COMP251: Review

Jérôme Waldispühl
School of Computer Science
McGill University

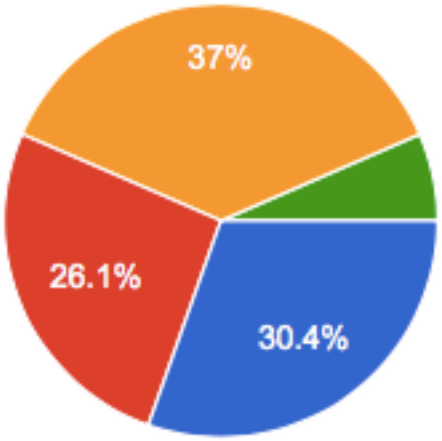
Running-time O (m+n) is equivalent to:

A. O(m) + O (n)

B. O(max(m,n))

Which of these possibilities are true?

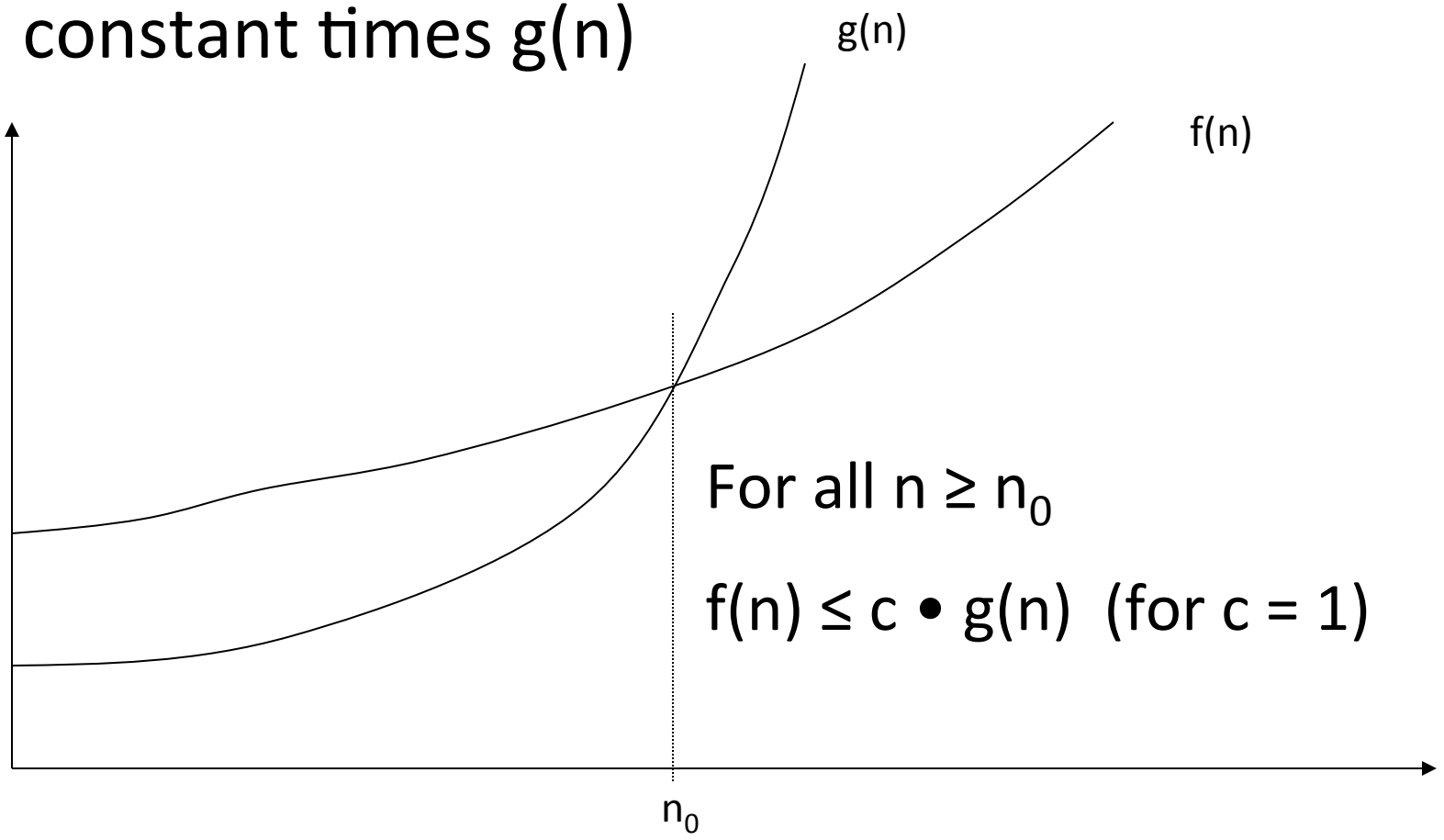
- A and B
- A only
- B only
- neither A or B



A and B	**28**	30.4%
A only	**24**	26.1%
B only	**34**	37%
neither A or B	**6**	6.5%

Intuition and visualization

- "f(n) is O(g(n))" iff there exists a point n_0 beyond which f(n) is less than some fixed constant times g(n)



From COMP250 M. Blanchette's slides

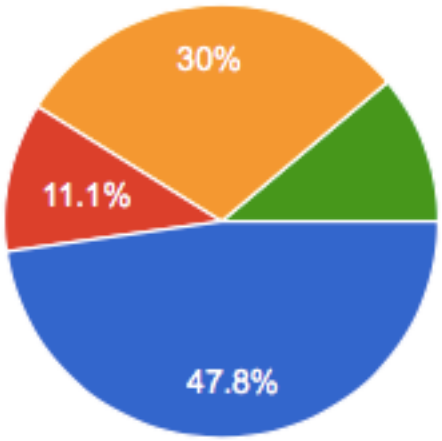
Let T1(n) = O(f(n)) and T2 (n) = O(f(n)). Given the statements:

A. T1(n) / T2(n) = O(1)

B. T1(n) + T2 (n) = O(f(n)).

Which of them are true?

- A and B
- A only
- B only
- neither A or B

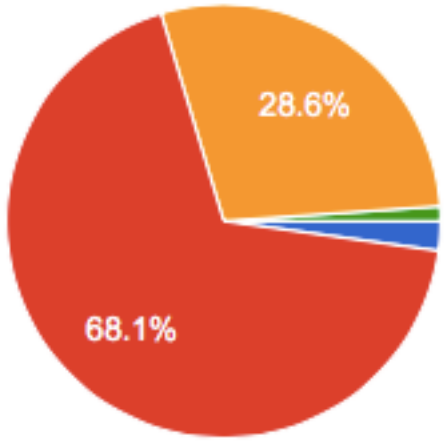


A and B	**43**	47.8%
A only	**10**	11.1%
B only	**27**	30%
neither A or B	**10**	11.1%

What is the time-complexity of the following piece of code in Big-Oh notation?

```
sum = 0;
for (int i = 0; i < n; i++) {
   for (j = 1; j < n; j = j*2) {
      sum += n; }}
```

- O(n)
- O(n*log(n))
- O(n^2)
- O(log(n))



