

2	10	5	4	3
•	•	•	•	•
↑	↑	↑	↑	↑
2	12	17	21	24

$$\begin{aligned} \text{Ave} &= \frac{2+12+17+21+24}{5} \\ &= \frac{76}{5} = 15.2 \end{aligned}$$

Week 04

2	3	4	5	10
↑	↑	↑	↑	↑
2	5	9	14	24

$$\begin{aligned} \text{Ave} &= \frac{2+5+9+14+24}{5} \\ &= \frac{54}{5} = 10.8 \end{aligned}$$

Priority Queues, binary heaps,
Symbol Tables, ordered, unordered,
BST's

PQ API

```
public class MaxPQ<Key extends Comparable<Key>>
```

```
    MaxPQ()
```

create an empty priority queue

```
    MaxPQ(Key[] a)
```

create a priority queue with given keys

```
    void insert(Key v)
```

insert a key into the priority queue

```
    Key delMax()
```

return and remove a largest key

```
    boolean isEmpty()
```

is the priority queue empty?

```
    Key max()
```

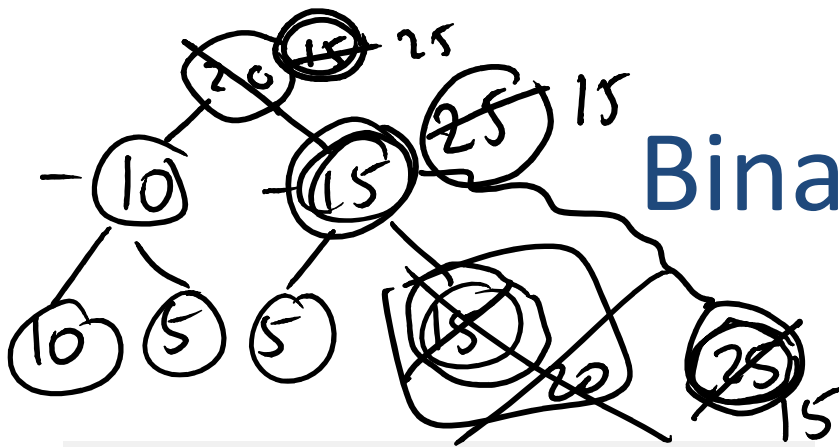
return a largest key

```
    int size()
```

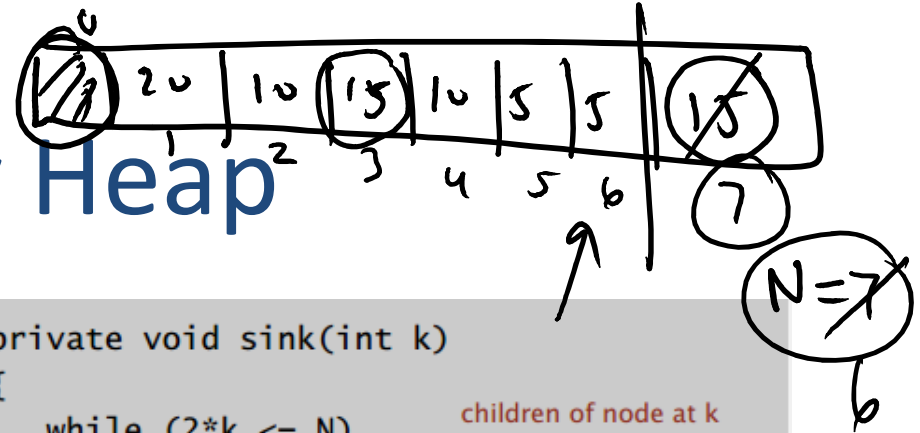
number of entries in the priority queue

Implementation

implementation	insert	del max	max
unordered array	1	N	N
ordered array	N	1	1
goal	$\log N$	$\log N$	$\log N$



Binary Heap



Heap-ordered binary tree.

- Keys in nodes
- Parent's key no smaller than children's keys.

max-heap

Array representation.

