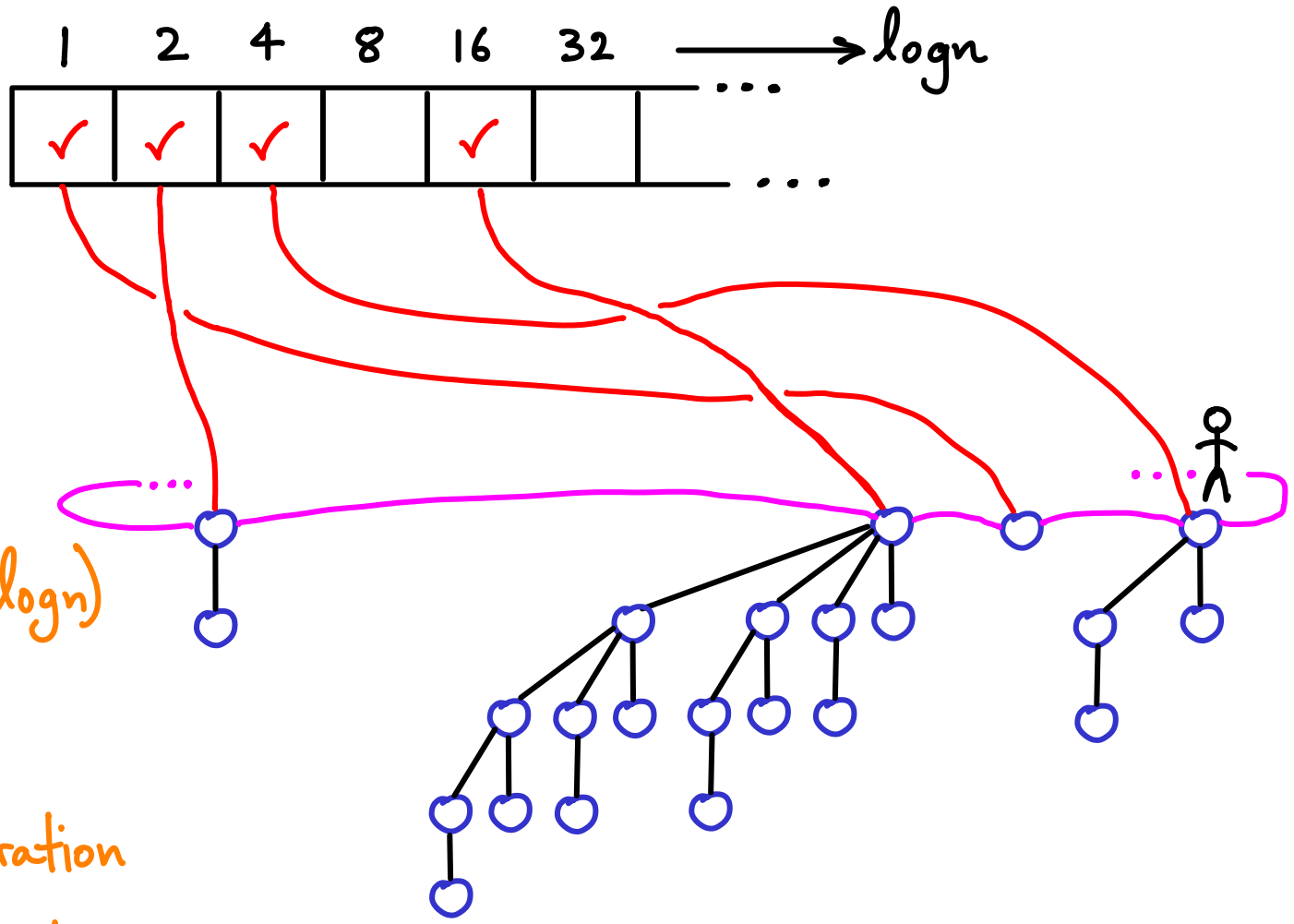
- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

4) Make sure ≤1 binomial tree of each size



Traversal: Θ(root list size + logn)

Conflict
- link 2 trees, O(1) per iteration
- promotions: O(logn) per iteration

FIBONACCI HEAPS

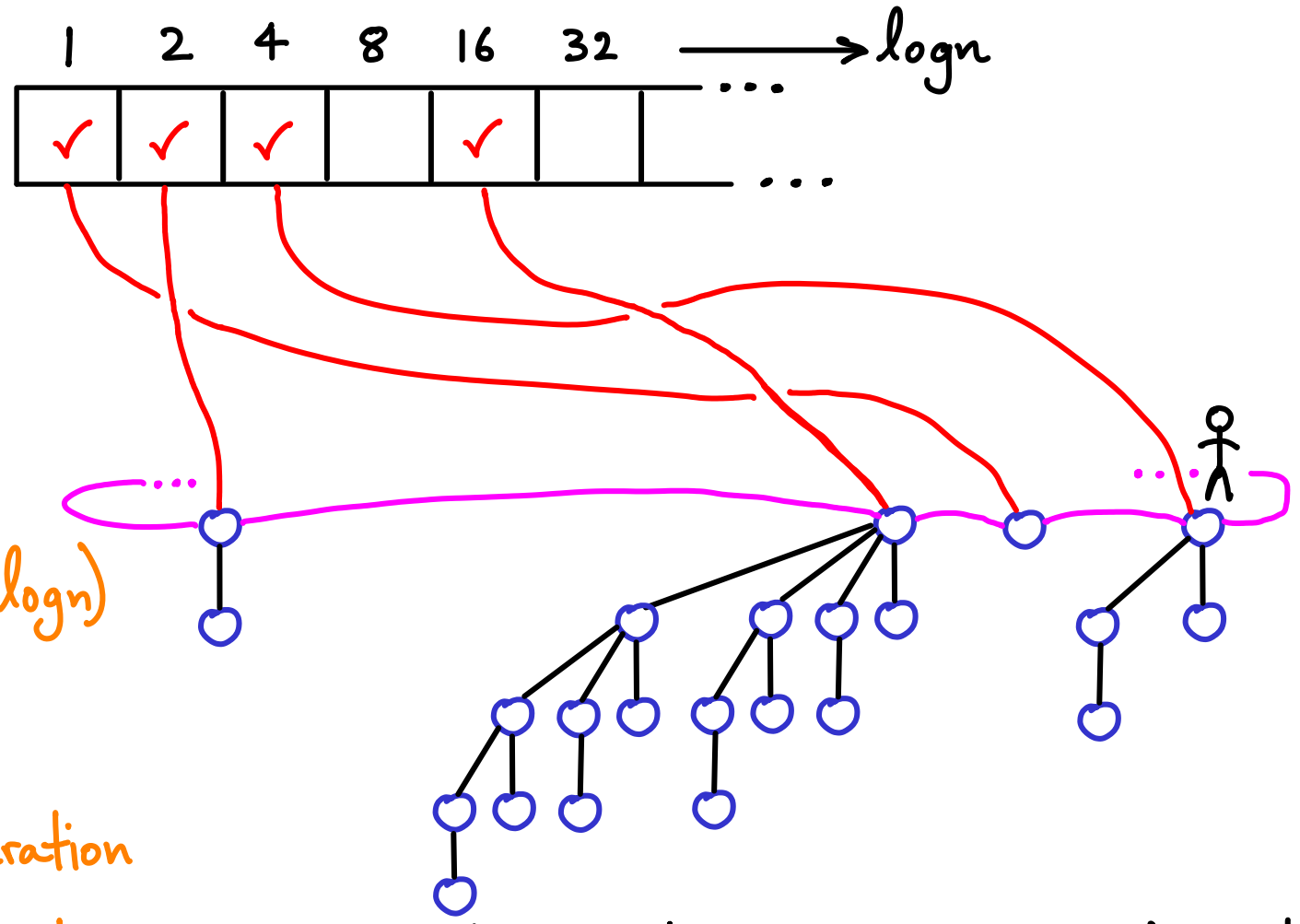
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(logn) am.

4) Make sure ≤1 binomial tree of each size



Traversal: Θ(root list size + logn)

Conflict

- link 2 trees, O(1) per iteration
- promotions: O(logn) per iteration

* but each promotion removes one element

FIBONACCI HEAPS

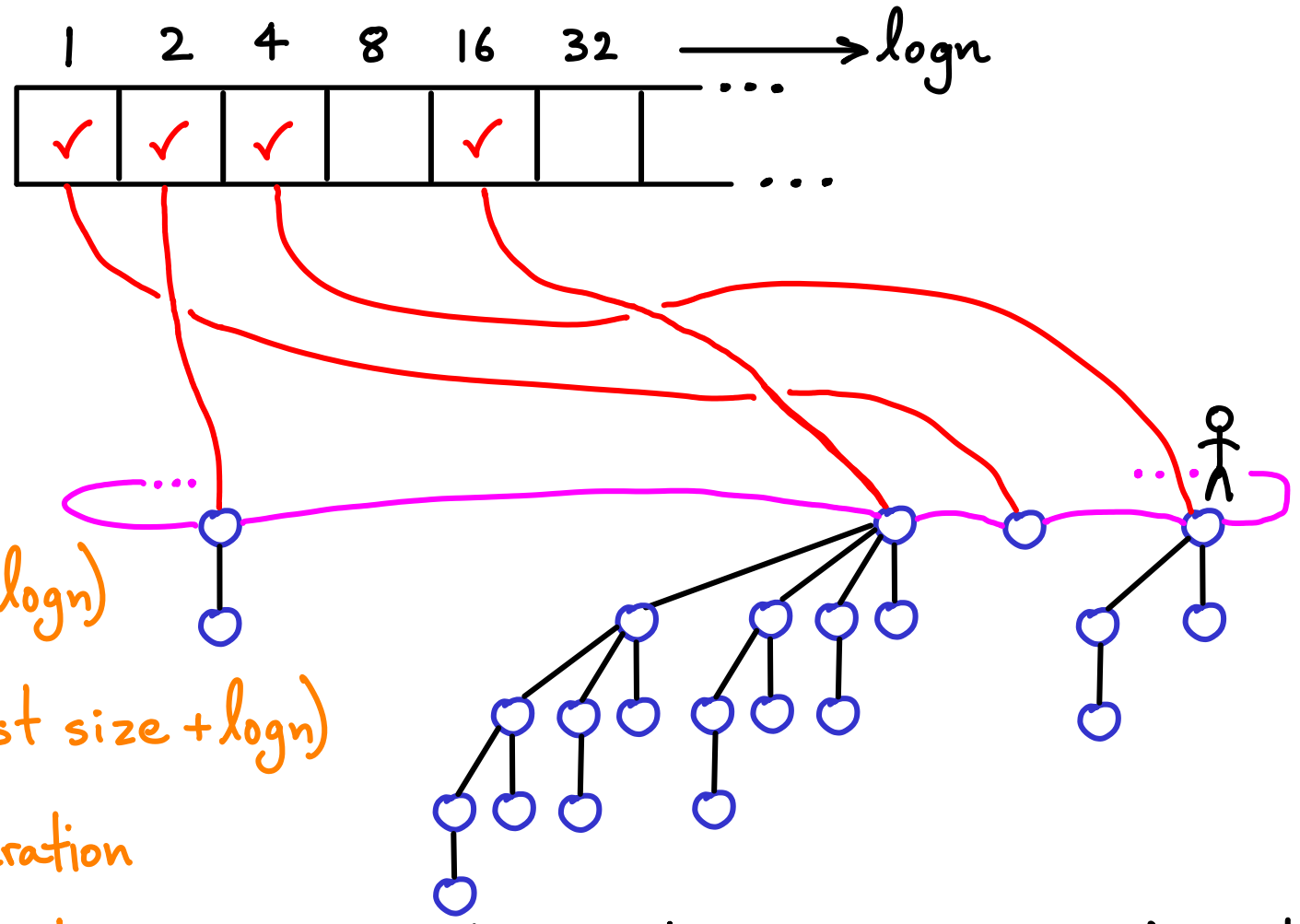
- MIN
- INSERT
- MERGE

} O(1)

DECREASE ~~O(n)~~ O(1) am.

EXTRACT MIN O(n) O(logn) am.

4) Make sure ≤1 binomial tree of each size



Traversal: Θ(root list size + logn)

Conflict *overall: Θ(root list size + logn)

- link 2 trees, O(1) per iteration
- promotions: O(logn) per iteration

*but each promotion removes one element

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$ / $O(1)$ am.

EXTRACT MIN $O(n)$ / $O(\log n)$ am.