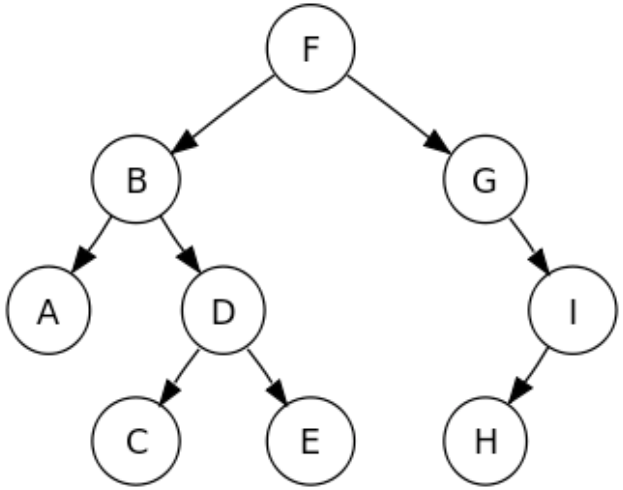
O(n)	**2**	2.2%
O(n*log(n))	**62**	68.1%
O(n^2)	**26**	28.6%
O(log(n))	**1**	1.1%

```
for (i=1; i<N; i=i*2) { … }
```

Value of i after k iterations: 2^k

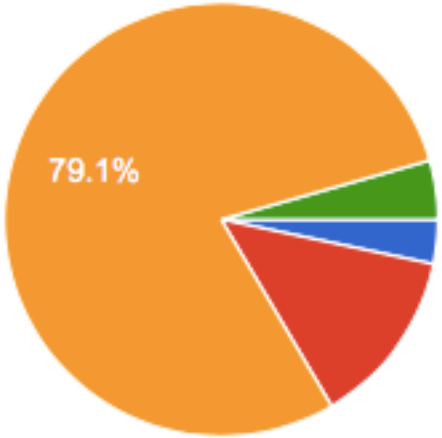
We have $i < N \Rightarrow 2^k < N \Rightarrow k < \log_2(N)$.

There is less than $\log_2(N)$ iterations, and the running time of this loop is O(log(n)).



For this Binary Tree, which of the following represents a post-order traversal?

- A, B, C, D, E, F, G, H, I
- F, B, A, D, C, E, G, I, H
- A, C, E, D, B, H, I, G, F
- None of the above

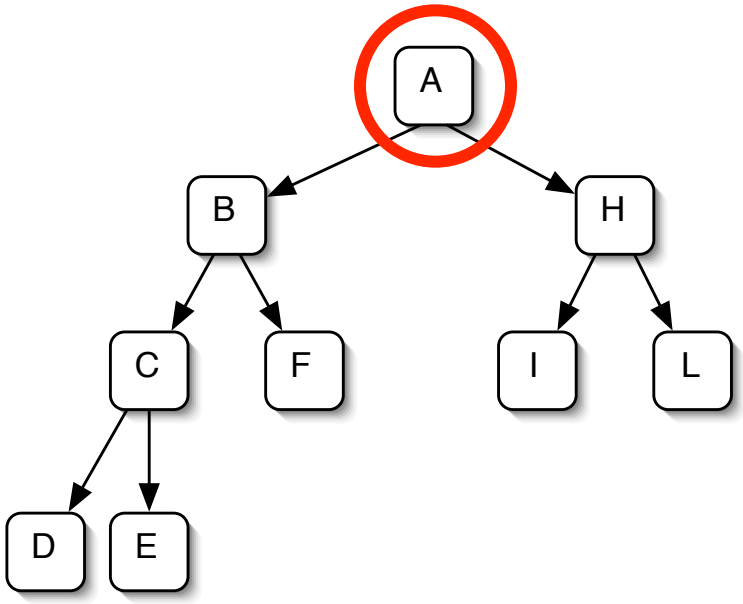


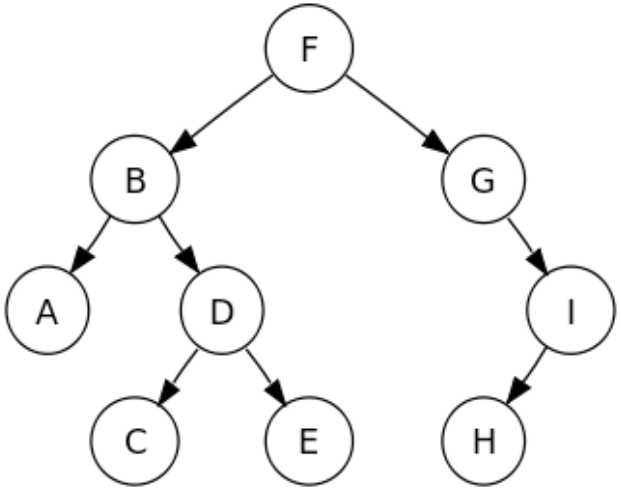
A, B, C, D, E, F, G, H, I	**3**	3.3%
F, B, A, D, C, E, G, I, H	**12**	13.2%
A, C, E, D, B, H, I, G, F	**72**	79.1%
None of the above	**4**	4.4%

Post-order traversal

```
postorderTraversal(treeNode x)
   for each c in children(x) do
      postorderTraversal(c);
   print x.value;
```

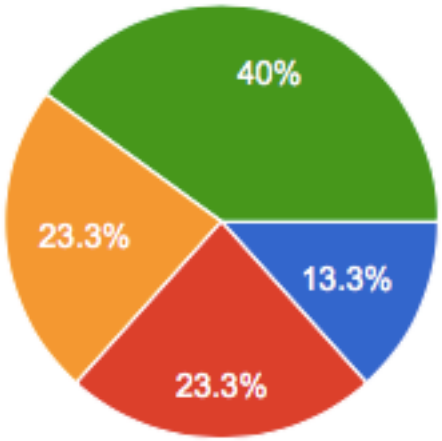
D E C F B I L H A





For the Binary Search Tree shown above, deletion of node F would result in which of the following nodes becoming the root node?

- B
- G
- B or G
- E or G



B	**12**	13.3%
G	**21**	23.3%
B or G	**21**	23.3%
E or G	**36**	40%

BST - remove

1) Find the node N to be removed using the "find" algo

```
2) If  (N is a leaf) { remove it }
   Else if (N is an internal node with only one child)
   {
      replace N by its child
   }
   Else if (N is an internal node with two children)
   {
      N will be replaced by the node N' that has the
      next largest key after (or before) N.
   }
```

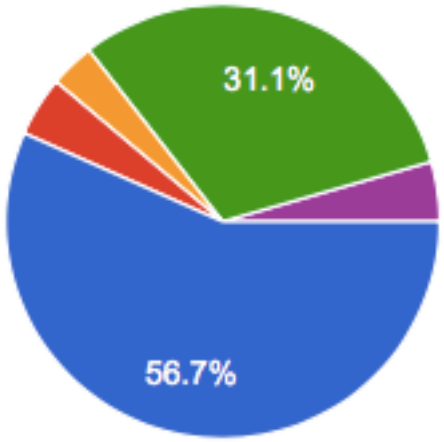
To find N':

1. Go to the right child of N
2. Go down left until no left child is found.

The node found is N'

Suppose we need to sort a list of employee records in ascending order, using the social security number (a 9-digit number) as the key (i.e., sort the records by social security number). If we need to guarantee that the running time will be no worse than n log n , which sorting methods could we use?