- Indices start at 1.
- Take nodes in level order.
- No explicit links needed!

```
private void sink(int k)
```

```
{
```

```
    while (2*k <= N)
```

```
    {
```

```
        int j = 2*k;
```

```
        if (j < N && less(j, j+1)) j++;
```

```
        if (!less(k, j)) break;
```

```
        exch(k, j);
```

```
        k = j;
```

```
    }
```

```
}
```

children of node at k
are 2*k and 2*k+1

```
private void swim(int k)
```

```
{
```

```
    while (k > 1 && less(k/2, k))
```

```
    {
```

```
        exch(k, k/2);
```

```
        k = k/2;
```

```
    }
```

```
}
```

parent of node at k is at k/2

Priority queues: quiz 1

How many compares (in the worst case) to **insert** in a d -way heap?

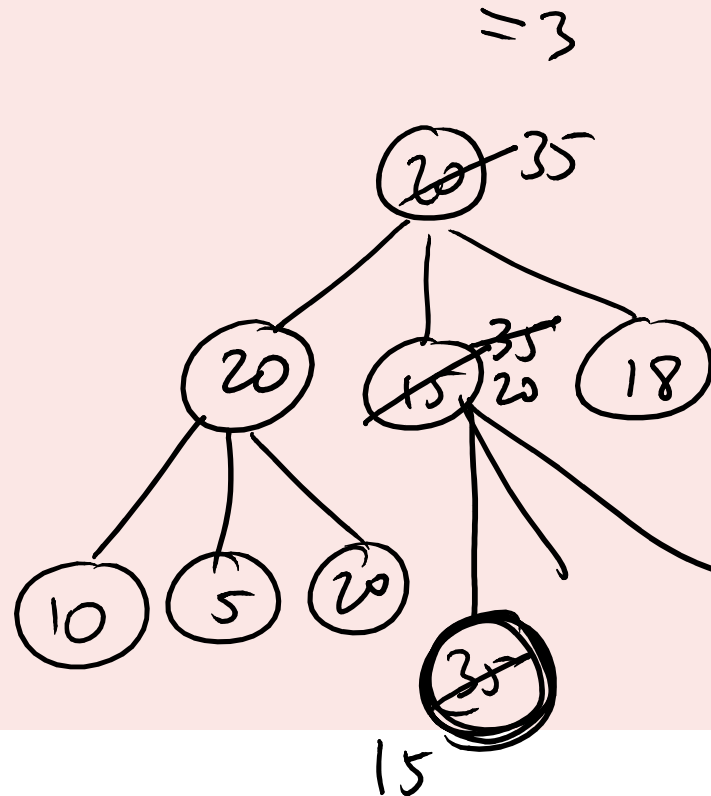
A. $\sim \log_2 N$

B. $\sim \log_d N$

C. $\sim d \log_2 N$

D. $\sim d \log_d N$

E. *I don't know.*



Implementation cost summary

implementation	insert	del max	max
unordered array	1	N	N
ordered array	N	1	1
binary heap	$\log N$	$\log N$	1
d-ary heap	$\log_d N$	$d \log_d N$	1
Fibonacci	1	$\log N^\dagger$	1
Brodal queue	1	$\log N$	1
impossible	1	1	1

← sweet spot: $d = 4$

← why impossible?

\dagger amortized

order-of-growth of running time for priority queue with N items



Priority queues: quiz 3

What is the name of this sorting algorithm?

```
public void sort(String[] a)
{
    int N = a.length;
    MaxPQ<String> pq = new MaxPQ<String>();
    for (int i = 0; i < N; i++)
        pq.insert(a[i]);
    for (int i = N-1; i >= 0; i--)
        a[i] = pq.delMax();
}
```