EXTRACT MIN:

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

EXTRACT MIN : $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

EXTRACT MIN : $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

root list size → starts = 0, gets reset to $O(\log n)$ after EXTRACT MIN

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

EXTRACT MIN : $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

root list size → starts = 0, gets reset to $O(\log n)$ after EXTRACT MIN
→ +1 w/ INSERT

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

root list size → starts = 0, gets reset to $O(\log n)$ after EXTRACT MIN

→ +1 w/ INSERT: pay extra for step 4 of EXTRACT MIN

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

root list size → starts = 0, gets reset to $O(\log n)$ after EXTRACT MIN
- → +1 w/ INSERT: pay extra for step 4 of EXTRACT MIN
- → +0 w/ MERGE: sum of root list sizes doesn't change

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN : $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

root list size →
- starts = 0, gets reset to $O(\log n)$ after EXTRACT MIN
- +1 w/ INSERT: pay extra for step 4 of EXTRACT MIN
- +0 w/ MERGE: sum of root list sizes doesn't change

EXTRACT MIN $O(\log n)$ amortized

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN : $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

Φ_i = root list size = r_i

$\Phi_0 = 0 \quad \Phi_i \geq 0$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

$\Phi_i = \text{root list size} = r_i$

$\Phi_0 = 0 \quad \Phi_i \geq 0$

	c
MIN	$O(1)$
INSERT	$O(1)$
MERGE	$O(1)$
EXTRACT	$O(r_i + \log n)$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

$\Phi_i = \text{root list size} = r_i$

$\Phi_0 = 0 \quad \Phi_i \geq 0$

	c	$\Delta\Phi$
MIN	$O(1)$	0
INSERT	$O(1)$	+1
MERGE	$O(1)$	0
EXTRACT	$O(r_i + \log n)$	

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

Φ_i = root list size = r_i

$\Phi_0 = 0 \quad \Phi_i \geq 0$

	c	$\Delta\Phi$
MIN	$O(1)$	0
INSERT	$O(1)$	+1
MERGE	$O(1)$	0
EXTRACT	$O(r_i + \log n)$	$\leq -r_i + \log n$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$ ✓

DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am. ✓

EXTRACT MIN: $O(\log n + \text{root list size})$

1) Delete MIN node
2) Set children's parent pointers to NULL $O(\log n)$
3) Link root lists: $O(1)$
4) Make sure ≤ 1 binomial tree of each size (& update MIN) $\Theta(\log n + \text{root list size})$

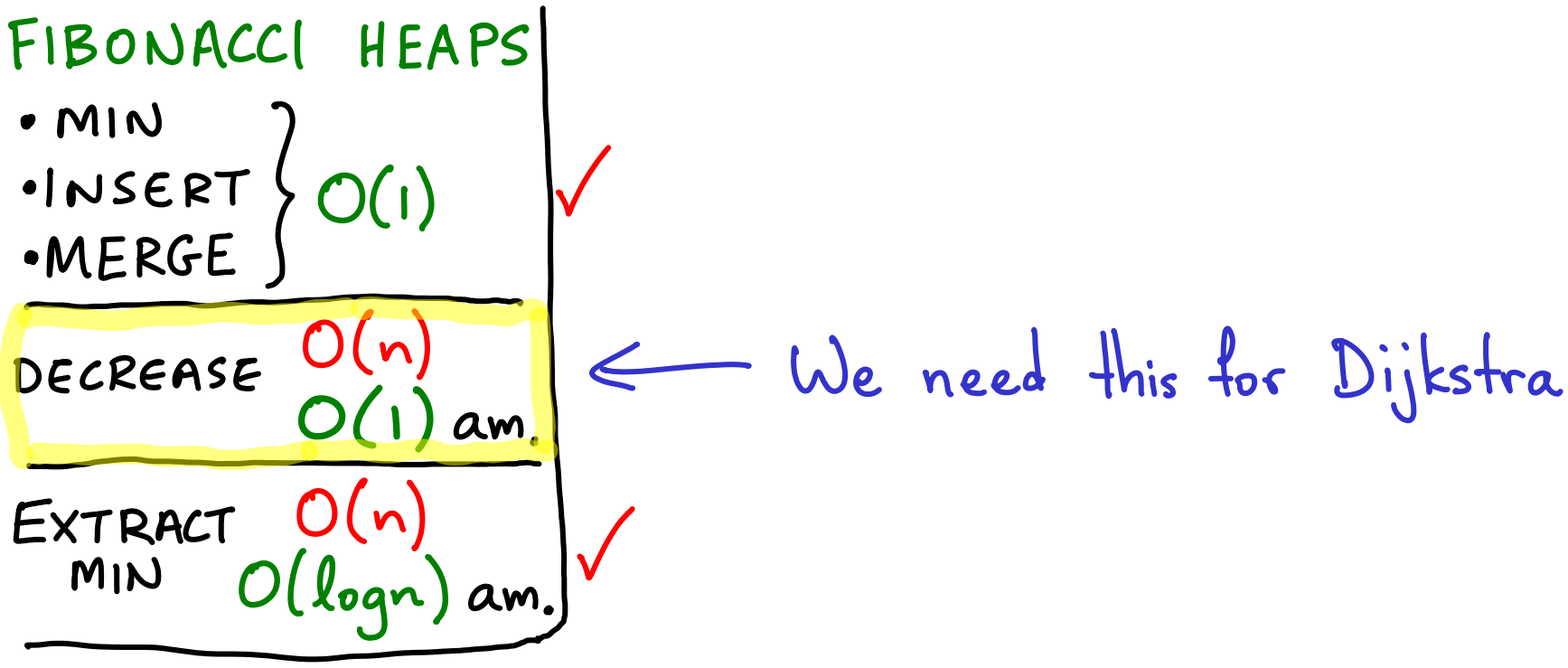
$\Phi_i = \text{root list size} = r_i$

$\Phi_0 = 0 \quad \Phi_i \geq 0$

	c	$\Delta\Phi$	$\hat{c}$
MIN	$O(1)$	0	$O(1)$
INSERT	$O(1)$	$+1$	$O(1)$
MERGE	$O(1)$	0	$O(1)$
EXTRACT	$O(r_i + \log n)$	$\leq -r_i + \log n$	$O(\log n)$

HEAPS:

	REGULAR	BINOMIAL	halfway there
REPORT MIN:	O(1)	←	✓ ←
EXTRACT MIN:	O(logn)	←	✓ O(logn) amortized O(n)
INSERT:	O(logn) O(1) amortized	←	✓ O(1)
DECREASE KEY:	O(logn)	←	● O(1) amortized O(n)
MERGE/UNION:	O(n)	O(logn)	✓ O(1)



We need this for Dijkstra

FIBONACCI HEAPS

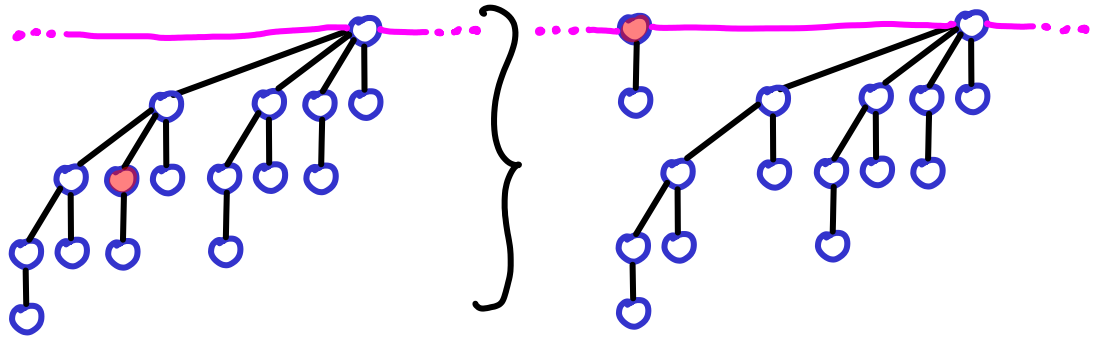
- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

→ 1) Move key's subtree to root list



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

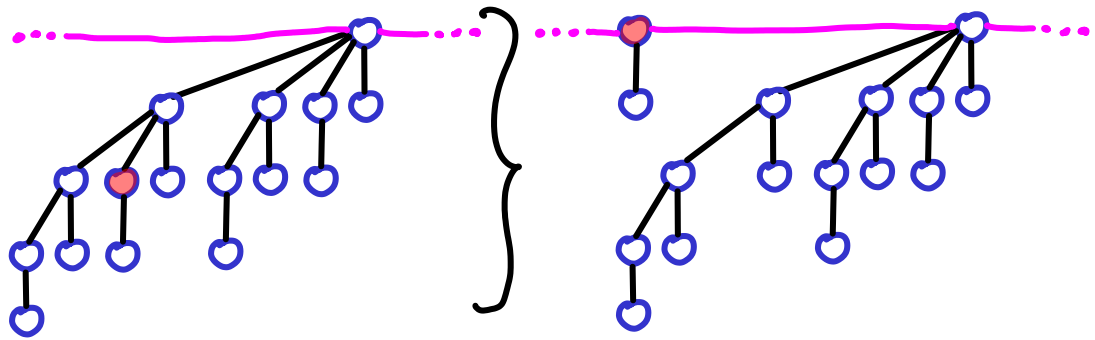
} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

→ 1) Move key's subtree to root list

2) Hope nothing else happens



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

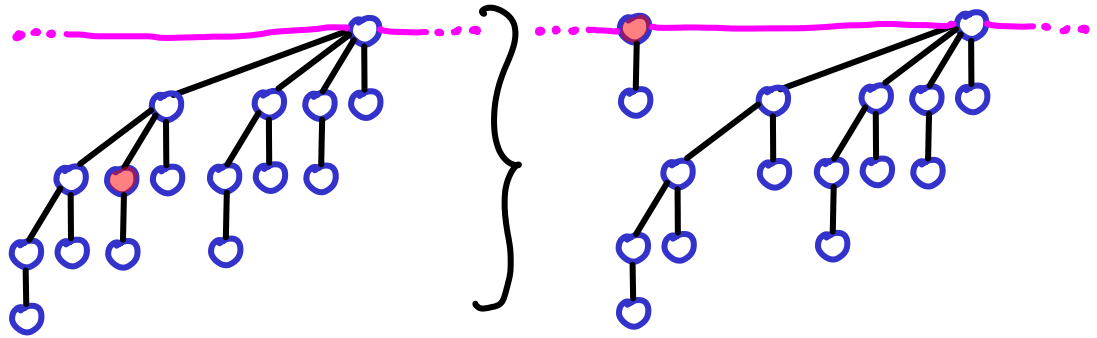
EXTRACT MIN O(n)
O(logn) am.

1) Move key's subtree to root list
2) Hope nothing else happens

Binomial tree structure degraded.

↳ was important for EXTRACT MIN

Θ(logn + root list size)
Θ(deg(MIN) + root list size)
...



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

1) Move key's subtree to root list
2) Hope nothing else happens

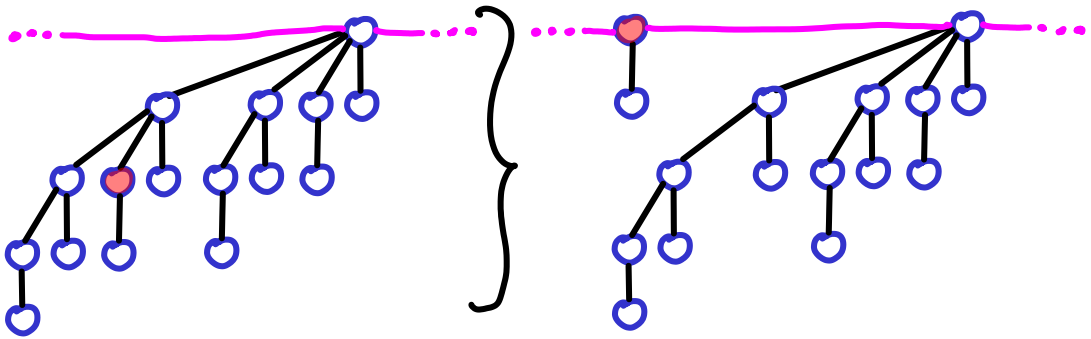
Binomial tree structure degraded.

↳ was important for EXTRACT MIN

↳ $\Theta(\log n + \text{root list size})$

↳ $\Theta(\deg(\text{MIN}) + \text{root list size})$

lost property:
O(logn) degree values



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

→ 1) Move key's subtree to root list

2) Hope nothing else happens

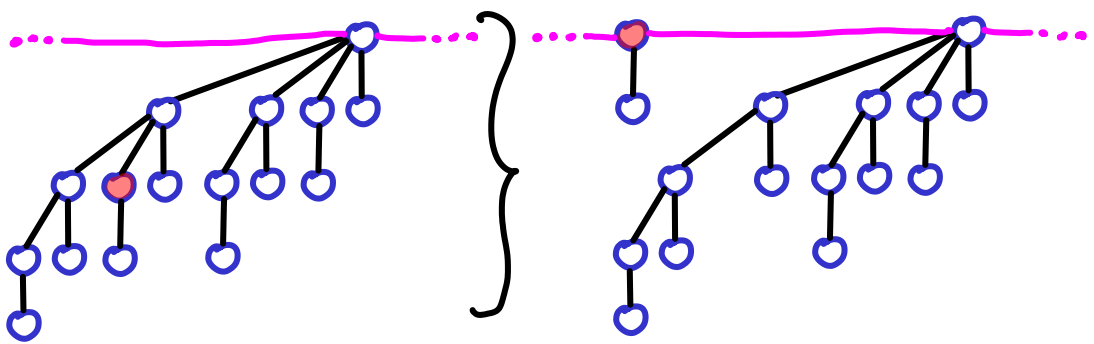
Binomial tree structure degraded.

↳ was important for EXTRACT MIN

↳ $\Theta(\log n + \text{root list size})$

↳ $\Theta(\deg(\text{MIN}) + \text{root list size})$

lost property:
O(log n) degree values



If no (more) EXTRACT MIN, we're ok.

| but then we would never make binomial trees... just use linked list

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

1) Move key's subtree to root list

2) ~~Hope nothing else happens~~

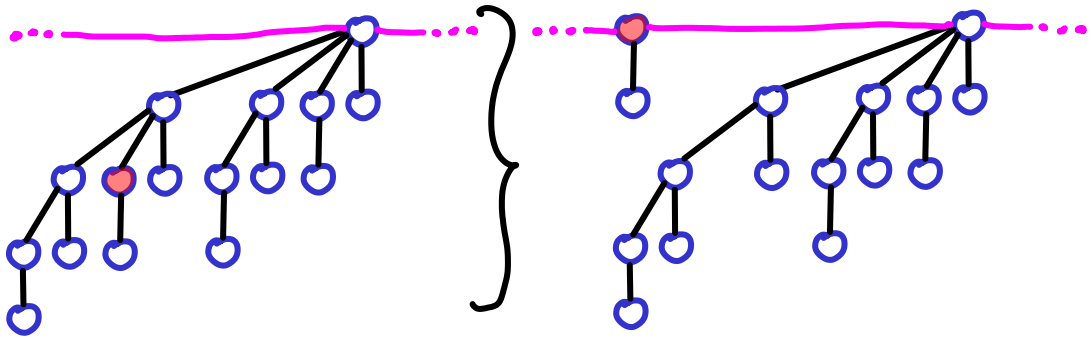
Binomial tree structure degraded.