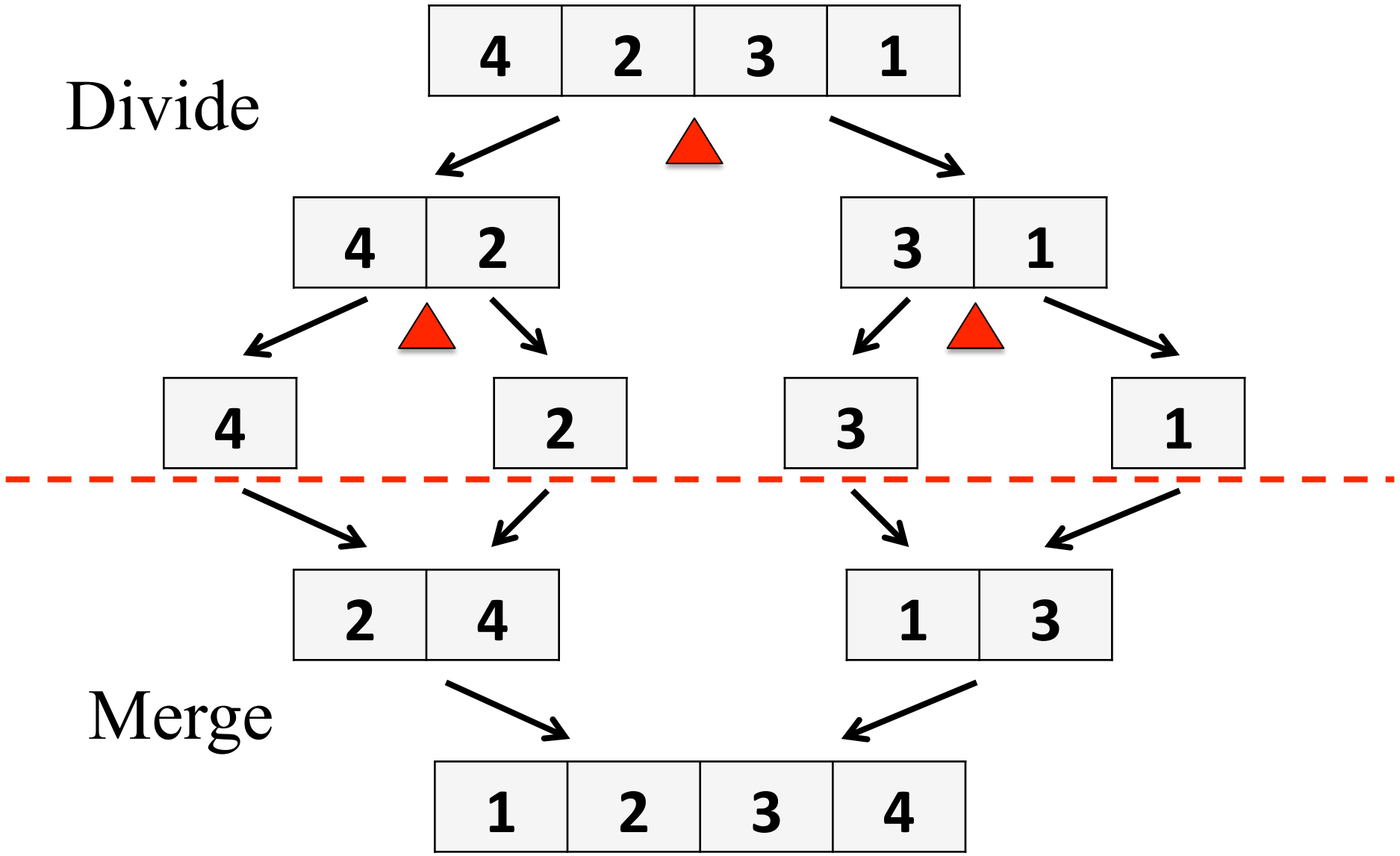
- Merge sort
- Quicksort
- Insertion sort
- Either merge sort or quicksort
- None of these sorting algorithms



Mergesort	**51**	56.7%
Quicksort	**4**	4.4%
Insertionsort	**3**	3.3%
Either mergesort or quicksort	**28**	31.1%
None of these sorting algorithms	**4**	4.4%

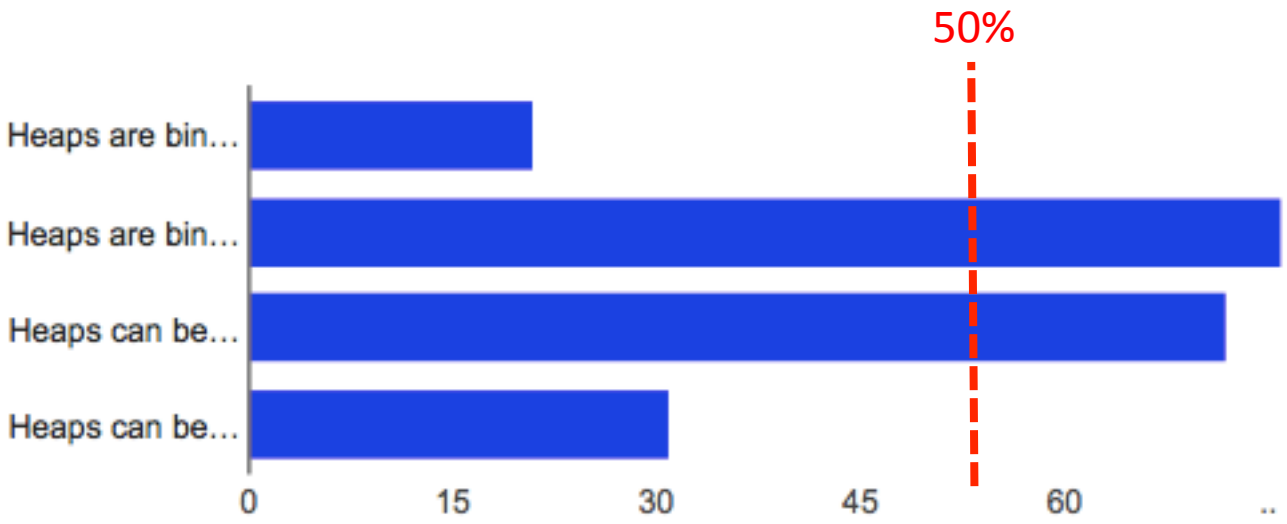
Algo	Best case	Average case	Worst case
MergeSort	O(n * log(n))	O(n * log(n))	O(n * log(n))
QuickSort	O(n * log(n))	O(n * log(n))	O(n ^ 2)
InsertionSort	O(n)	O(n ^ 2)	O(n ^ 2)
HeapSort	O(n * log(n))	O(n * log(n))	O(n * log(n))
BubbleSort	O(n)	O(n ^ 2)	O(n ^ 2)