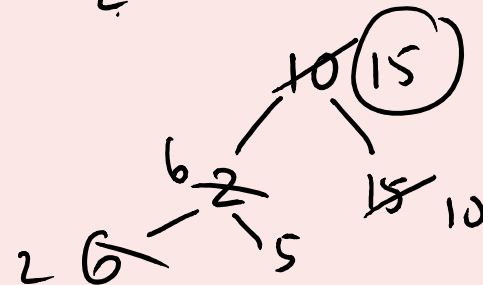
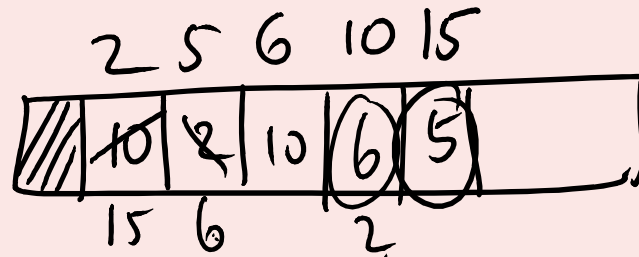
A. Insertion sort.

B. Mergesort.

C. Quicksort.

D. *None of the above.*

E. *I don't know.*

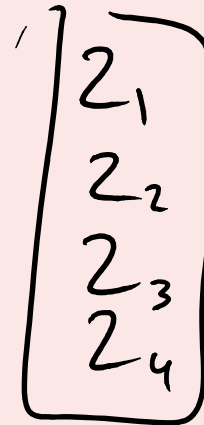
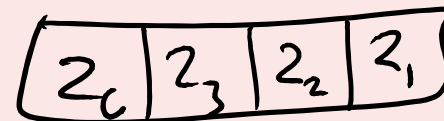
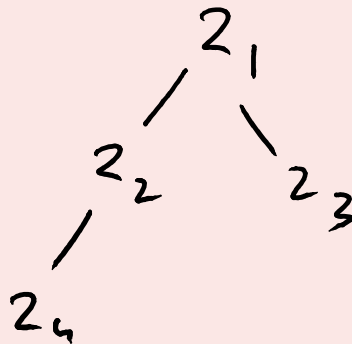


Priority queues: quiz 4

What are its properties?

```
public void sort(String[] a)
{
    int N = a.length;
    MaxPQ<String> pq = new MaxPQ<String>();
    [ for (int i = 0; i < N; i++) ]  $N \log N$ 
      pq.insert(a[i]);
    [ for (int i = N-1; i >= 0; i--) ]  $N \log N$ 
      a[i] = pq.delMax();
}
```

- A. $O(N \log N)$ compares in the worst case. ✓
- B. In-place. ✗
- C. Stable. ✗
- D. All of the above.
- E. I don't know.



Sorting algorithms: summary

	inplace?	stable?	best	average	worst	remarks
selection	✓		$\frac{1}{2} N^2$	$\frac{1}{2} N^2$	$\frac{1}{2} N^2$	N exchanges
insertion	✓	✓	N	$\frac{1}{4} N^2$	$\frac{1}{2} N^2$	use for small N or partially ordered
shell	✓		$N \log_3 N$?	$c N^{3/2}$	tight code; subquadratic
merge		✓	$\frac{1}{2} N \lg N$	$N \lg N$	$N \lg N$	$N \log N$ guarantee; stable
timsort		✓	N	$N \lg N$	$N \lg N$	improves mergesort when preexisting order
quick	✓		$N \lg N$	$2 N \ln N$	$\frac{1}{2} N^2$	$N \log N$ probabilistic guarantee; fastest in practice
3-way quick	✓		N	$2 N \ln N$	$\frac{1}{2} N^2$	improves quicksort when duplicate keys
heap	✓		N	$2 N \lg N$	$2 N \lg N$	$N \log N$ guarantee; in-place
?	✓	✓	N	$N \lg N$	$N \lg N$	holy sorting grail

Basic symbol table API

Associative array abstraction. Associate one value with each key.