↳ was important for EXTRACT MIN

↳ $\Theta(\log n + \text{root list size})$

↳ $\Theta(\deg(\text{MIN}) + \text{root list size})$

lost property:
O(logn) degree values



Need to make sure $\deg(\text{MIN}) = O(\log n)$

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

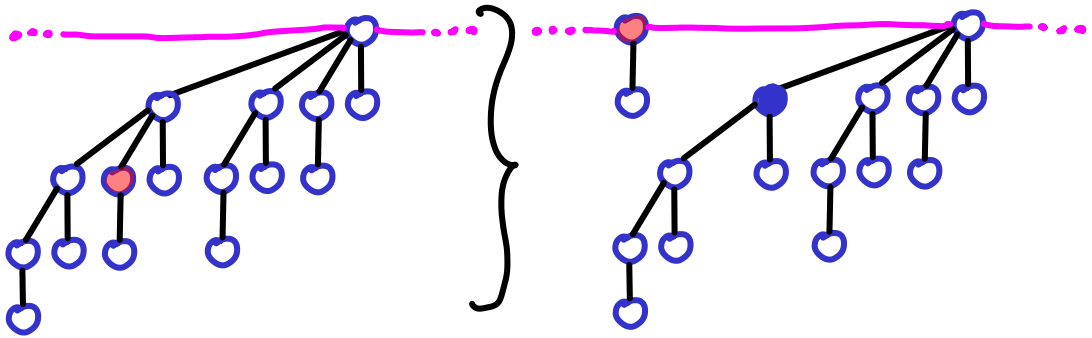
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.

↳ if parent is unmarked, mark it •



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

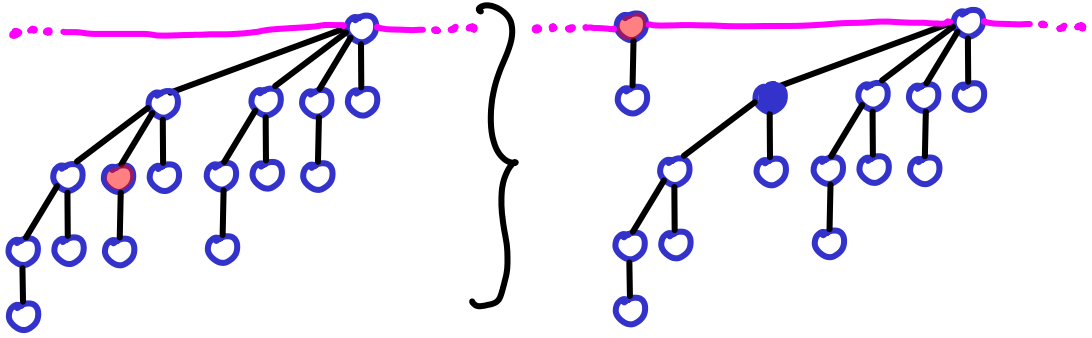
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(logn) am.

1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.

↳ if parent is unmarked, mark it ●
else move parent's subtree to root (recurse)



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

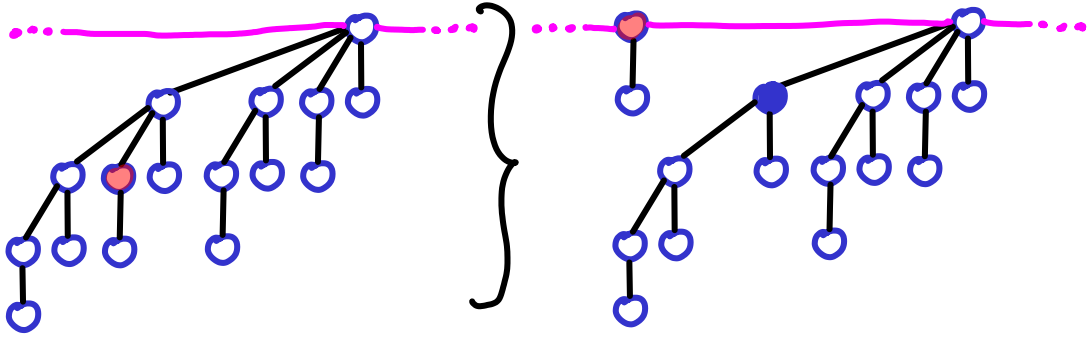
} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.
   - if parent is unmarked, mark it •
     else move parent's subtree to root (recurse)
   - unmark node



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

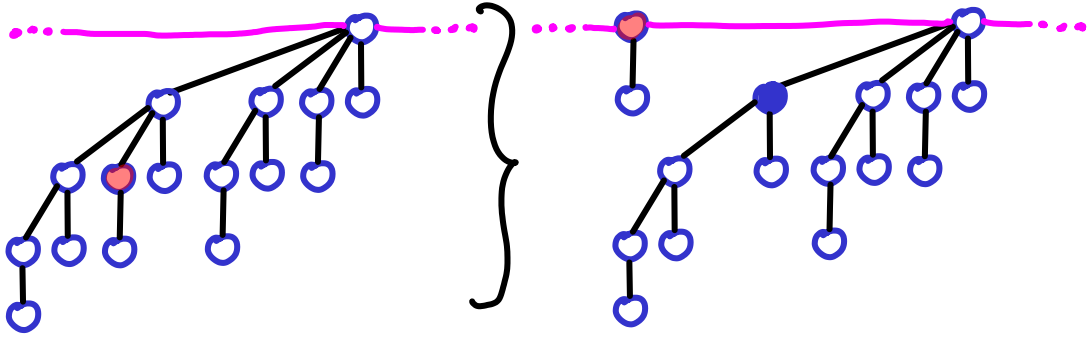
EXTRACT MIN O(n)
O(logn) am.

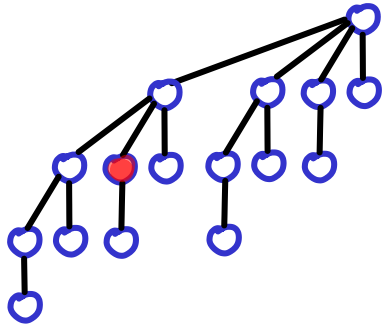
1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.

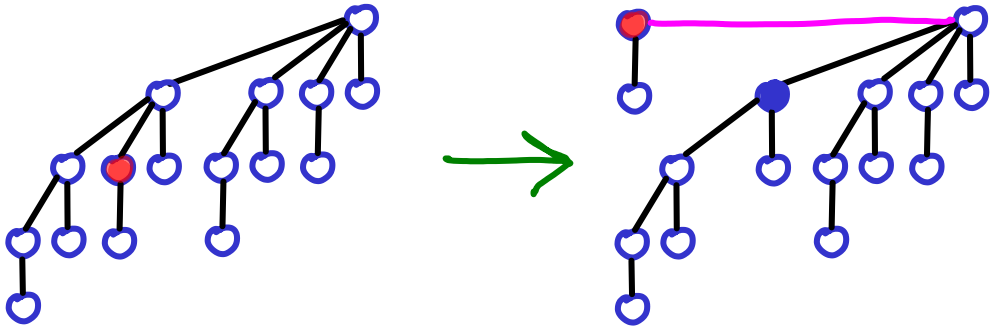
- if parent is unmarked, mark it •
- else * move parent's subtree to root (recurse)
- unmark node

(* parent just lost 2nd child since being unmarked)



