BFS_SP(G,s) // Finds shortest paths from s to all other nodes in unweighted graph G

1. For all v, v.dist = \infty.
2. s.dist = 0
3. Q.enqueue(s)
4. while (Q not empty)
5. v = Q.dequeue()
6. for each neighbor u of v
7. if (u.dist = \infty)
8. u.dist = v.dist + 1
9. Q.enqueue(u)

Q: s d d b e y
One issue with BFS:

- Have to search every node of every level before finding x.

Ideas:
- Search as far down as possible. Then go back up.

Depth-First Search – pre-order traversal on tree (parent first)

Recursive function
// Searches G starting at s for x (x must be accessible from s)

DFS_main(G, s, x) {
    color all vertices white // not searched
    return DFS(G, s, x)
}

DFS(G, v, x) {
    color v gray // searched
    if v == x return true

    for each neighbor u of v:
        if u.color == white
            DFS(G, u, x)
}

ex: DFS(G, s, h)
Run Time?

Still $O(1V^1 + 1E)$
- every vertex colored white exactly once
- every vertex colored from white to gray exactly once
- for loop executed once for every edge

Total: $O(21V + 1E)$ = $O(1V^1 + 1E)$
1) 2 eggs, 100-story building

Find: highest floor to drop an egg w/o breaking.

Can break both eggs in process.

Solution:

Split floors into sets of size 10.

If break at 10
  search linearly from 1-9

Else check 20
  if break at 20
    search linearly from 11-19

\[ \approx O(2^{m}) \text{ checks} \]

2) 4 ppl need to cross bridge
4) Must use flashlight to cross (1 flashlight)
3) At most 2 ppl on bridge
2) Different times to cross: 1 min, 2 min, 7 min, 10 min

Fastest time to cross?

Put 7+10 together (Do not use 1 min as escort).
1 + 2 \Rightarrow 2 \text{ min} \\
1 \text{ goes back} \Rightarrow 1 \text{ min} \\
7 + 10 \Rightarrow 10 \text{ min} \\
2 \text{ goes back} \Rightarrow 2 \text{ min} \\
1 + 2 \Rightarrow 2 \text{ min} \\
17 \text{ min}

Find heaviest stone using fewest \# comparisons?

(ABC)(DEF)

\[
\begin{array}{c}
ABC \quad 1 \\
\mid \\
(A)(B) \quad 1 \\
\mid \\
A \quad 1 \\
\mid \\
(A) \quad 1 \\
\mid \\
(B) \quad 1 \\
\ \quad (D) \quad (E)
\end{array}
\]

Reverse "sample" using constant space.

Just swap first/last, second/second-to-last.

(2) Now reverse "This is a sample" to "sample a is This" using constant space.

Reverse whole string: elpmas a si sihT

(1) Reverse whole string: elpmas a si sihT

"word: sample a is This"