Merge Sort



Which of the following assertions are true? (multiple choices)

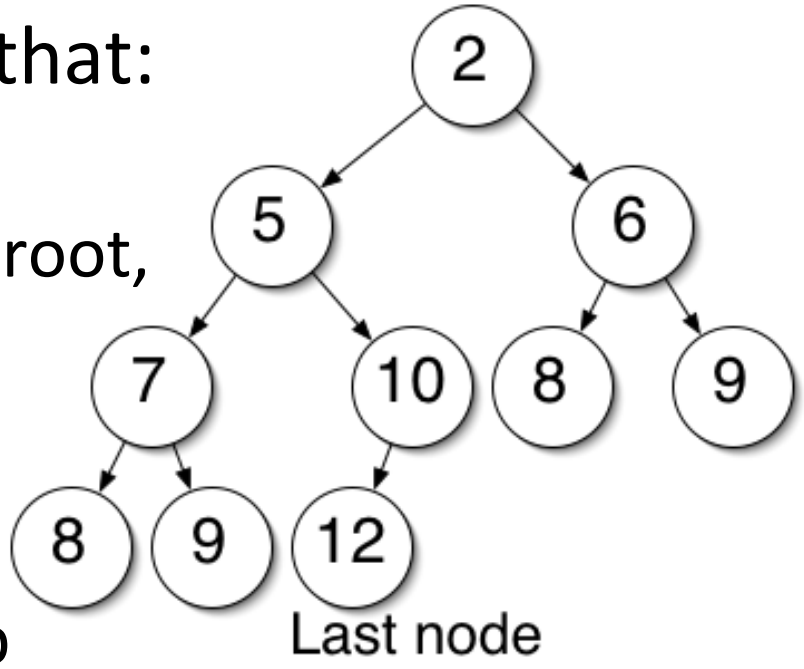
- ~~Heaps are binary search trees.~~
- Heaps are binary trees.
- Heaps can be used to implement priority queues.
- ~~Heaps can be used to implement lists.~~



Heap - Definition

A **heap** is a binary tree such that:

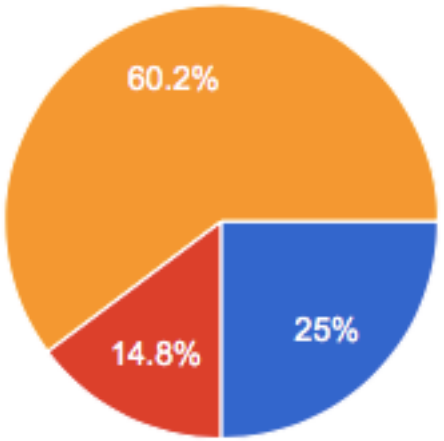
– For any node *n* other than the root,
key(n) ≥ key(parent(n))



– Let h be the height of the heap
- First h-1 levels are full:
  For i = 0,…,h-1, there are 2^i nodes of depth i
- At depth h, the leaves are packed on the left side of the tree

What is the time-complexity of the removal of the highest priority key in a heap (where n is the number of keys stored)?

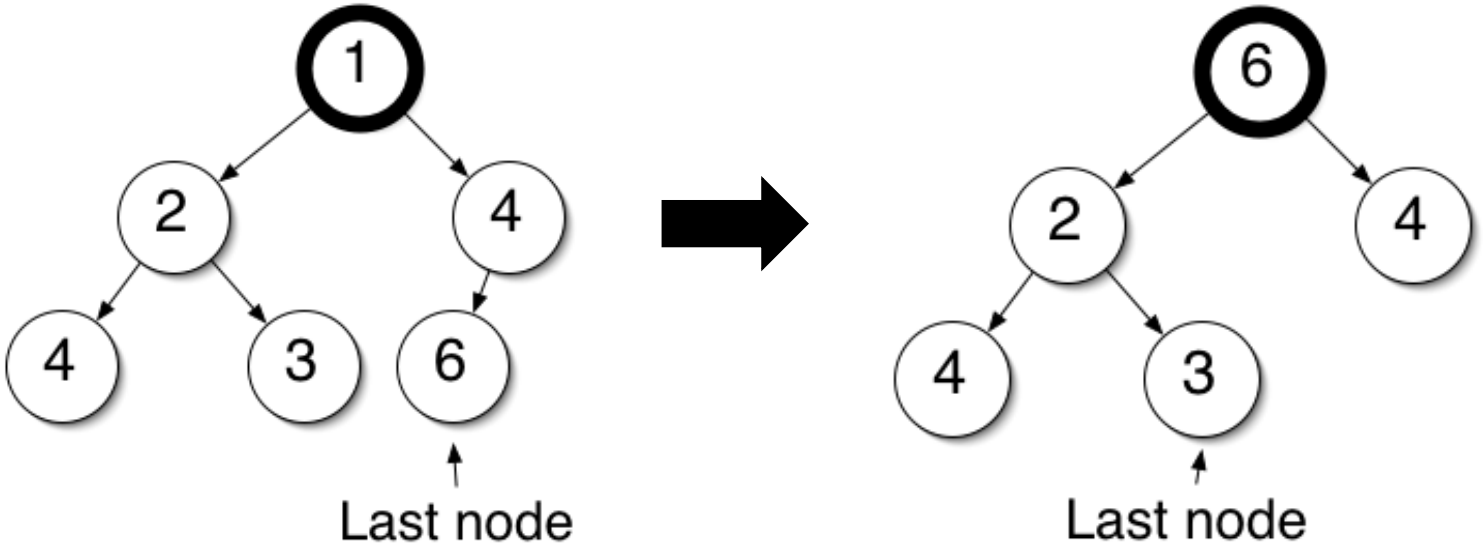
- O(1)
- O(n)
- O(log(n))



O(1)	**22**	25%
O(n)	**13**	14.8%
O(log(n))	**53**	60.2%

Heaps: RemoveMin()

- The minimum key is always at the root of the heap!
- Replace the root with last node