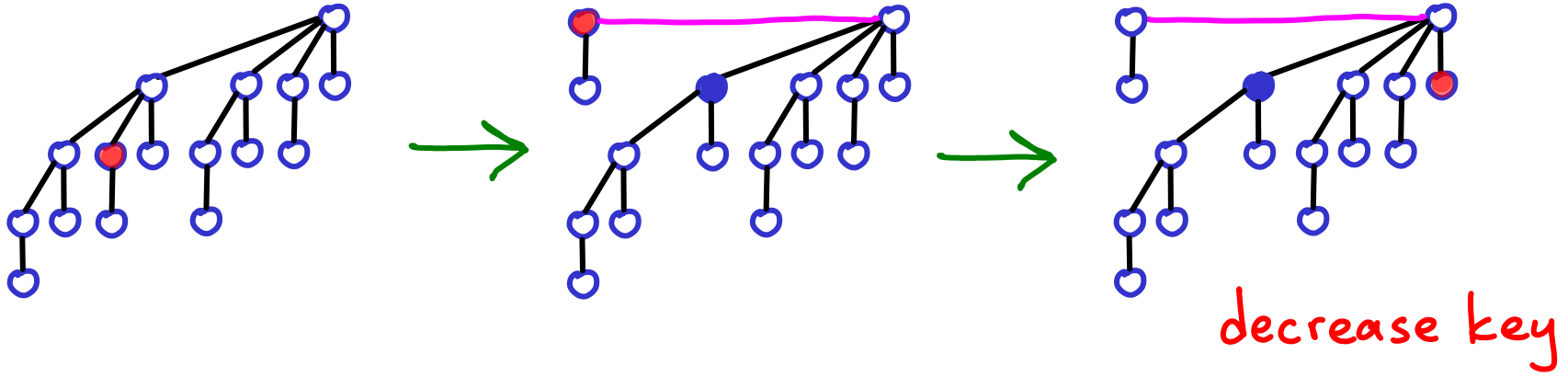


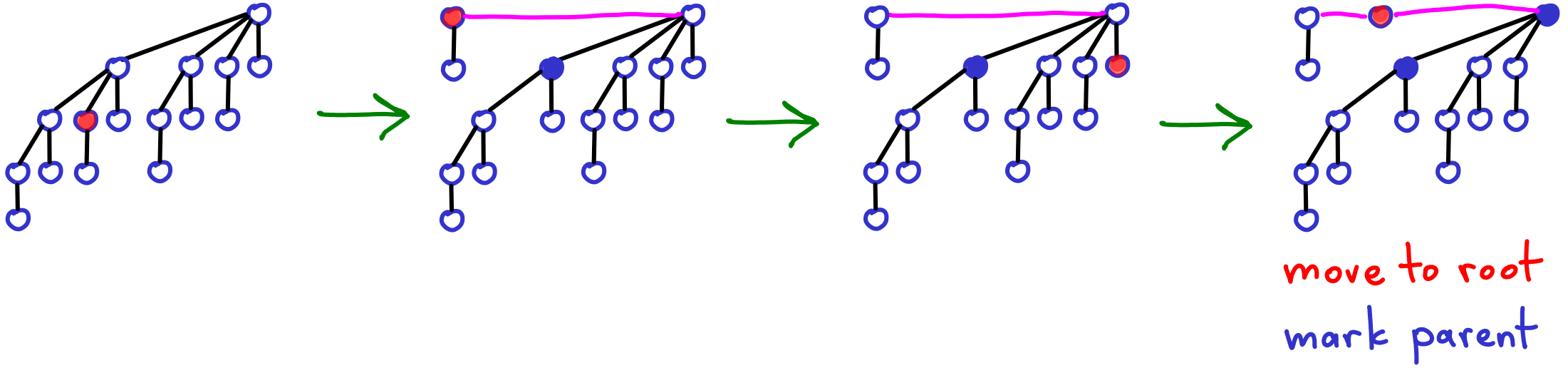
decrease key

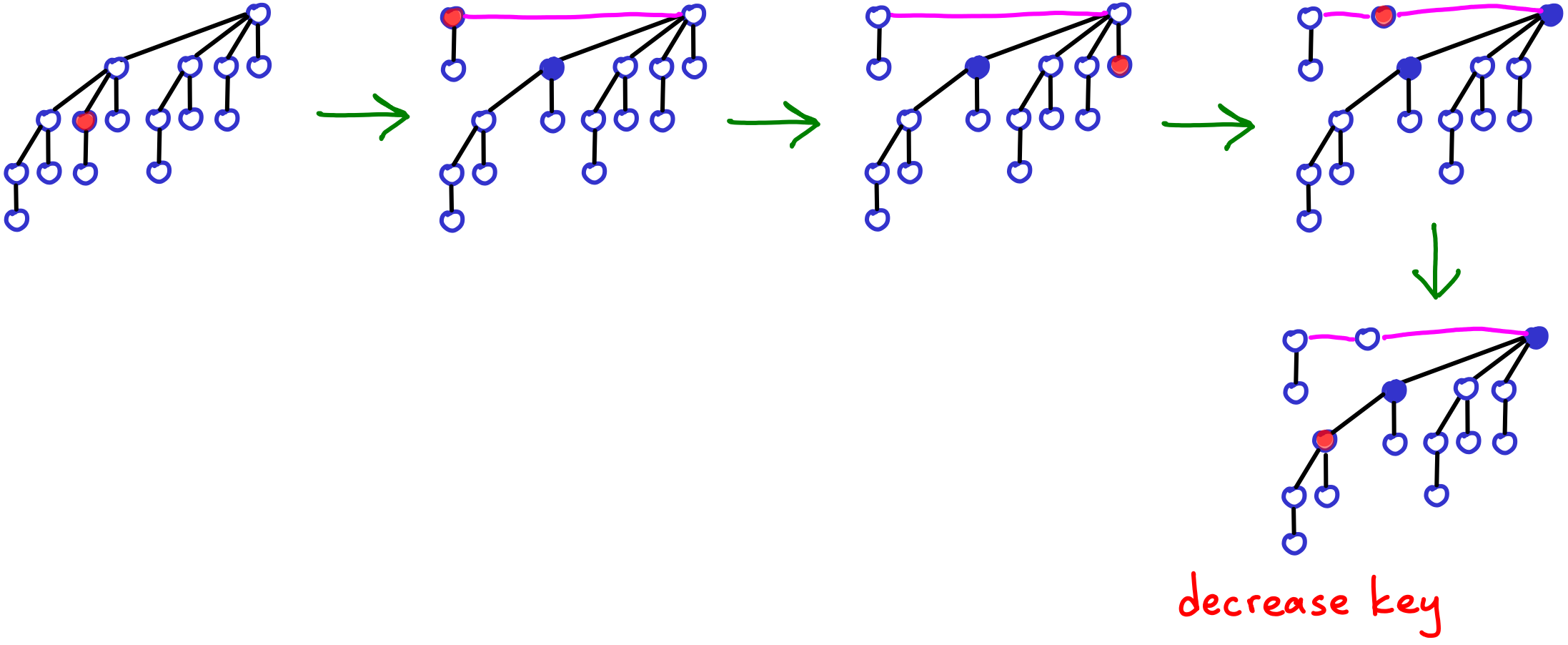


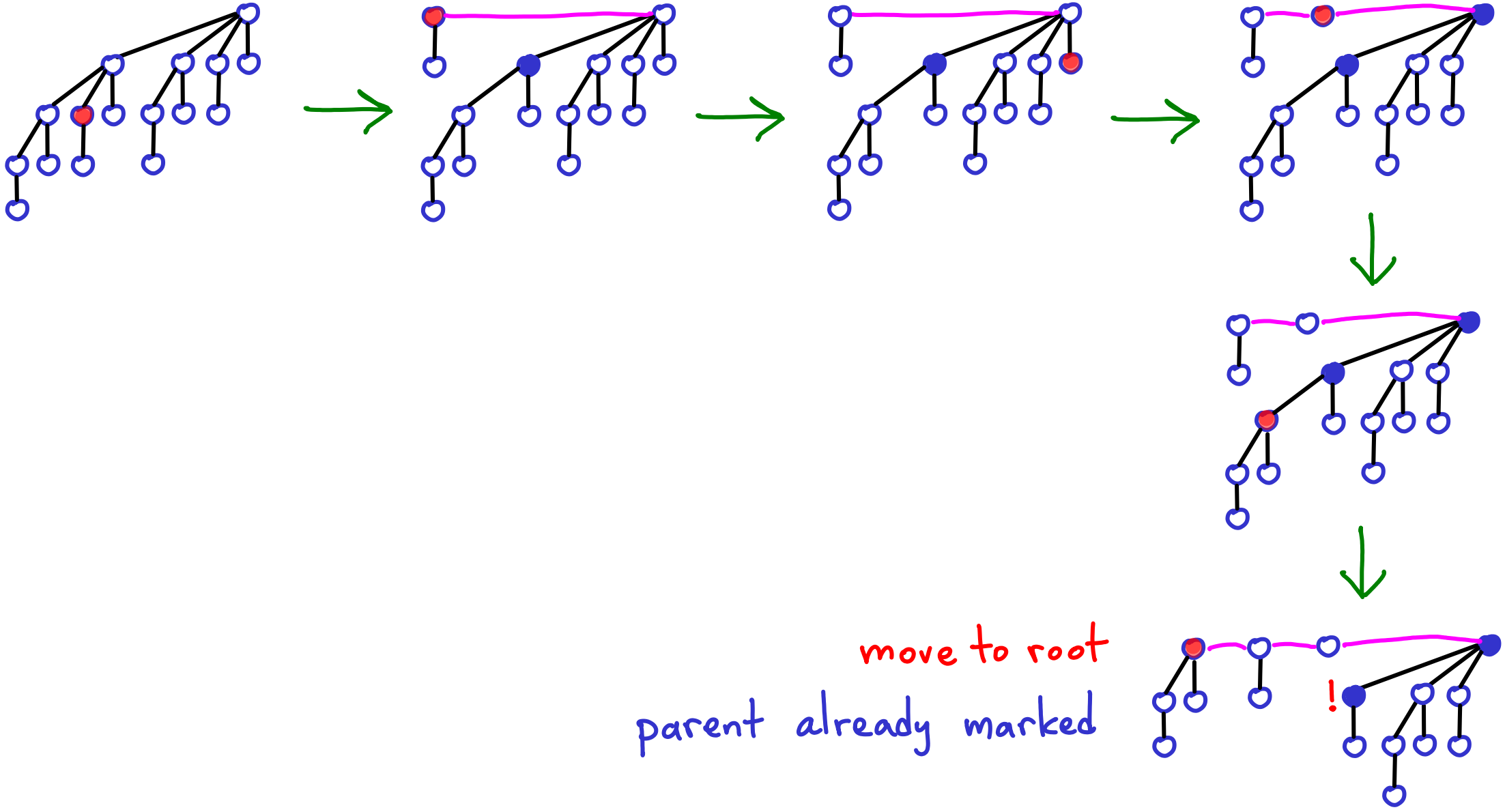
move to root

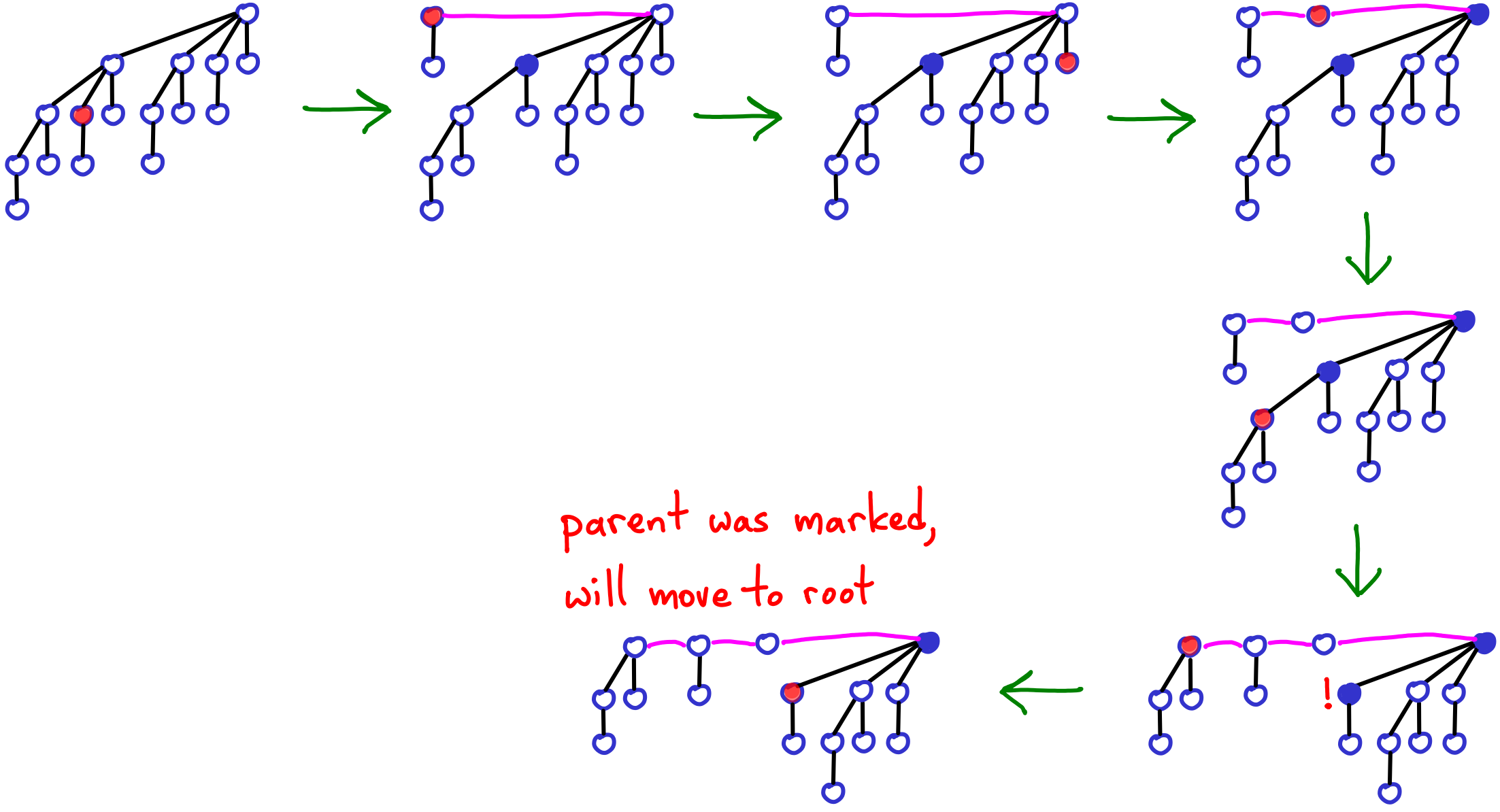
mark parent

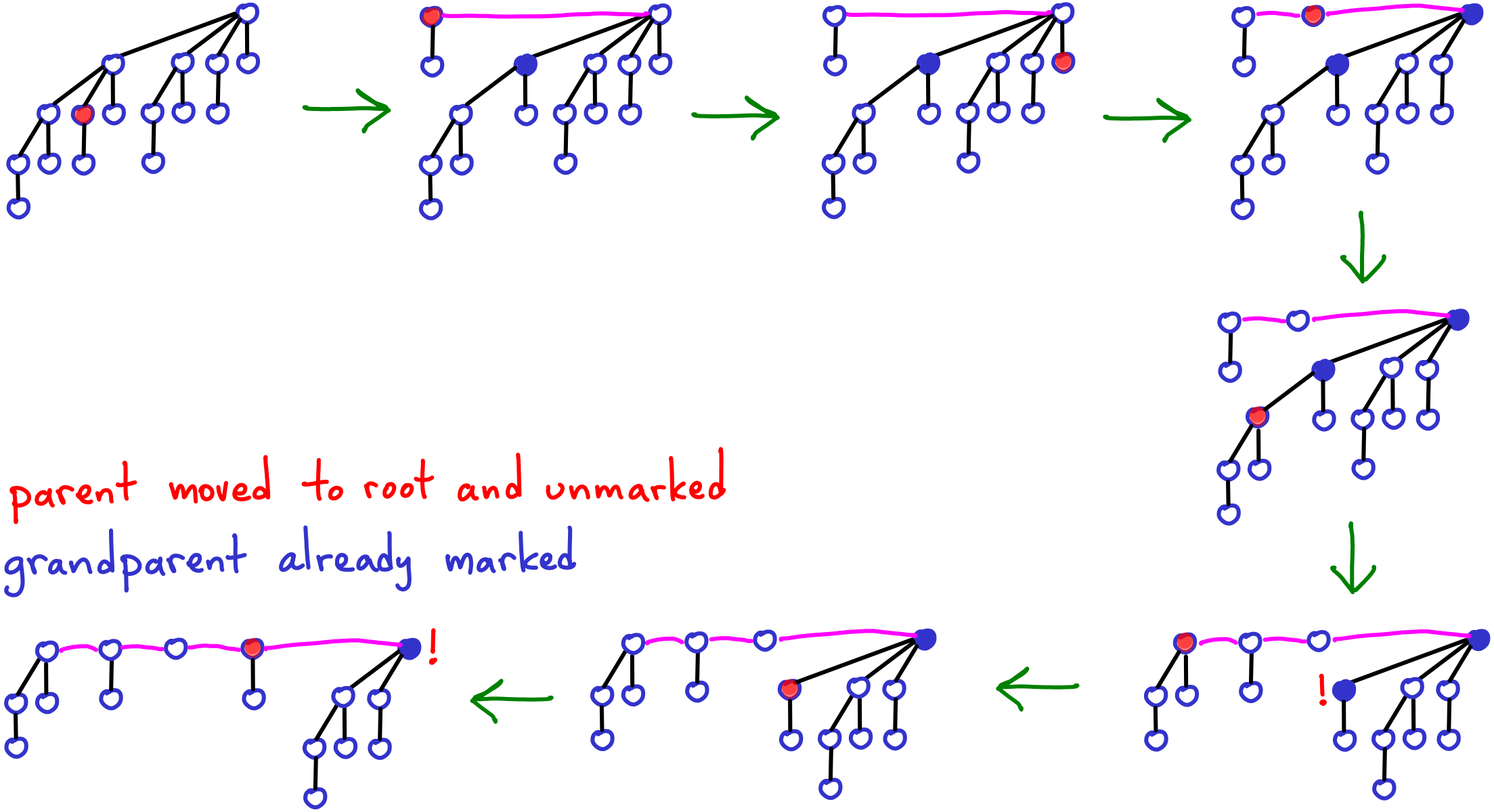

decrease key

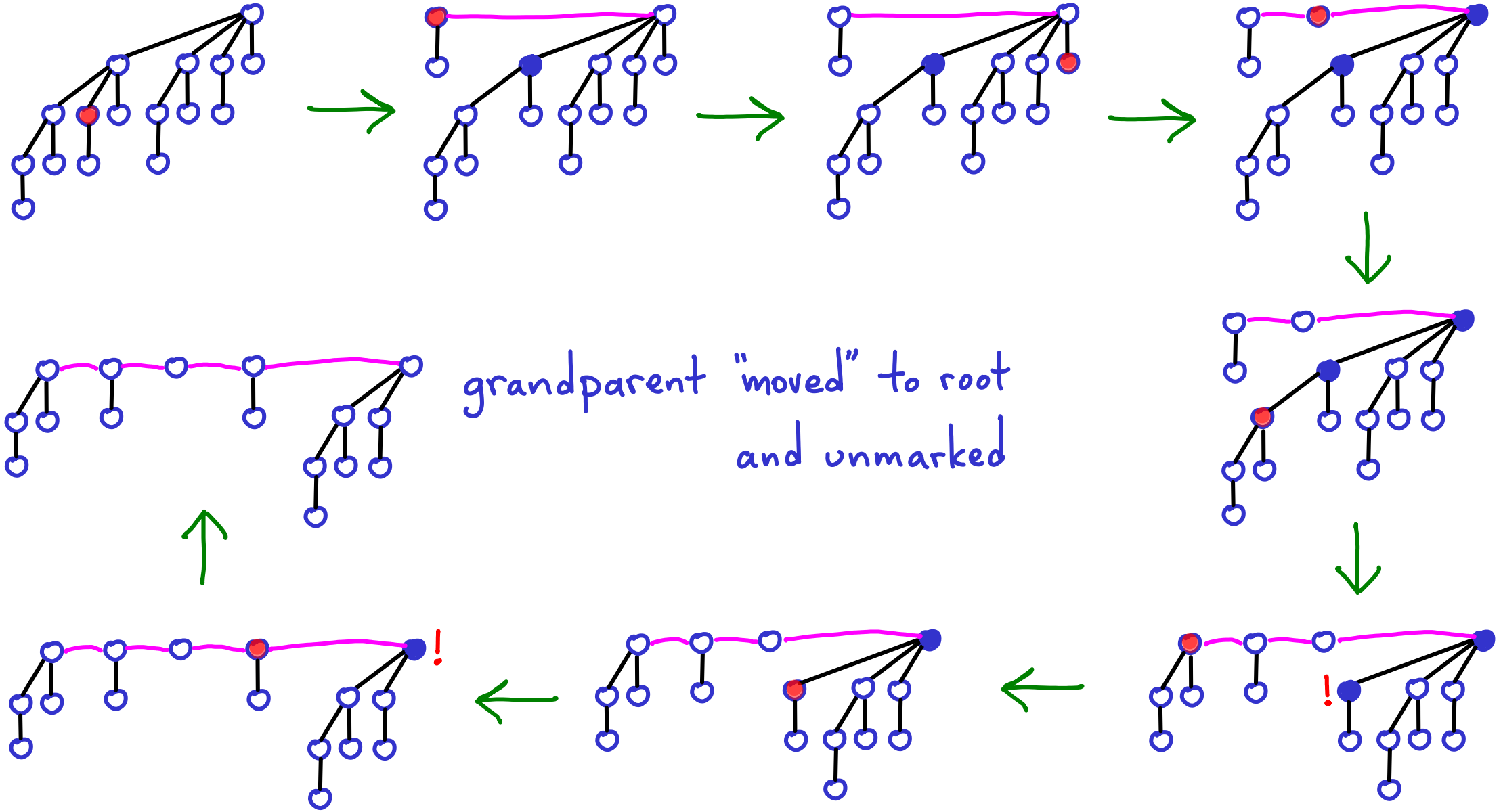

move to root
mark parent


decrease key


move to root
parent already marked
!


parent was marked,
will move to root
!


parent moved to root and unmarked
grandparent already marked
!
!


grandparent "moved" to root
and unmarked
!
!

FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

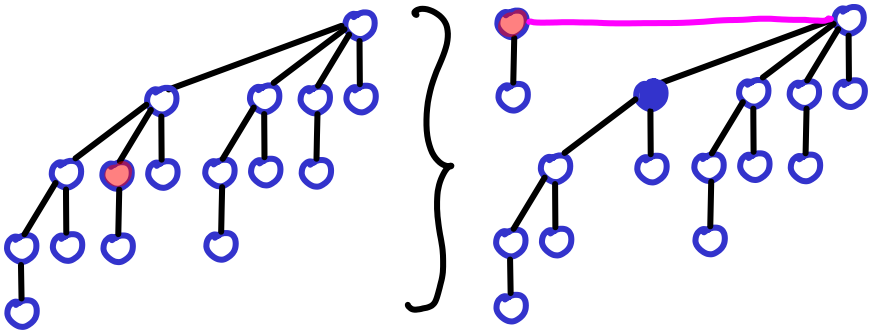
DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.
   - if parent is unmarked, mark it •
     else move parent's subtree to root (recurse)
   - unmark node

Cost: $\Theta(1 + \#\text{consecutive marked ancestors})$



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

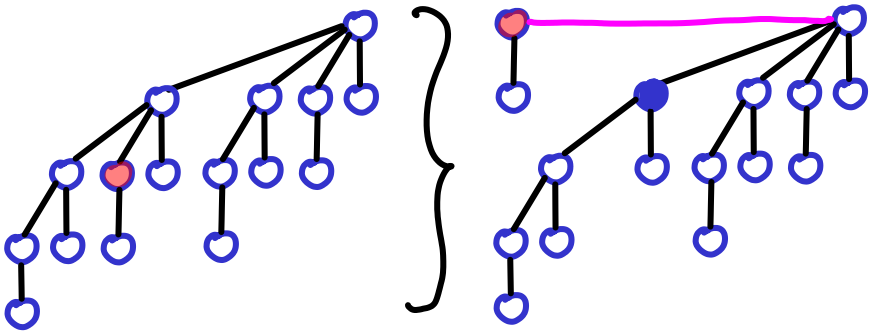
EXTRACT MIN O(n)
O(log n) am.

1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.
   - if parent is unmarked, mark it •
     else move parent's subtree to root (recurse)
   - unmark node

Cost: $\Theta(1 + \#\text{consecutive marked ancestors})$

Amortize: $\Phi = \#\text{marked nodes} \rightarrow \hat{c} = O(1)$



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} O(1)

DECREASE O(n)
O(1) am.

EXTRACT MIN O(n)
O(log n) am.

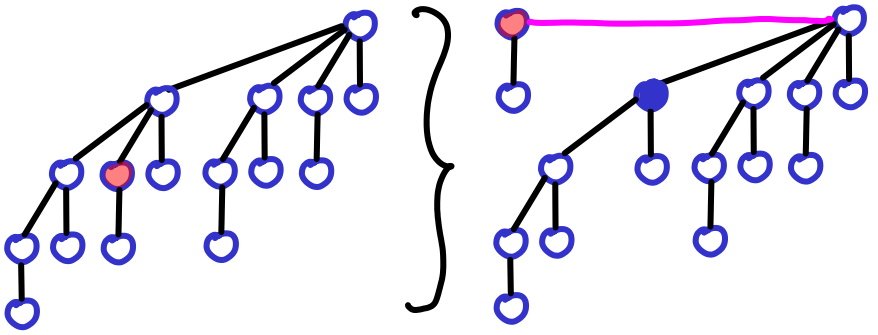
1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.
   - if parent is unmarked, mark it •
     else move parent's subtree to root (recurse)
   - unmark node

Cost: $\Theta(1 + \#\text{consecutive marked ancestors})$

Amortize: $\Phi = \#\text{marked nodes} \rightarrow \hat{c} = O(1)$