<code>public class ST<Key, Value></code>		
<code>ST()</code>	<i>create an empty symbol table</i>	
<code>void put(Key <u>key</u>, Value <u>val</u>)</code>	<i>put key-value pair into the table</i>	← <code>a[key] = val;</code>
<code>Value get(Key key)</code>	<i>value paired with key</i>	← <code>a[key]</code>
<code>boolean contains(Key key)</code>	<i>is there a value paired with key?</i>	
<code>Iterable<Key> <u>keys()</u></code>	<i>all the keys in the table</i>	
<code>void delete(Key key)</code>	<i>remove key (and its value) from table</i>	
<code>boolean isEmpty()</code>	<i>is the table empty?</i>	
<code>int size()</code>	<i>number of key-value pairs in the table</i>	

Ordered symbol table API

```
public class ST<Key extends Comparable<Key>, Value>
```

```
    :
```

```
    Key min() smallest key
```

```
    Key max() largest key
```

```
    Key floor(Key key) largest key less than or equal to key
```

```
    Key ceiling(Key key) smallest key greater than or equal to key
```

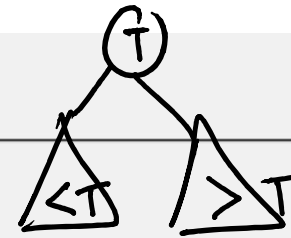
```
    int rank(Key key) number of keys less than key
```

```
    Key select(int k) key of rank k
```

```
    :
```

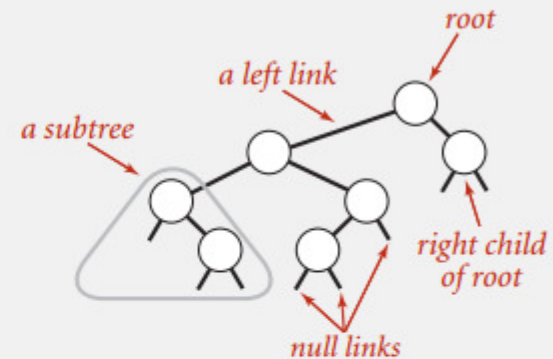
Binary search trees

Definition. A BST is a **binary tree** in **symmetric order**.



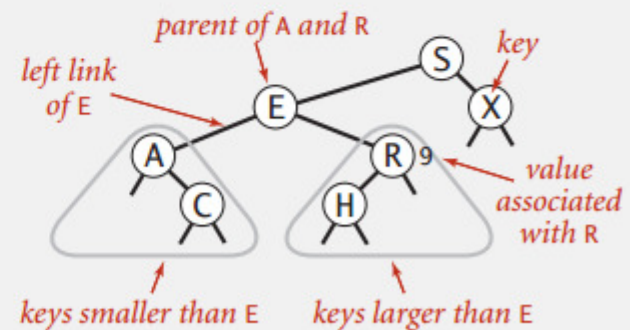
A binary tree is either:

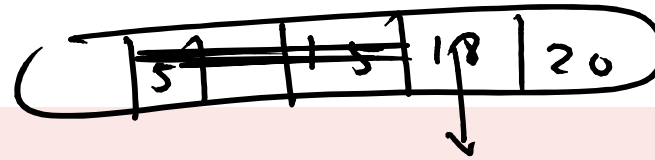
- Empty.
- Two disjoint binary trees (left and right).



Symmetric order. Each node has a key, and every node's key is:

- Larger than all keys in its left subtree.
- Smaller than all keys in its right subtree.





Binary search trees: quiz 1

Given N distinct keys, what is the name of this sorting algorithm?

1. **Shuffle** the keys.
2. **Insert** the keys into a BST, one at a time.
3. Do an **inorder traversal** of the BST.

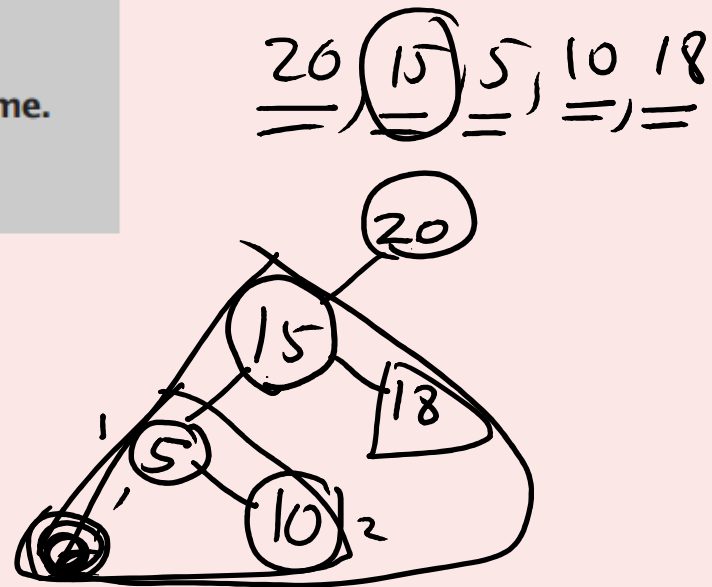
A. Insertion sort.