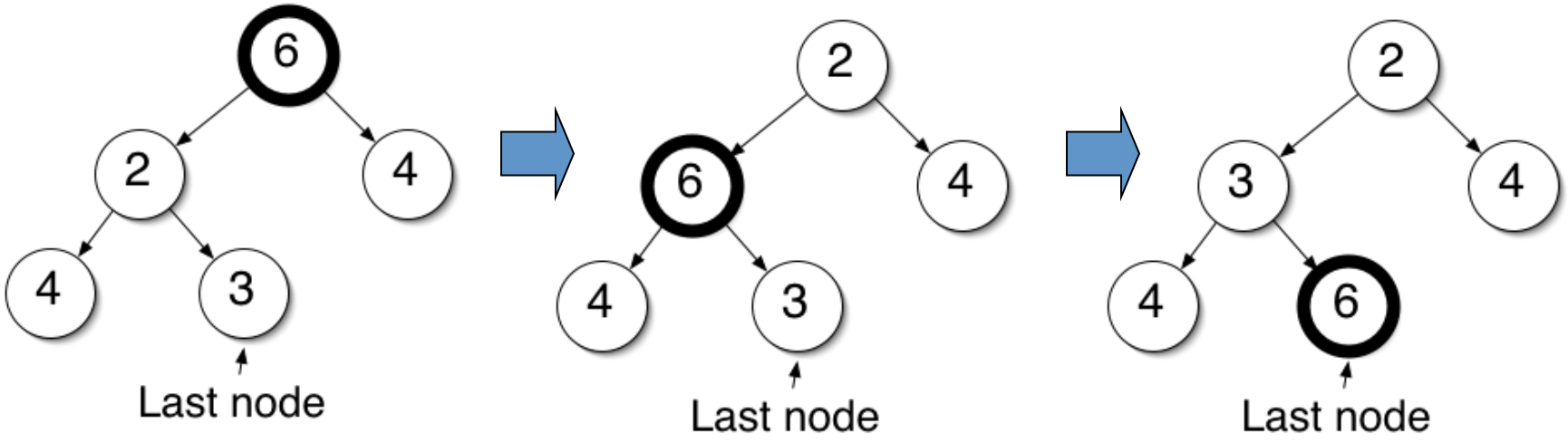


- Restore heap-order property (see next)

Heaps: Bubbling-down

Restoring the heap-order property:

- Keep swapping the node with its smallest child as long as the node's key is larger than it's child's key

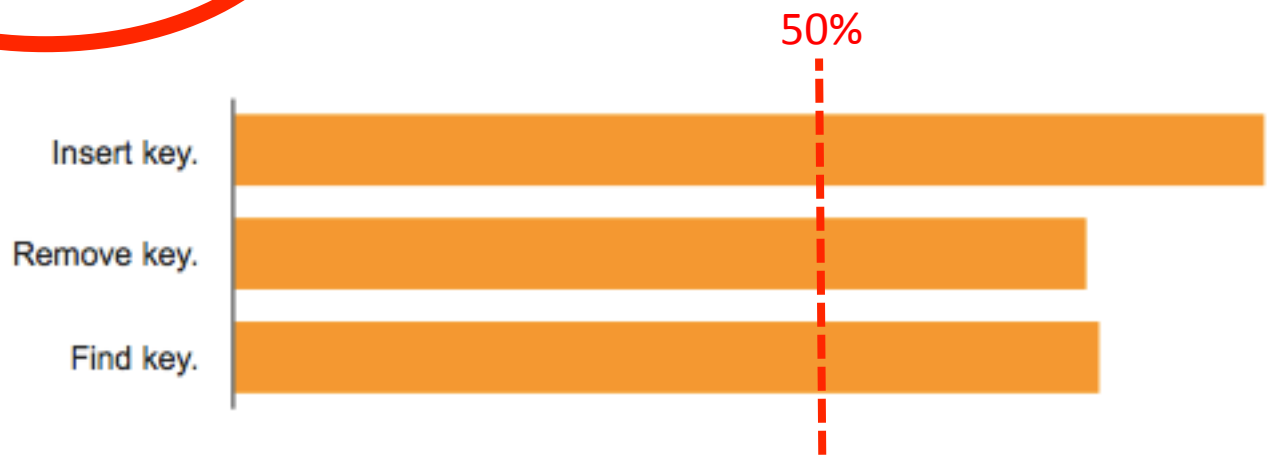


Running time?

$$O(h) = O(\log(n))$$

You are using a hash table to store keys. Assuming there is **no collision**, which of the following operations have a O(1) time-complexity? (multiple choices)

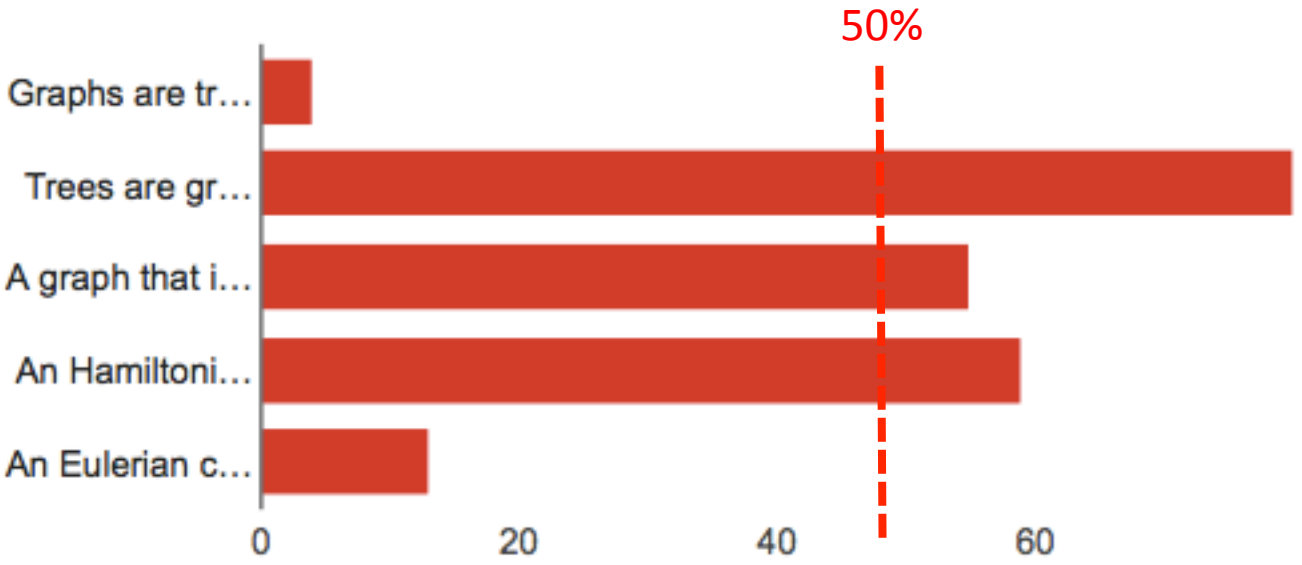
- Insert key.
- Remove key.
- Find key.



Insert key.	**75**	84.3%
Remove key.	**62**	69.7%
Find key.	**63**	70.8%

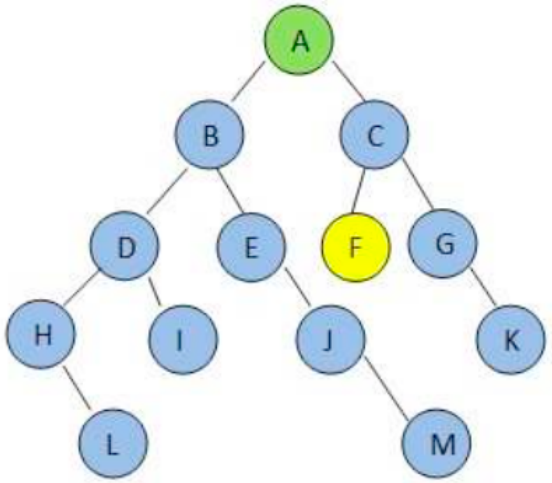
Which of the following assertions are true?* (multiple choices)

- ~~Graphs are trees.~~
- Trees are graphs.
- A graph that is not a tree has at least one cycle.**
- An Hamiltonian cycle visits each vertex exactly once.
- ~~An Eulerian cycle visits each vertex exactly once.~~



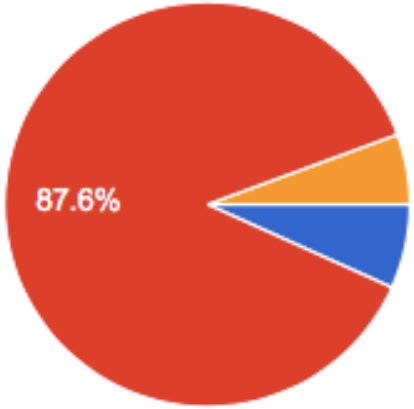
* The graph is connected, undirected and #nodes > 2.

