B.D.A Assignment- 2

Nishant Kumar, 2022326

Summary Table:

Results vs. Ground Truth

	Metric	Ground Truth	Computed	Notes on Difference
0	Nodes	7,115	7,115	Match. Exact equality.
1	Edges	103,689	103,689	Match. Exact equality.
2	Largest WCC (nodes)	7,066 (0.993)	7,066	Match. Exact equality.
3	Largest WCC (edges)	103,663 (1.000)	103,663	Match. Exact equality.
4	Largest SCC (nodes)	1,300 (0.183)	1,300	Match. Exact equality.
5	Largest SCC (edges)	39,456 (0.381)	39,456	Match. Exact equality.
6	Average clustering coefficient	0.1409	0.1409	Match. Exact equality.
7	Number of triangles	608,389	608,389	Match. Exact equality.
8	Fraction of closed triangles	0.04564	0.04183	Close match. Minor sampling/rounding gap.
9	Diameter	7	7	Match. Exact equality.
10	90% effective diameter	3.8	4.0	Close match. Minor sampling/rounding gap.

1) Algorithm design choices

- **Load & clean:** parse to edges_df(src,dst); build vertices_df(id) from distinct sources/dests. Simple, reusable base tables.

- **WCC (label propagation):** Symmetrize once (src↔dst, distinct), init label=id, then per round set each node's label to the min of itself and neighbors; stop after stability. Chosen for clarity and fast convergence on this graph.

- **SCC (Kosaraju on driver):** Build forward/reverse adjacency with collect_set, run DFS on G^T (finishing order) then DFS on G (component labels), push (vid, component) back to Spark. Chosen for correctness and simplicity at roughly 7k nodes.

- **Triangles & clustering (undirected):** Canonicalize to (u,v) with u<v, drop loops/dupes. For each (u,v), try $w \in N^+(u)$ with w>v; keep if (v,w) exists → each triangle counted once.
  Local clustering per node: tri_v / (deg*(deg-1)/2) if deg≥2, else 0. **Average clustering** = mean of locals. **Fraction of closed triangles** = triangles / Σ (deg*(deg-1)/2).

- **Diameter & 90% effective diameter (on largest WCC, undirected):**
  Two-sweep diameter (BFS farthest-from-farthest over several trials).
  Effective-90 by sampling BFS sources, pooling all dist>0, taking 90th percentile.

2) Iterative implementation in Spark

- **WCC loop (label propagation):**

  1. edges_undirected ⋈ labels to bring neighbor labels
  2. groupBy(id).min(label) → min_neighbor
  3. labels = least(old_label, min_neighbor)
  4. detect “no change” → stop. Hot DataFrames persisted (MEMORY_AND_DISK).

- **Triangles:**

    1. Build canonical E_canon(u<v)
    2. groupBy(u).collect_set(v) → $N^+(u)$
    3. explode candidates (u,v,w) with w>v
    4. inner join to confirm (v,w) exists → triangles
    5. per-vertex triangle counts → local clustering → average and global fraction.

- **SCC (driver Kosaraju):**

    1. collect_set to Python dicts (forward & reverse)
    2. DFS on reverse (finishing order)
    3. DFS on forward (label components)
    4. create Spark DF (vid, component); compute sizes; filter edges inside largest SCC with broadcast.

- **Diameter / Effective-90 (driver on largest WCC):**

    1. Filter edges to largest WCC (broadcast vertex set)
    2. Build undirected adjacency on driver
    3. Two-sweep trials for diameter; sampled BFS sources for 90th percentile.

3) Performance and discrepancies

- **Performance tactics**

    - **Caching/persisting:** edges_df, undirected edges, degree tables, $N^+(u)$, and local-clustering DF cached to avoid rescans.

- **Broadcast joins:** Used to quickly keep only edges inside largest WCC/SCC.
- **Search pruning:** Canonical (u<v) and w>v massively reduces triangle candidates.
- **Planner knobs:** Fixed shuffle partitions; AQE off for predictability while developing. Defaults are fine at this scale.
- **Driver work (SCC, diameter):** Safe for roughly 7k nodes; would switch to fully distributed methods / heavier sampling on larger graphs.

- **Where results can differ & why**
  - **Fraction of closed triangles:** Close match. I divide unique **undirected** triangles by Σ (deg*(deg−1)/2) from the **undirected** graph; some references use a different denominator or directed wedges, which yields slightly higher values.

  - **90% effective diameter:** Higher than 3.8. I compute it on the **undirected** largest WCC by pooling distances from sampled BFS sources and taking the overall 90th percentile; the 3.8 figure likely reflects a different convention (e.g., directed paths or a different percentile aggregation).

  - **All other metrics (counts, WCC/SCC sizes, triangle total, avg clustering, diameter):** Match. Implementations and preprocessing align with the dataset's ground-truth definitions.