...but we are affecting Φ(EXTRACT MIN)
(making new roots)



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

DECREASE	$O(n)$ $O(1)$ am.
EXTRACT MIN	$O(n)$ $O(\log n)$ am.

→ 1) Move key's subtree to root list

2) When a node moves to root during DECREASE op.

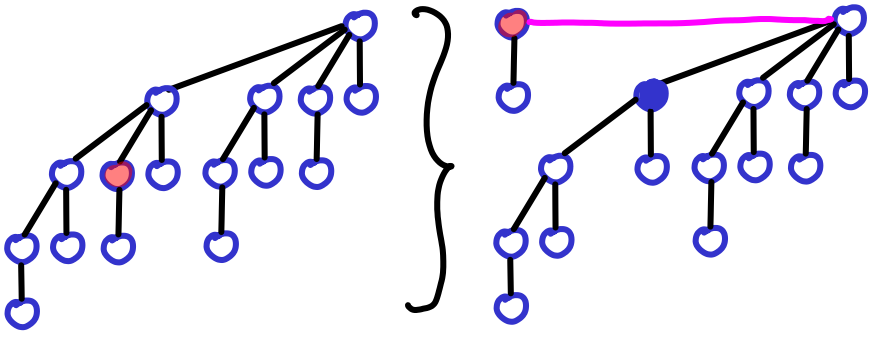
- if parent is unmarked, mark it •
  else move parent's subtree to root (recurse)
- unmark node

Cost: $\Theta(1 + \#\text{consecutive marked ancestors})$

Amortize: $\Phi = \#\text{marked nodes} \rightarrow \hat{c} = O(1)$

...but we are affecting Φ(EXTRACT MIN)
(making new roots)

Fix → $\Phi = 2 \cdot \#\text{marked nodes}$



FIBONACCI HEAPS

- MIN
- INSERT
- MERGE

} $O(1)$

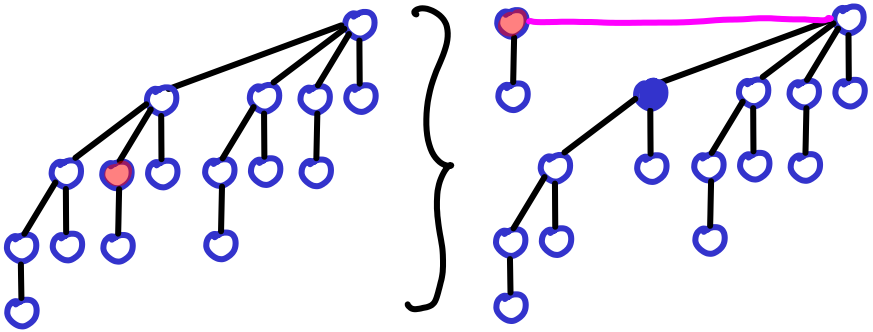
DECREASE $O(n)$, $O(1)$ am.

EXTRACT MIN $O(n)$, $O(\log n)$ am.

$\Phi = 2 \cdot \#\text{marked nodes} = 2m$

$\Phi = \#\text{roots} = r$

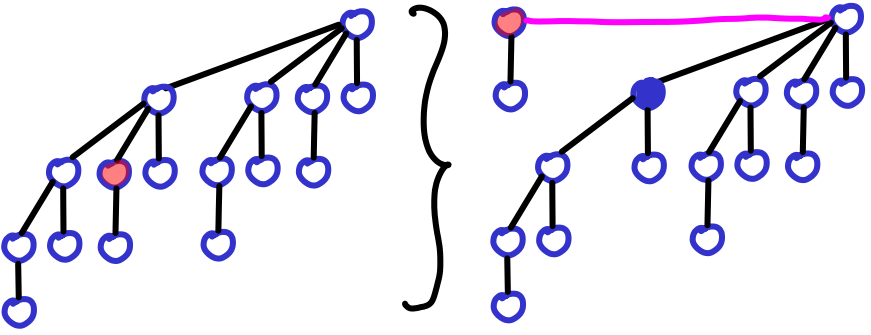
} total $\Phi = r + 2m$



we still don't have degree $= O(\log n)$

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE	$O(1)$	$O(1)$	
DECREASE	$O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	
EXTRACT MIN	$O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	

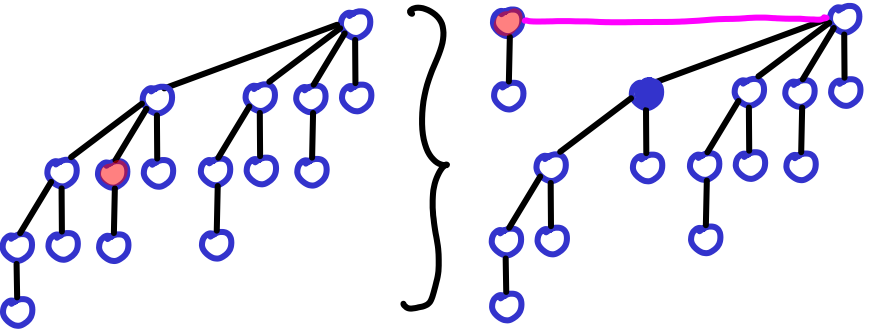


$\Phi = r + 2m$

roots + 2 · # marked nodes

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE	$O(1)$	$O(1)$	$O(1)$
DECREASE	$O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	
EXTRACT MIN	$O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	

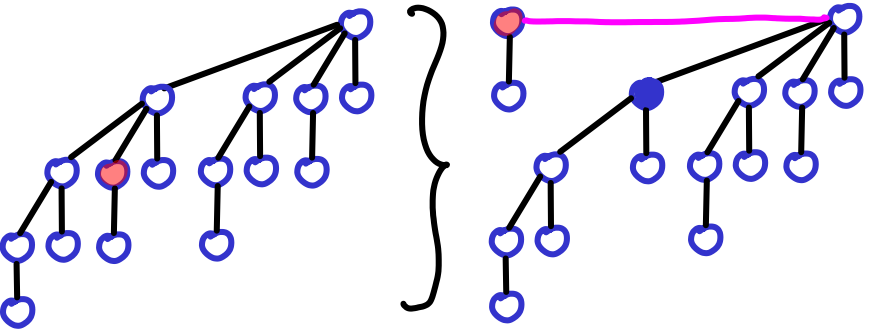


$\Phi = r + 2m$

roots + 2 · # marked nodes

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE	$O(1)$	$O(1)$	$O(1)$
DECREASE	$O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$
EXTRACT MIN	$O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	