** True only if edges are undirected.



In the graph shown above, starting from the green node at the top, which algorithm will visit the least number of nodes before visiting the yellow goal node?

- Depth First Search (DFS)
- Breadth First Search (BFS)
- BFS and DFS encounter same number of nodes before encounter the goal node.

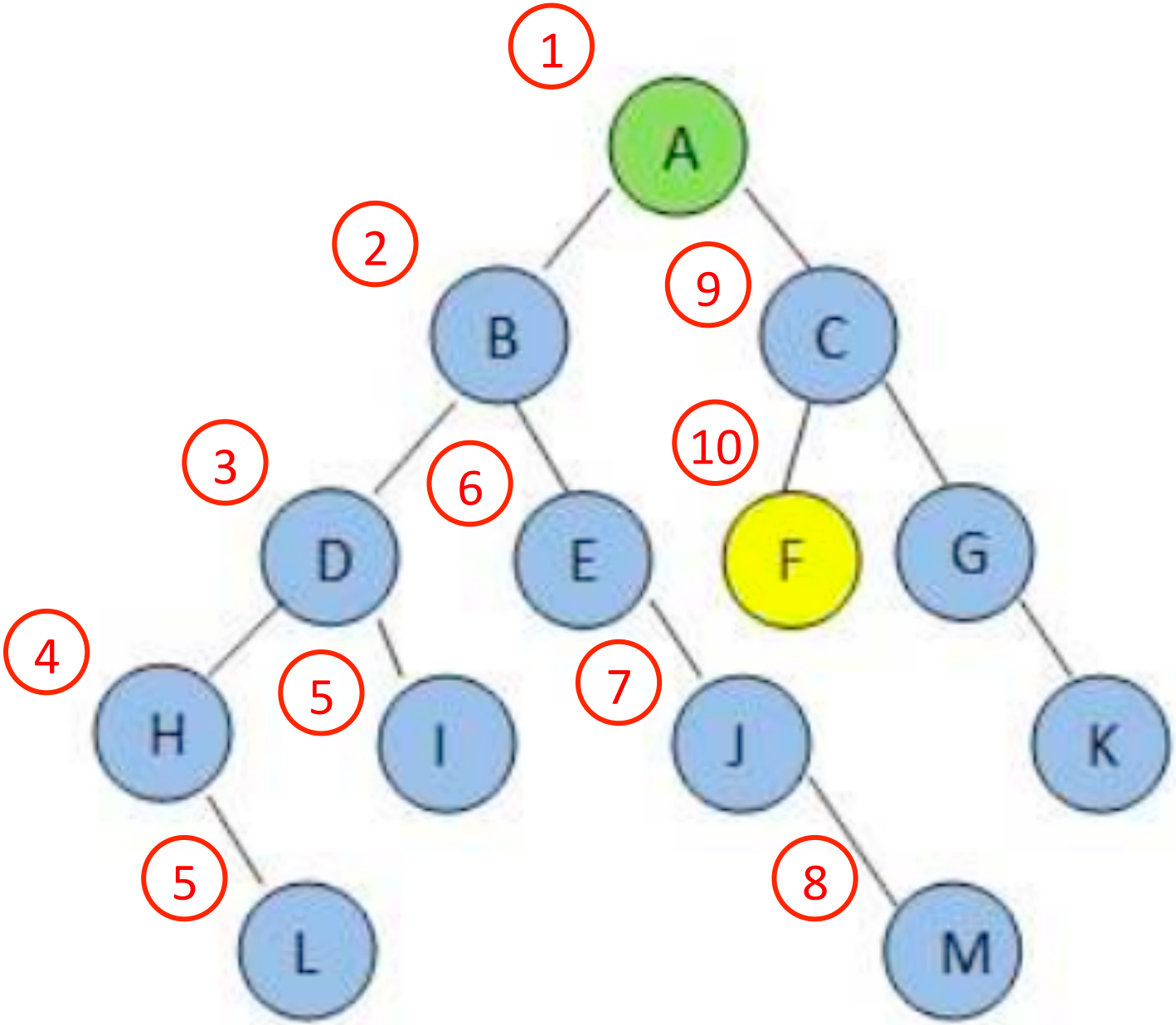


Depth First Search (DFS)	**6**	6.7%
Breadth First Search (BFS)	**78**	87.6%
BFS and DFS encounter same number of nodes before encounter the goal node.	**5**	5.6%

Depth-First Search

Idea:

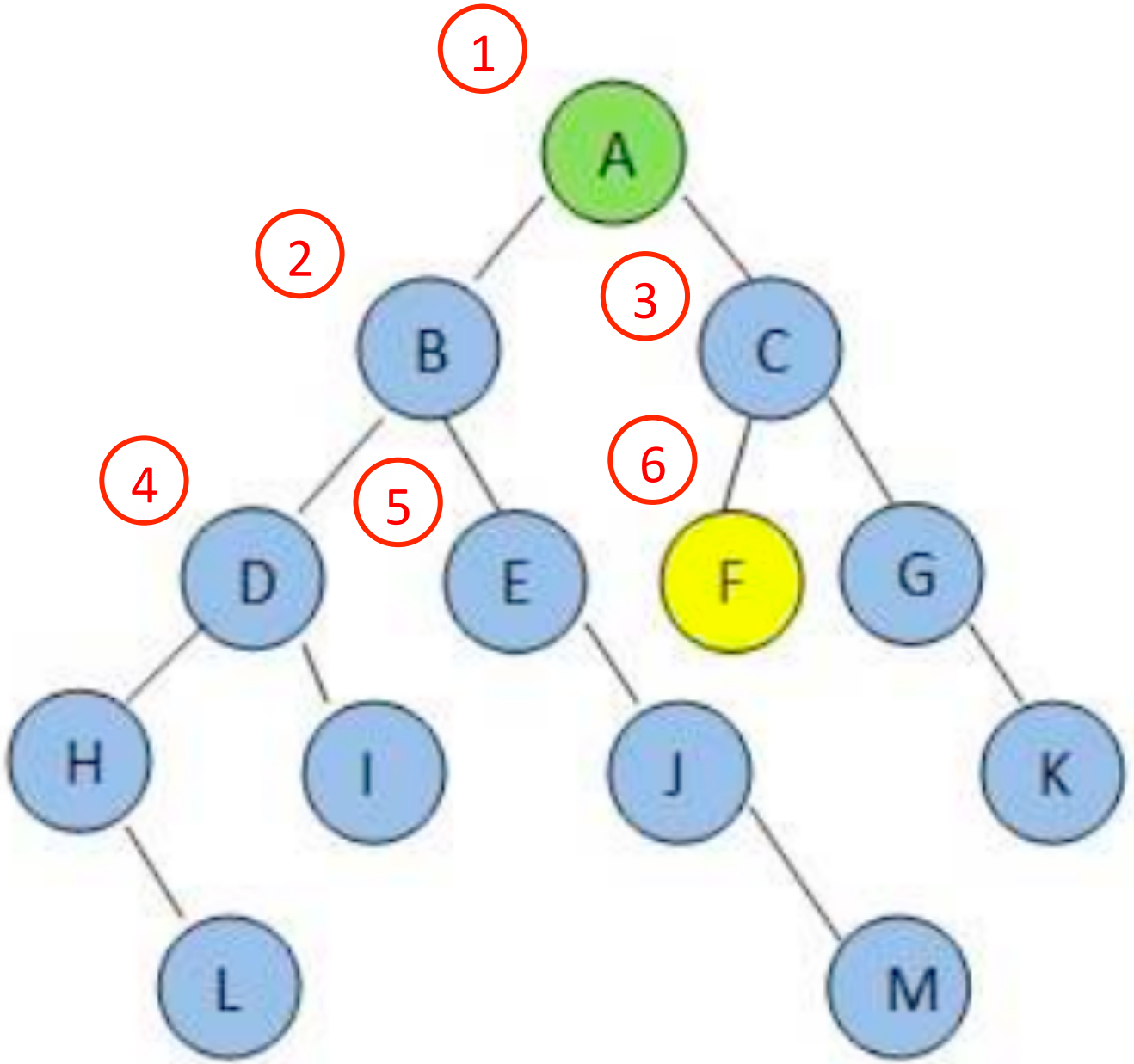
- Search "deeper" in the graph whenever possible
- Start at some vertex v
- After visiting vertex v, the next vertex to be explored is the first unvisited neighbor of v
- If v has no neighbor or if all its neighbors have explored, backtrack to the vertex from which we reached v
- Corresponds to adventurous web browsing: always click the first unvisited link available. Click "back" when you hit a dead-end.


1
A
2
B
9
C
3
6
10
D
E
F
G
4
5
7
H
I
J
K
5
8
L
M

Breadth-First Search

Idea:

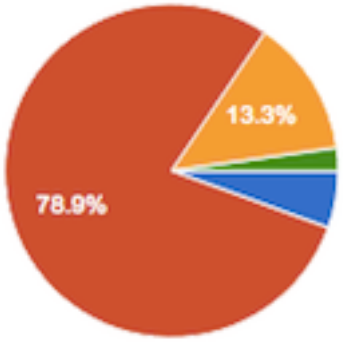
- Explore graph layers by layers
- Start at some vertex v
- Then explore all the neighbors of v
- Then explore all the unvisited neighbors of the neighbors of v
- Then explore all the unvisited neighbors of the neighbors of the neighbors of v
- until no more unvisited vertices remain


1
A
2
B
3
C
4
D
5
E
6
F
G
H
I
J
K
L
M

You read the following statement in a Java program that compiles and executes:

```
submarine.dive(depth);
```

- depth must be an int.
- dive must be a method.
- submarine must be the name of a class.
- submarine must be a method.



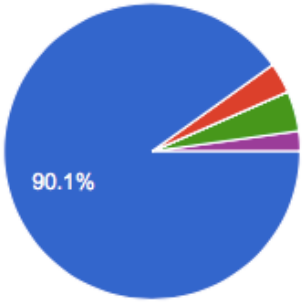
depth must be an int.	**5**	5.6%
dive must be a method.	**71**	78.9%
submarine must be the name of a class.	**12**	13.3%
submarine must be a method.	**2**	2.2%

Consider the following program:

```
public class MyClass{
   public MyClass() { /*code*/ };
   // more code...
}
```

How would you instanciate MyClass?

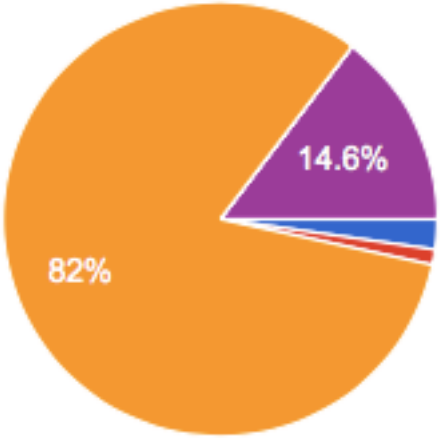
- `MyClass mc = new MyClass();`
- `MyClass mc = MyClass();`
- `MyClass mc = new MyClass;`
- `MyClass mc = new MyClass;`
- It can't be done. The constructor of MyClass should be defined as: `public void MyClass(){/*code*/}`



MyClass mc = new MyClass();	**82**	90.1%
MyClass mc = MyClass();	**3**	3.3%
MyClass mc = MyClass;	**0**	0%
MyClass mc = new MyClass;	**4**	4.4%
It can't be done. The constructor of MyClass should be defined as: public void MyClass(){/*code*/}	**2**	2.2%

You want to initialize all of the elements of a double array a to the same value equal to 1.5. What could y ou write? Assume that the array has been correctly initialized.

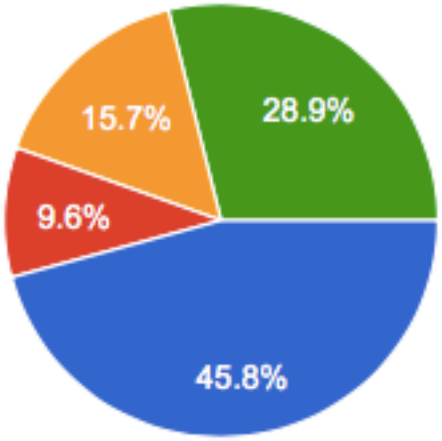
- `for(int i=1; i<a.length; i++) a[i] = 1.5;`
- `for(int i=0; i<=a.length; i++) a[i] = 1.5;`
- `for(int i=0; i<a.length; i++) a[i] = 1.5;`
- `for(int i=0; i<a.length+1; i++) a[i] = 1.5;`
- `for(int i=0; i<a.length-1; i++) a[i] = 1.5;`



for(int i=1; i<a.length; i++) a[i] = 1.5;	**2**	2.2%
for(int i=0; i<=a.length; i++) a[i] = 1.5;	**1**	1.1%
for(int i=0; i<a.length; i++) a[i] = 1.5;	**73**	82%
for(int i=0; i<a.length+1; i++) a[i] = 1.5;	**0**	0%
for(int i=0; i<a.length-1; i++) a[i] = 1.5;	**13**	14.6%

T(n) = T(n-1) + O(n) is a recurrence for the running time of?

