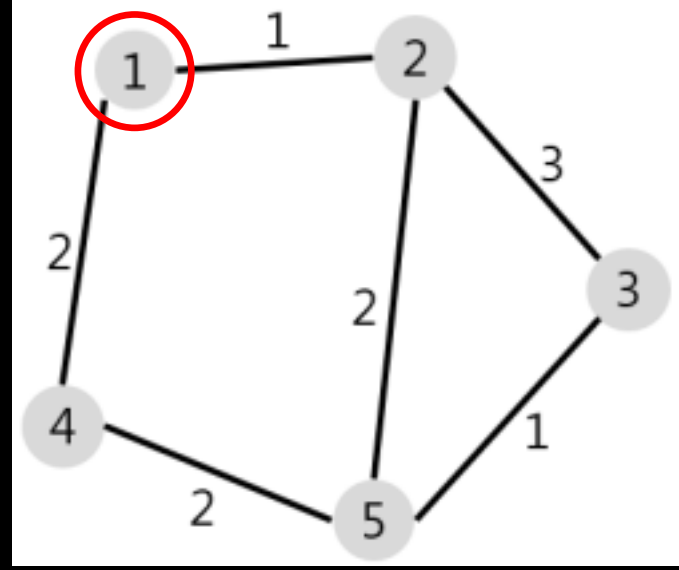
```python
from collections import deque
def bfs(graph, start):

    q = deque([(start,0)]) # (node, depth)
    visited = {start:0} # {node:depth}

    while q:
        cur_node, cur_depth = q.popleft()

        for next_node in graph[cur_node]:
            if next_node not in visited:
                next_depth = cur_depth + 1
                visited[next_node] = next_depth
                q.append((next_node, next_depth))

    return visited
```



{1: 0, 2: 1, 4: 1, 3: 2, 5: 2}

```python
import heapq
def dijkstra(graph, start):
    pq=[]
    heapq.heappush(pq, (0,start)) # (weight, node)
    visited={} # {node:cost}

    while pq:
        cur_cost, cur_node = heapq.heappop(pq)

        if cur_node not in visited:
            visited[cur_node] = cur_cost
            for next_node in graph[cur_node]:
                next_cost = cur_cost + graph[cur_node][next_node]
                heapq.heappush(pq, (next_cost, next_node))

    return visited
```

{1: 0, 2: 1, 4: 2, 5: 3, 3: 4}