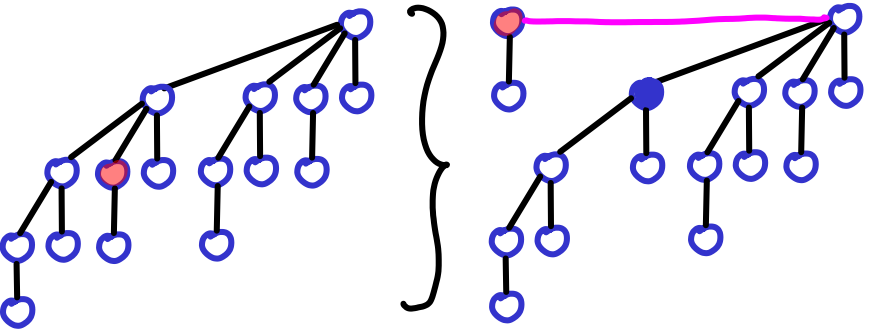


$\Phi = r + 2m$

roots + 2 · # marked nodes

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE	$O(1)$	$O(1)$	$O(1)$
DECREASE	$O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$	$O(1)$
EXTRACT MIN	$O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$		

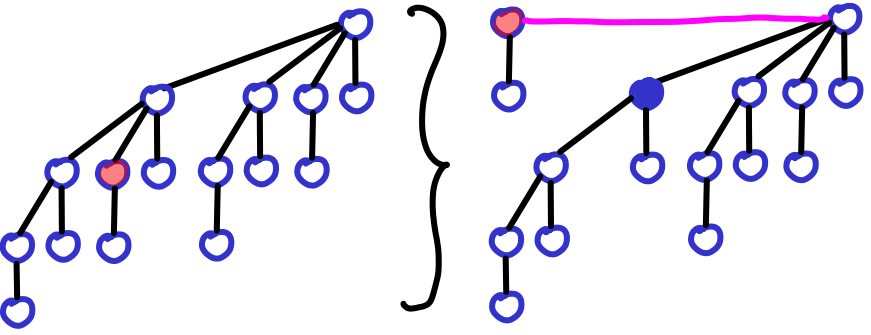


$\Phi = r + 2m$

roots + 2 · # marked nodes

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE — $O(1)$	$O(1)$	$O(1)$	$O(1)$
DECREASE $O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$	$O(1)$
EXTRACT MIN $O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	$r_{i+1} - r_i$ $(\Delta m = 0)$	

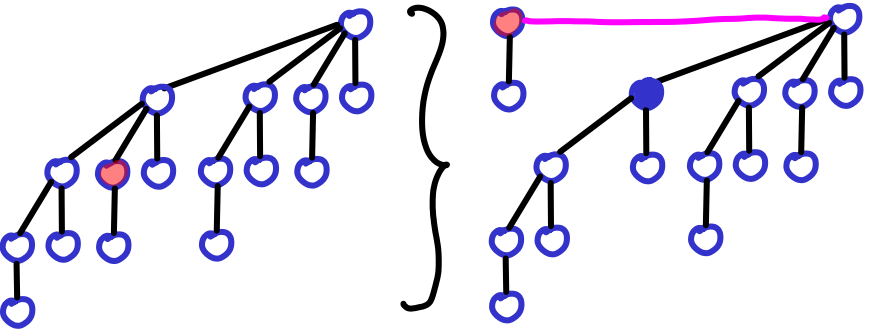


$\Phi = r + 2m$

roots + 2 · # marked nodes

FIBONACCI HEAPS

		c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE	$O(1)$	$O(1)$	$O(1)$	$O(1)$
DECREASE	$O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$	$O(1)$
EXTRACT MIN	$O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	$r_{i+1} - r_i$ $(\Delta m = 0)$	$r_{i+1} + \deg(\text{MIN})$



$\Phi = r + 2m$

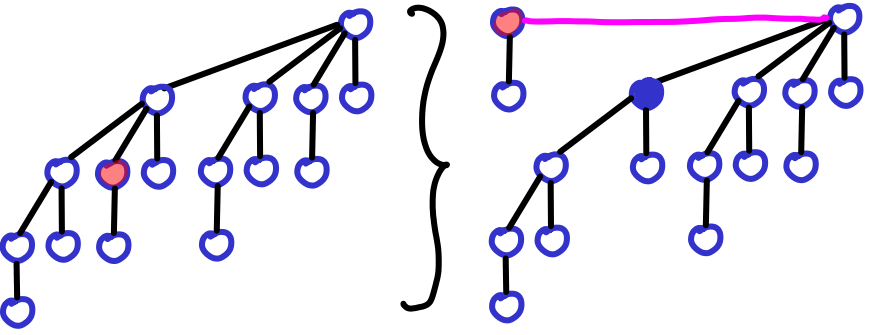
roots + 2 · # marked nodes

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE — $O(1)$	$O(1)$	$O(1)$	$O(1)$
DECREASE — $O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$	$O(1)$
EXTRACT MIN — $O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$	$r_{i+1} - r_i$ $(\Delta m = 0)$	$r_{i+1} + \deg(\text{MIN})$

need $\deg(\text{MIN}) = O(\log n)$

↳ implies $r_{i+1} = O(\log n)$



$\Phi = r + 2m$

roots + 2 · # marked nodes

FINALE: Bounding the degree of a node

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want max degree} = O(\log n)}$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: subtree size$(x) = s(x) \geq F_{\deg(x)+2}$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: subtree size$(x) = s(x) \geq F_{\deg(x)+2}$

$$F_k = \frac{\Phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: subtree size$(x) = s(x) \geq F_{\deg(x)+2}$

subtree size$(x) = s(x) \geq \underline{F_{\deg(x)+2} = \Omega(\Phi^{\deg(x)})}$

$F_k = \frac{\Phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: subtree size$(x) = s(x) \geq F_{\deg(x)+2}$

$\hookrightarrow$ $n \geq$ subtree size$(x) = s(x) \geq \underline{F_{\deg(x)+2} = \Omega(\Phi^{\deg(x)})}$

$F_k = \frac{\Phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: subtree size$(x) = s(x) \geq F_{\deg(x)+2}$

$\hookrightarrow$ $n \geq$ subtree size$(x) = s(x) \geq F_{\deg(x)+2} = \Omega(\Phi^{\deg(x)})$

$F_k = \frac{\Phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$

$\hookrightarrow$ $\log_\Phi n = \Omega(\deg(x))$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

We will show: $\text{subtree size}(x) = s(x) \geq F_{\deg(x)+2}$

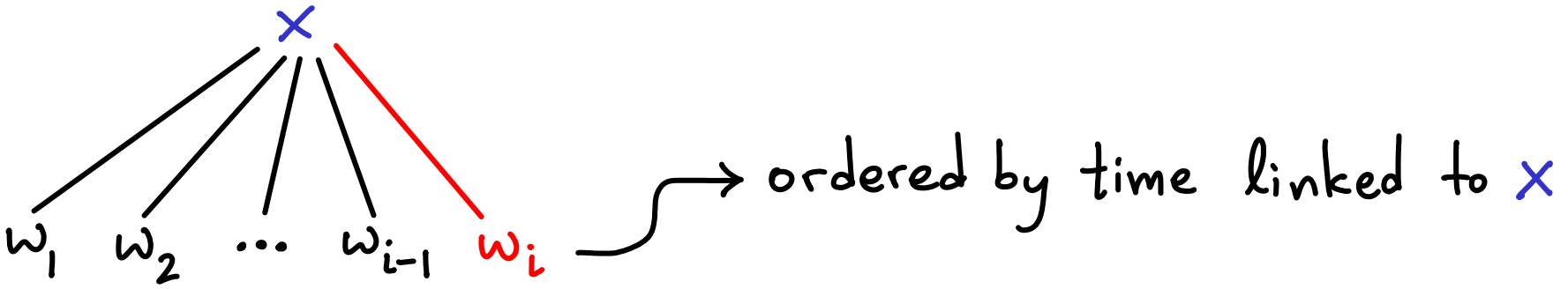
↳ $n \geq \text{subtree size}(x) = s(x) \geq F_{\deg(x)+2} = \Omega(\phi^{\deg(x)})$

$F_k = \frac{\phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$

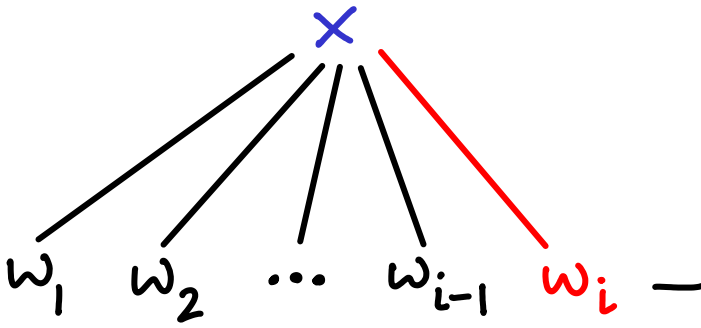
↳ $\log_\phi n = \Omega(\deg(x))$

subtree size$(x) = s(x) \geq F_{\deg(x)+2}$?

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?



subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

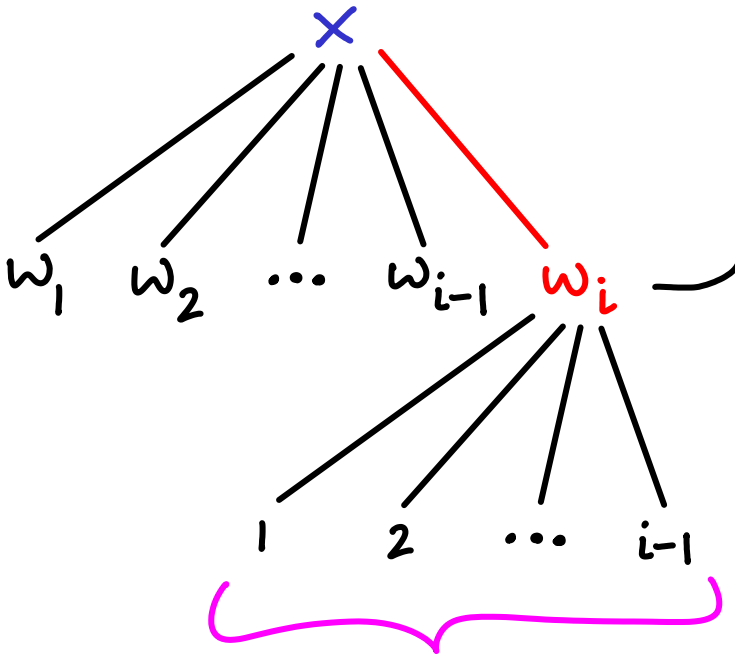


→ ordered by time linked to x

Just before adding w_i, x had $\geq i-1$ children.

(could have lost some after w_i)

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

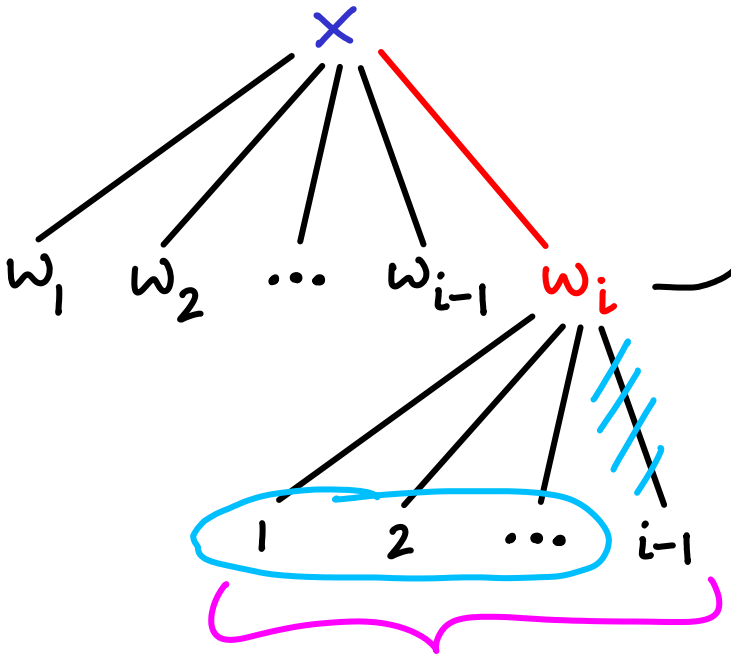


→ ordered by time linked to x

Just before adding w_i, x had $\geq i-1$ children.
(could have lost some after w_i)

So w_i had $\underline{deg(w_i) = deg(x)} \geq i-1$ (we link trees of equal degree)

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?



