BFS (tree/graph之外)

```
#### BFS @ Tree ####
bfs = collections.deque([root])
while bfs:
    sz = len(bfs)
    for _ in range(sz):
        cur = bfs.popleft()
        if cur.left: bfs.append(cur.left)
        if cur.right: bfs.append(cur.right)

#### BFS @ Graph ####
# => pushed => bfs_queue => poped =>
# pushed is the entry door, it records whatever went into the queue
# poped is the exit door, it records whatever left the queue
bfs = collections.deque([root])
pushed = set(bfs) # poped = set()
while bfs:
    sz = len(bfs)
    for _ in range(sz):
        cur = bfs.popleft() # poped.add(cur)
        for n in cur.neighbors:
            if n not in pushed:
                bfs.append(n)
                pushed.add(n)
```

BFS + Queue

01 Matrix (distance to 0) `BFS [all 0 pos]`
Walls and Gates (distance to gates/0) `BFS [all 0 pos]`
Remove Invalid Parentheses `BFS [initial string]` `)((((((`
Sliding Puzzle `BFS [initial board]`
Open the Wheel Lock `BFS ["0000"]`
Cut Off Trees for Golf Event `BFS [start]`
Word Ladder I (hasPath) `BFS [start]`
Word Ladder II (allPath) `BFS [start]`

multi-BFS

Pacific Atlantic Water Flow `2-BFS: [Pacific] [Atlantic]` `hitSum`
Shortest Distance from All Buildings `K-BFS [each_building]` `distSum, hitSum`

BFS + Heap

Swim in Rising Water `BFS heap[(g00, 0, 0)]`
Trapping Rain Water I (1D) `BFS: heap[boundary]` `max_visited`
Trapping Rain Water II (2D) `BFS: heap[boundary]` `max_visited`

smart

Bus Routes `BFS [stops reachable by 0 bus]`
Shortest Path Visiting All Nodes `state: (cur, visited)` `BFS [all starting nodes]`

todo

Shortest Path to Get All Keys