☆ franks BFS code (To find degrees between 2 super Heroes)

- BFS in spark →
repeat : flat map → expand Gray frontier → reducebyKey merge updates → check accumulator → stop when Target found → one BFS level. (every iteration)

Pseudo Code →

```
start, target
hit Counter = sc.accumulator(0)

def convertToBfs(line):
        return (heroid, (connections, distance, color))
def create StartingRdd() :
            return file. map(convert to Bfs)
def bfsMap(node):
     if Target found → hitCounter → 1
   return (Carrid, (connection, distance, color)
def bfs Reduce(data1, data2)
    return (edges, distance, color)

for in range(0, 10):
   mapped = rdd. flatMap(bfsmap)
   if (hitCounter.value >0):
         break
   rdd = mapped. reducedByKey(bfsReduce)
```

Core node → (character id, (edges_list, distance, color))

E • edges list = neighbour (non-empty only in original record)
D • distance = 0(start), 9999 (unknown), or actual distance
C • Color = white(unseen) < GRAY (Frontier) < Black (done)

☆ Mapper (bfsMap) –

- if Node is GRAY →
  - emit (neighbour, ([], distance + 1, GRAY)) for each neighbour (discovery Tuple – no edges).
  - emit the original Node as BLACK : (id, (edges, distance, BLACK)
  - if neighbour == Target → hitcounter.add(1)

- if Node is WHITE / BLACK : re-emit the node unchanged.

why [] for edges? To avoid copying neighbour list for every discovery, Reducer restores edges from the original record.

☆ Reducer (bfsReduce) – more logic
when you get multiple Tuples for same id :
- Edges : keep non-empty lists (original edges wins)
- Distance : keep (min(...)) (9999 acts as ∞)
- Color : pick the darkest : Black > GRAY > WHITE

☆ Loop & stop condition →

```
mapped.count()
if hitcounter.value > 0 : break
rdd = map.ReduceByKey (bfsReduce)
```

→ This is required because worker side hit counter.add() actually execute.

- placing if before reduceByKey, avoids unnecessary Shuffle, (performance wins)

__/__/____

So,

- This is iterative flatmap expainson of GRAY nodes emmitting discovery Tuples [ ] with edges.
- Then ReduceByKey to merge & choose min(distance) & 'darkest' colour.
- used an accumlator to signal discovry of Target to driver;
- degree = iteration index. (or lookup)