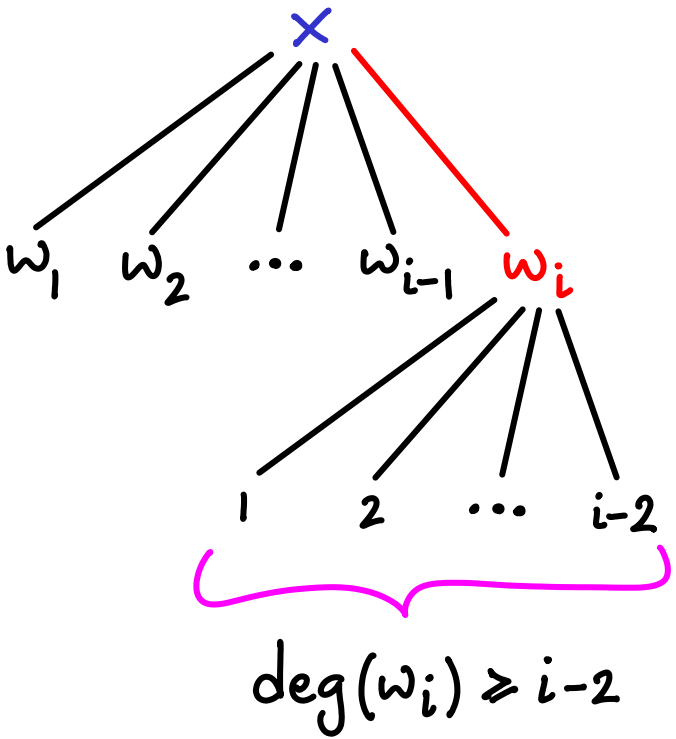
→ ordered by time linked to x

Just before adding w_i, x had $\geq i-1$ children.
(could have lost some after w_i)

So w_i had $deg(w_i) = deg(x) \geq i-1$ (we link trees of equal degree)

At this moment, $deg(w_i) \geq i-2$ (if it had lost 2 children, it would be a root)

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

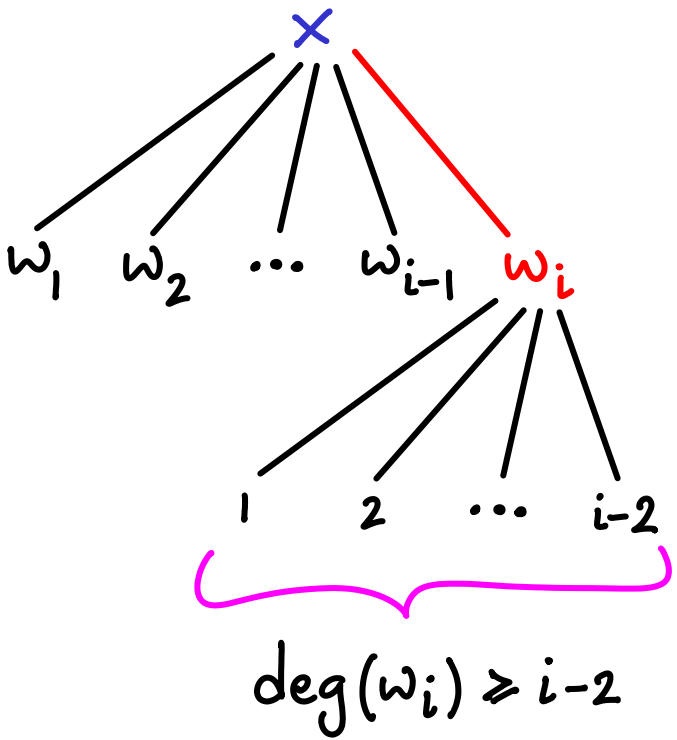


subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

---

if x has degree $deg(x) = d$,

how small can $s(x)$ be? Let $min = s_d$



$deg(w_i) \geq i-2$

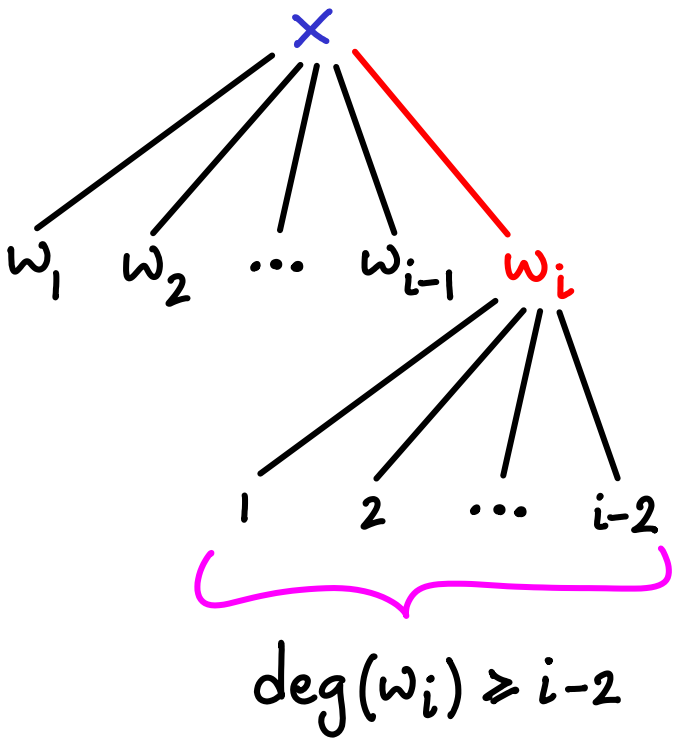
subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

---

if x has degree $deg(x) = d$,

how small can $s(x)$ be? Let $min = s_d$

$s_0 = 1$



$deg(w_i) \geq i-2$

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

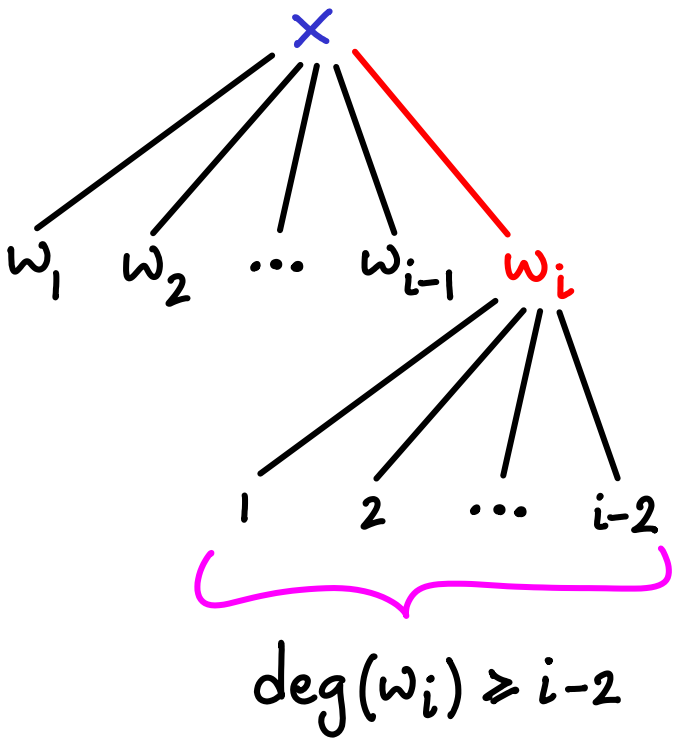
---

if x has degree $deg(x) = d$,

how small can $s(x)$ be? Let min = s_d

$s_0 = 1$

$s_1 = 2$



$deg(w_i) \geq i-2$

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

---

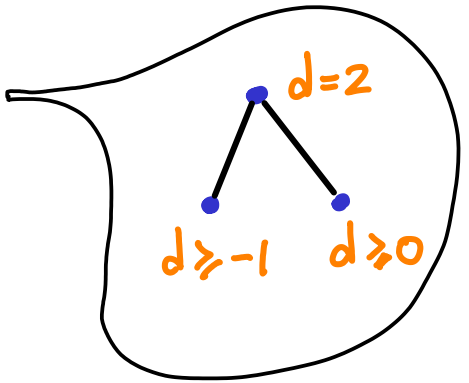
if x has degree $deg(x) = d$,

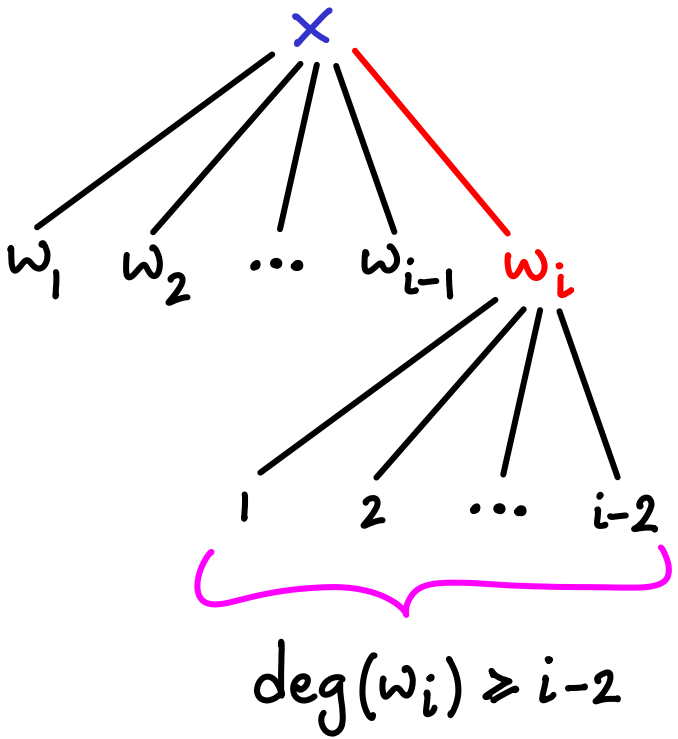
how small can $s(x)$ be? Let min = s_d

$s_0 = 1$

$s_1 = 2$

$s_2 = 3$





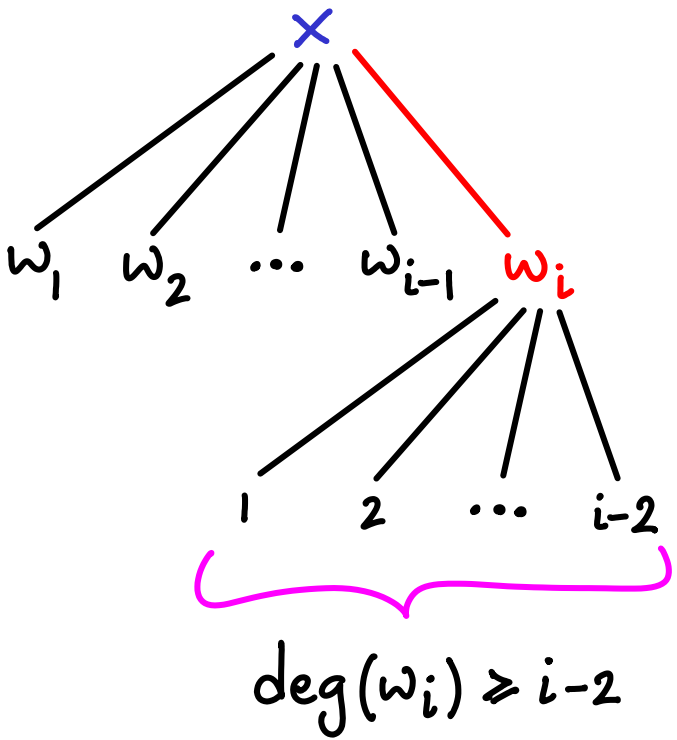
$deg(w_i) \geq i-2$

subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

---

if x has degree $deg(x) = d$,

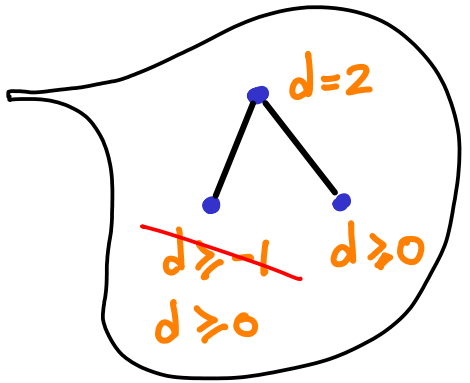
how small can $s(x)$ be? Let min = s_d



$s_0 = 1$

$s_1 = 2$

$s_2 = 3$



subtree size(x) = $s(x) \geq F_{deg(x)+2}$?

---

if x has degree $deg(x) = d$,

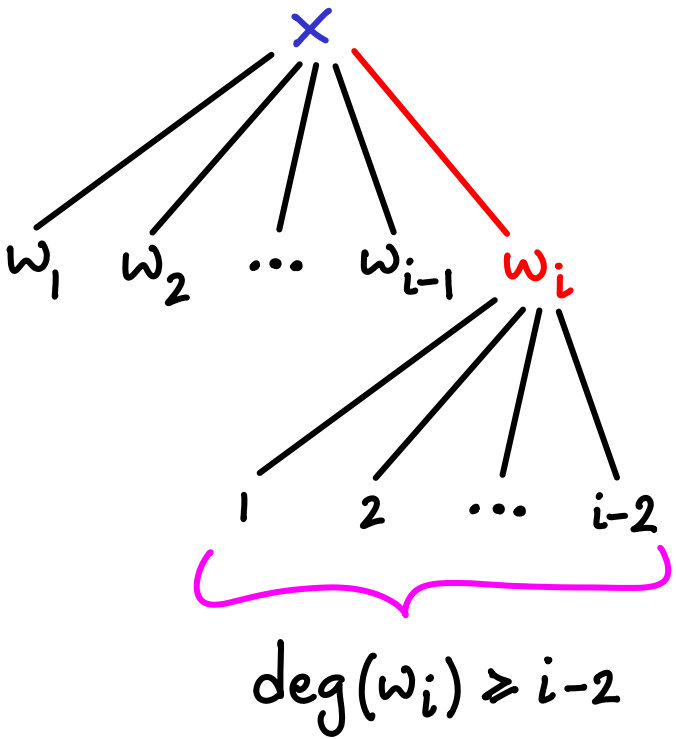
how small can $s(x)$ be? Let min = s_d

$s_0 = 1$

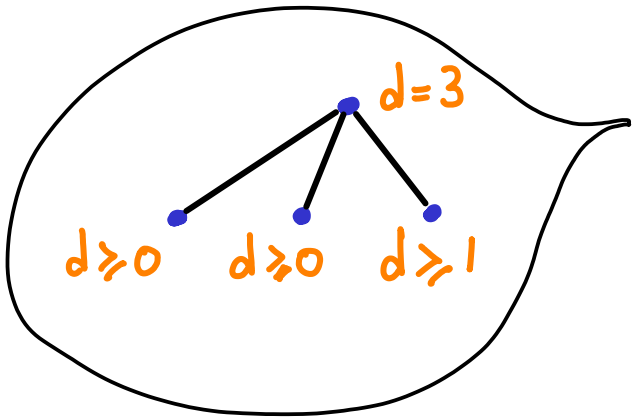
$s_1 = 2$

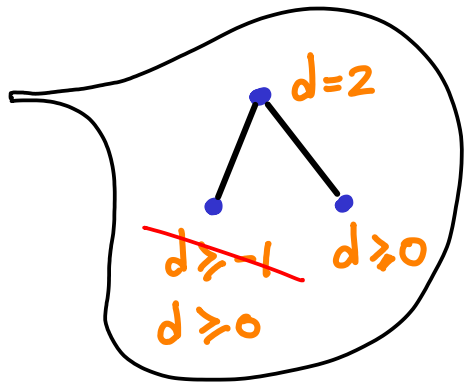
$s_2 = 3$

s_3



$deg(w_i) \geq i-2$





subtree size$(x) = s(x) \geq F_{\deg(x)+2}$?