B. Mergesort.

C. Quicksort. *hhu*

D. None of the above.

E. I don't know.

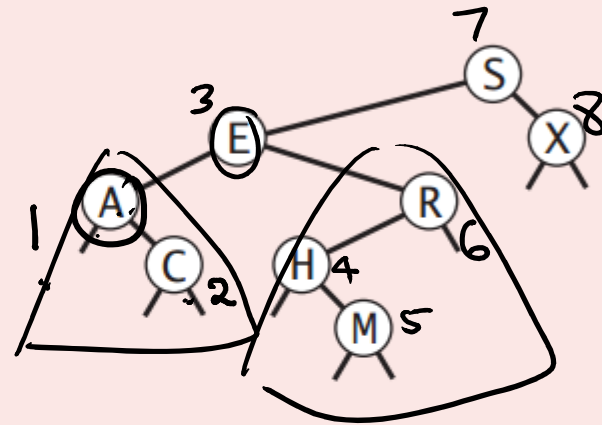


Inorder - L RT R
 5 10 15 18 20

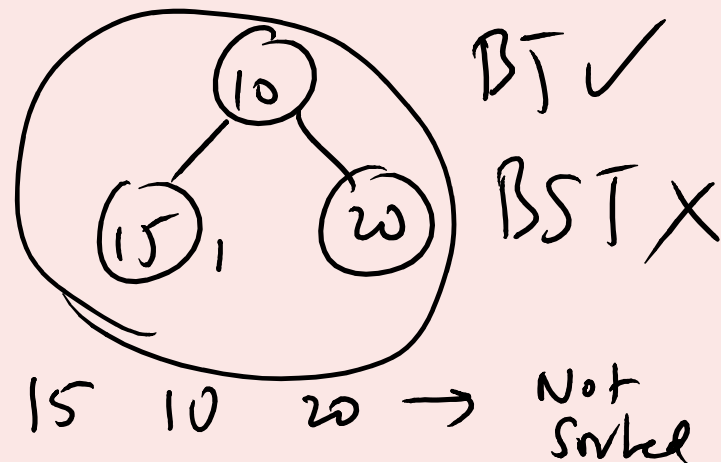
Binary search trees: quiz 2

In what order does the `traverse(root)` code print out the keys in the BST?

```
private void traverse(Node x)
{
    if (x == null) return;
    → traverse(x.left);
    → StdOut.println(x.key);
    traverse(x.right);
}
```




- A. A C E H M R S X
- B. A C E R H M X S
- C. S E A C R H M X
- D. C A M H R E X S
- E. *I don't know.*



BST: ordered symbol table operations summary

	sequential search	binary search	BST
search	N	$\log N$	h
insert	N	N	h
min / max	N	1	h
floor / ceiling	N	$\log N$	h
rank	N	$\log N$	h
select	N	1	h
ordered iteration	$N \log N$	N	N



h = height of BST
(proportional to $\log N$
if keys inserted in random order)

order of growth of running time of ordered symbol table operations

ST implementations: summary

implementation	guarantee		average case		ordered ops?	key interface
	search	insert	search hit	insert		
sequential search (unordered list)	N	N	N	N		<code>equals()</code>
binary search (ordered array)	$\log N$	N	$\log N$	N	✓	<code>compareTo()</code>
BST	N	N	$\log N$	$\log N$	✓	<code>compareTo()</code>
red-black BST	$\log N$	$\log N$	$\log N$	$\log N$	✓	<code>compareTo()</code>

Next lecture. **Guarantee** logarithmic performance for all operations.