- Insertion sort
- Merge sort
- Quick sort
- Bubble sort



Insertion sort	**38**	45.8%
Merge sort	**8**	9.6%
Quick sort	**13**	15.7%
Bubble sort	**24**	28.9%

InsertionSort

```
for i ← 1 to length(A) - 1
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
```

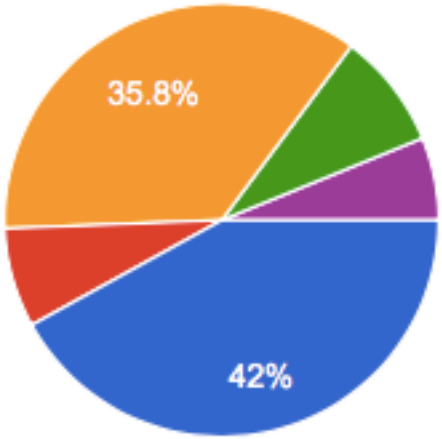
Example:

3 7 4 9 5 2 6 1
3 7 4 9 5 2 6 1
3 7 4 9 5 2 6 1
3 4 7 9 5 2 6 1
3 4 7 9 5 2 6 1
3 4 5 7 9 2 6 1
2 3 4 5 7 9 6 1
2 3 4 5 6 7 9 1
1 2 3 4 5 6 7 9

http://en.wikipedia.org/wiki/Insertion_sort

The probability of team A winning any game is 1/3. Team A plays team B in a tournament. If either team wins two games in a row, that team is declared the winner. At most three games are played in the tournament and, if no team has won the tournament at the end of three games, the tournament is declared a draw. What is the expected number of games in the tournament?

- 3
- 19/9
- 22/9
- 25/9
- 61/27



3	**34**	42%
19/9	**6**	7.4%
22/9	**29**	35.8%
25/9	**7**	8.6%
61/27	**5**	6.2%

Expectation value

$$\langle f(x) \rangle = E[f(x)] = \sum_x f(x) \cdot p(x)$$

- x is a random variable
- f(x) is a function over x
- P(x) probability of x

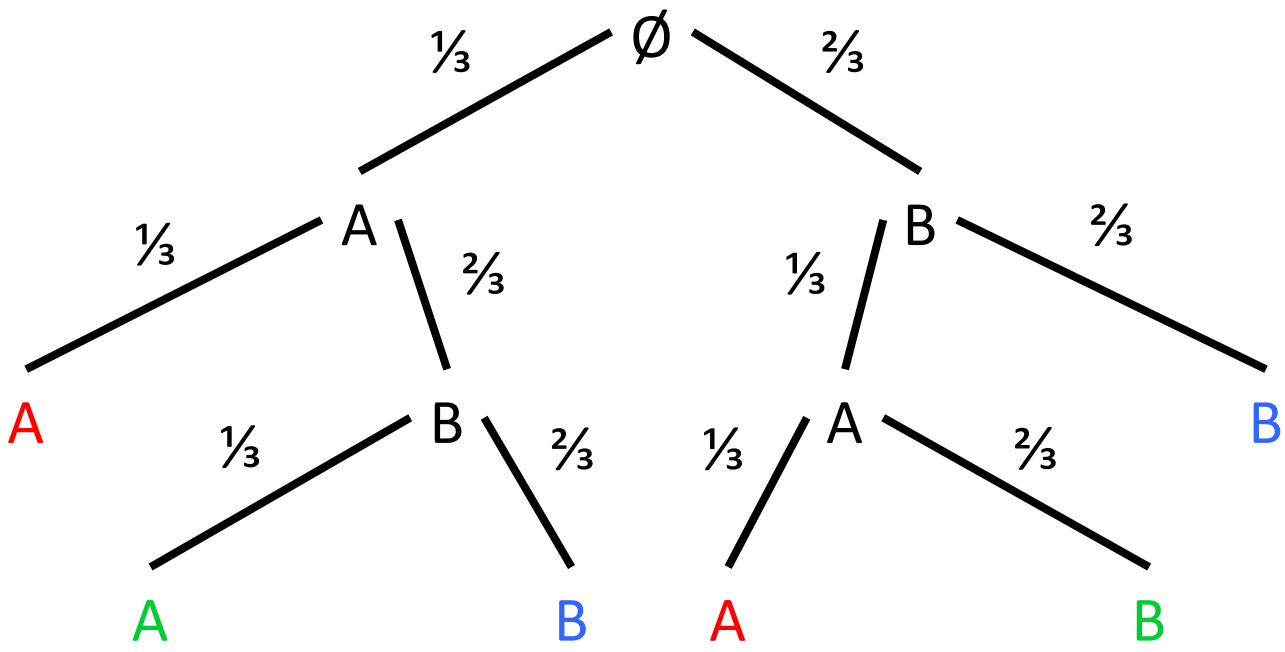
Example: Expected value of a fair 6-sided die roll?

- Each side has a probability of 1/6
- The value of each is the number associated with

$$\begin{aligned}
\langle f(x) \rangle &= f(1) \cdot P(1) + f(2) \cdot P(2) + f(3) \cdot P(3) + f(4) \cdot P(4) + f(5) \cdot P(5) + f(6) \cdot P(6) \\
&= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} + \\
&= 3.5
\end{aligned}$$

Solution

What are the possible outcomes?



A wins
B wins
Draw

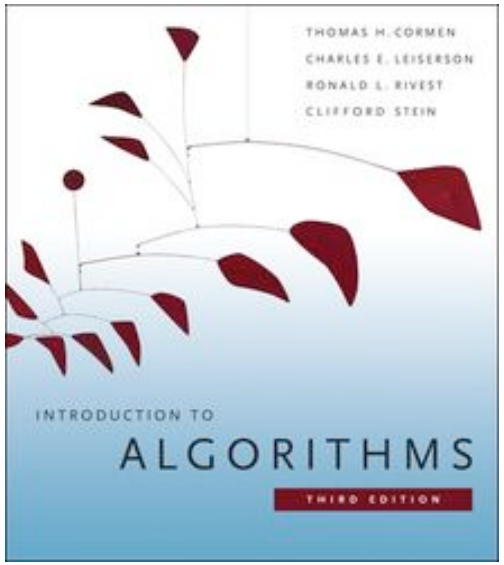
$$E = l(AA)\cdot P(AA) + l(BAA)\cdot P(BAA) + l(BB)\cdot P(BB) + l(ABB)\cdot P(ABB) + l(ABA)\cdot P(ABA) + l(BAB)\cdot P(BAB)$$

$$= 2\cdot\frac{1}{9} + 3\cdot\frac{2}{27} + 2\cdot\frac{4}{9} + 3\cdot\frac{4}{27} + 3\cdot\frac{2}{27} + 3\cdot\frac{4}{27}$$

$$= \frac{22}{9}$$

Review

[CLRS2009] Cormen, Leiserson, Rivest, & Stein, *Introduction to Algorithms*. (available as E-book)



Review Appendix A & C 1-4.