---

if x has degree $\deg(x) = d$,

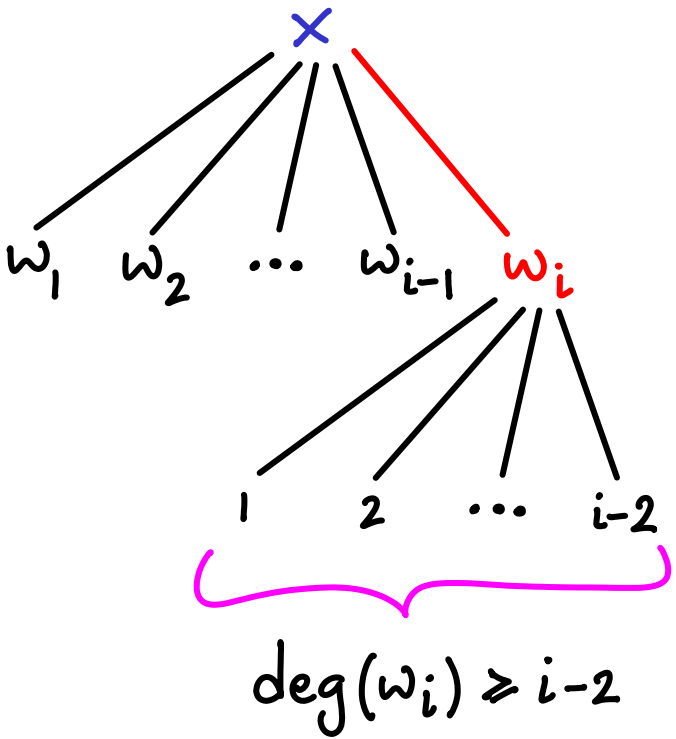
how small can $s(x)$ be? Let $\min = s_d$

$s_0 = 1$

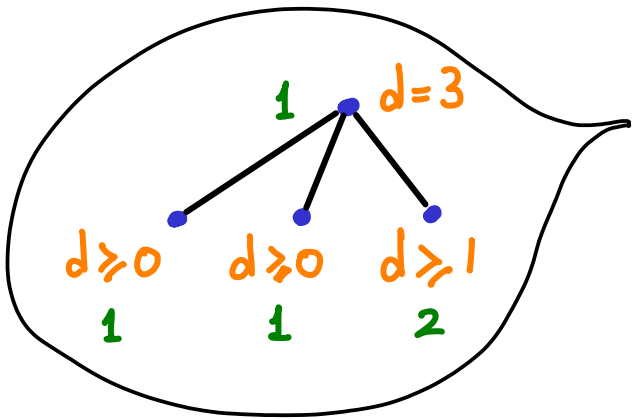
$s_1 = 2$

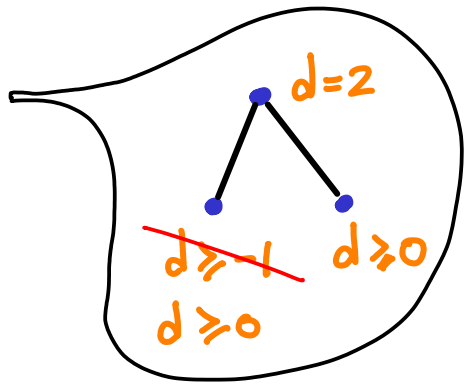
$s_2 = 3$

$s_3 = 5$



$\deg(w_i) \geq i-2$



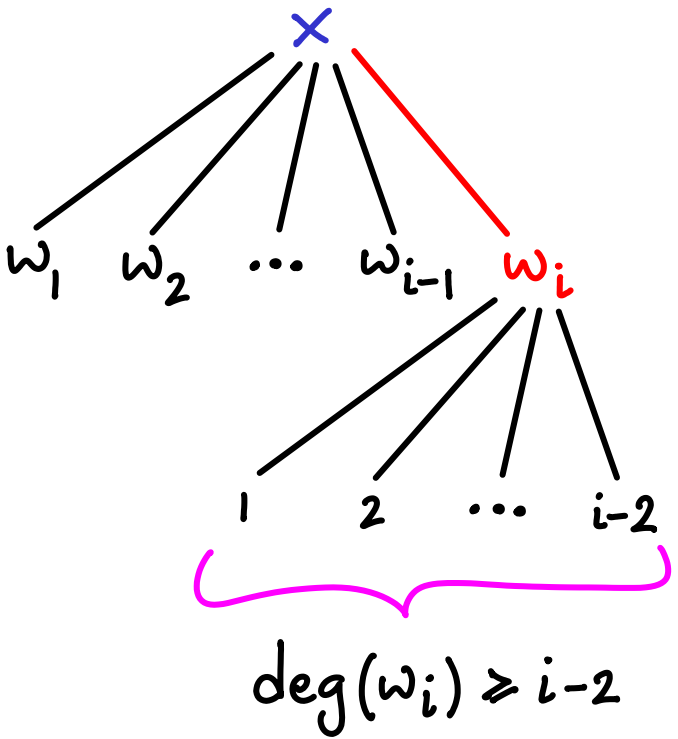


subtree size$(x) = s(x) \geq F_{\deg(x)+2}$?

---

if x has degree $\deg(x) = d$,

how small can $s(x)$ be? Let min $= s_d$



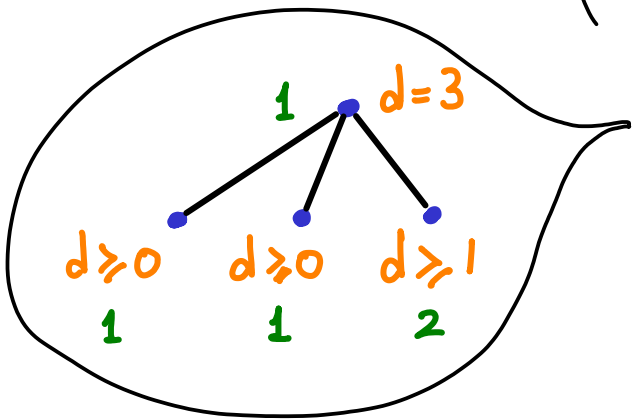
$s_{-1} = 1$

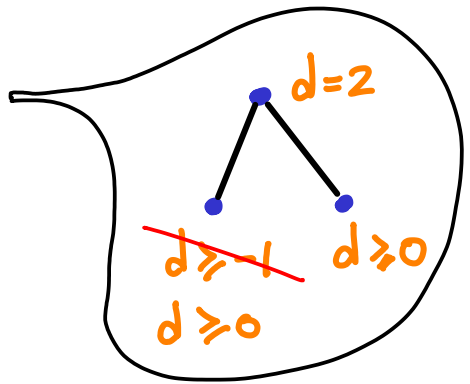
$s_0 = 1$

$s_1 = 2$

$s_2 = 3$

$s_3 = 5$





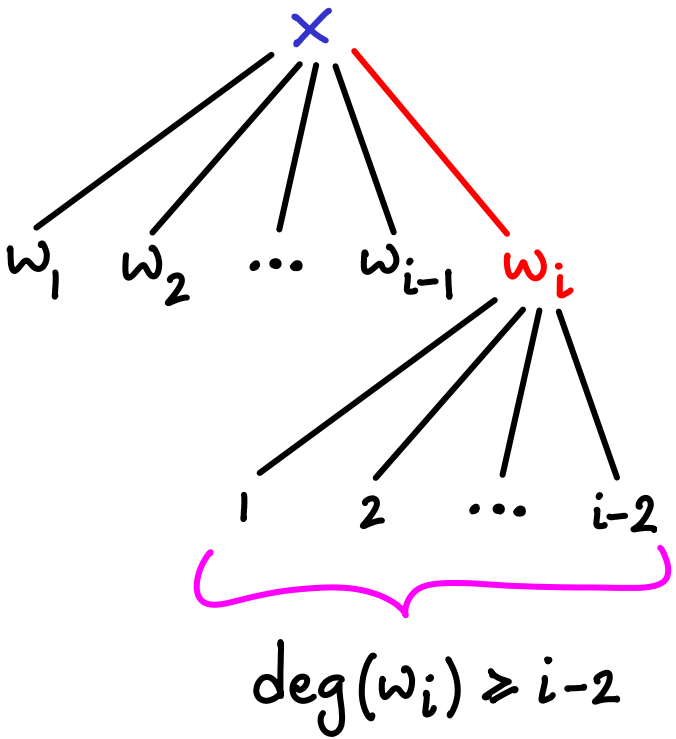
$$s_d = 1 + \sum_{i=1}^{\deg(x)} s_{i-2}$$

subtree size$(x) = s(x) \geq F_{\deg(x)+2}$?

---

if x has degree $\deg(x) = d$,

how small can $s(x)$ be? Let min $= s_d$



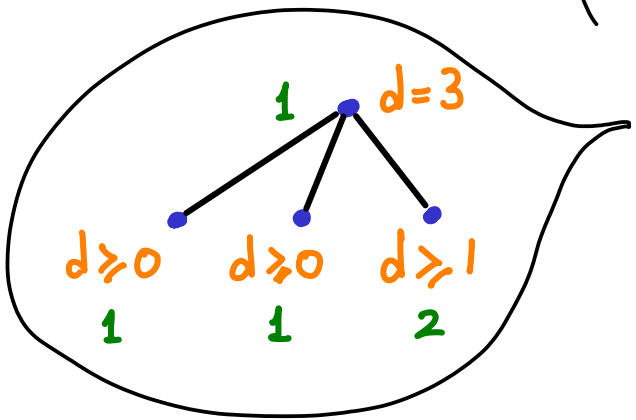
$s_{-1} = 1$ F_1

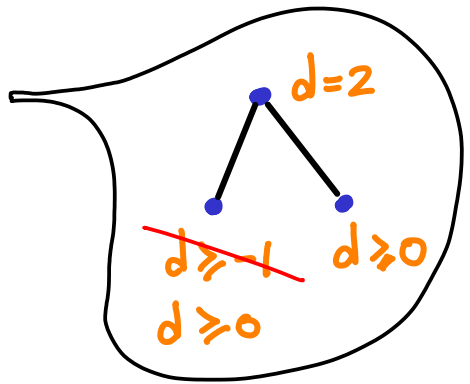
$s_0 = 1$ F_2

$s_1 = 2$ F_3

$s_2 = 3$ ⋮

$s_3 = 5$





$$s_d = 1 + \sum_{i=1}^{\deg(x)} s_{i-2} = F_{d+2}$$

FINALE: Bounding the degree of a node

EXTRACT-MIN $\hat{c} = r_{i+1} + \underbrace{\deg(\text{MIN})}_{\text{want}}$ max degree $= O(\log n)$

SHOWED: subtree size$(x) \geq F_{\deg(x)+2}$

↳ $n \geq$ subtree size$(x) \geq F_{\deg(x)+2} = \Omega(\Phi^{\deg(x)})$

$F_k = \frac{\Phi^k}{\sqrt{5}} > 2F_{k-2} = \Omega(2^{k/2})$

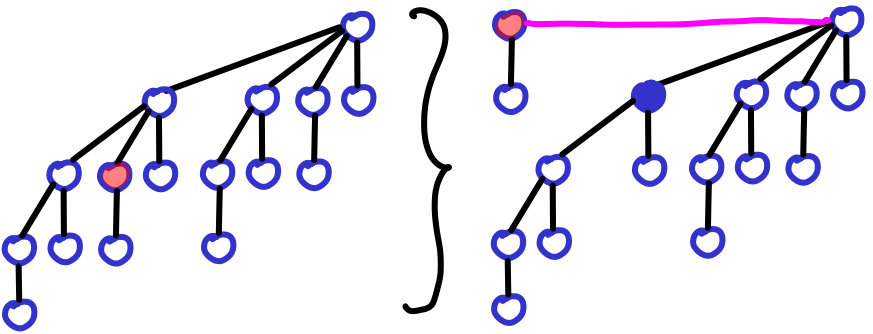
↳ $\boxed{\log_\Phi n = \Omega(\deg(x))}$

FIBONACCI HEAPS

	c	$\Delta\Phi$	$\hat{c}$
• MIN • INSERT • MERGE — $O(1)$	$O(1)$	$O(1)$	$O(1)$
DECREASE $O(n)$ $O(1)$ am.	$1+\Delta m$ $(\Delta m = m_i - m_{i+1})$	$1-\Delta m$ $(\Delta r = 1+\Delta m)$	$O(1)$
EXTRACT MIN $O(n)$ $O(\log n)$ am.	$r_i + \deg(\text{MIN})$ $= O(n) + O(\log n)$	$r_{i+1} - r_i$ $(\Delta m = 0)$	$r_{i+1} + \deg(\text{MIN})$ $= O(\log n)$

GOT $\deg(\text{MIN}) = O(\log n)$

↳ implies $r_{i+1} = O(\log n)$



$\Phi = r + 2m$

roots + 2 · # marked nodes