END-SEMESTER PREPARATION · PART 3 OF 3

# Big Data Analytics — Solved Numericals

All 10 numerical questions from the question bank, solved step by step with every calculation shown. Colours match Parts 1 and 2: each question uses its module's colour.

| Q | Module | Topic | Final answer |
|---|---|---|---|
| Q59 | 3 | Matrix multiplication with MapReduce | P = [[3, 8], [3, 7], [7, 18]] |
| Q60 | 3 | Grouping & aggregation (GROUP BY A, B; SUM C) | 8 groups, e.g. (1,2) → 9, (2,3) → 11 |
| Q63 | 3 | Union with MapReduce | 10 distinct tuples |
| Q79 | 4 | Flajolet–Martin, h(x) = (6x+1) mod 5 | R = 2 → about 4 distinct |
| Q80 | 4 | Flajolet–Martin with two hash functions | (a) 16 · (b) 8 |
| Q84 (B) | 4 | Bloom filter false-positive rate | ≈ 0.0118 (1.18%) |
| Q101 | 5 | PCY candidate pairs | 15 candidates → 12 frequent pairs |
| Q120 | 6 | PageRank, 3 iterations | A 1.054, B 0.729, C 1.217 (ranking C > A > B) |
| Q124 | 6 | HITS, k = 3 | Best authority: node 4 · best hub: node 1 |
| Q125 | 6 | PageRank, 6 pages, iteration 2 | A = F = 0.5166, B = C = 0.4847, D = E = 0.2722 |

## Q59 · Matrix multiplication using MapReduce (10 marks)

Evaluate matrix multiplication by MapReduce for $\begin{pmatrix}1&2\\2&1\\3&4\end{pmatrix}\times\begin{pmatrix}1&2\\1&3\end{pmatrix}$ i.e. M (3×2) × N (2×2).

**HOW TO READ THIS QUESTION**

The second matrix is 2×2, so this is a **matrix × matrix** product. It is the same as running matrix–vector multiplication twice, once for each column of N ($v_1$ = [1, 1], $v_2$ = [2, 3]). Both views are shown below; write whichever your teacher used.

**FORMULA**

$p_{ik} = \Sigma_j\, m_{ij} \cdot n_{jk}$ (i = row of M, j = shared index, k = column of N)

**1** **Write the input as (matrix, row, column, value) records.**
M: (M,1,1,1) (M,1,2,2) (M,2,1,2) (M,2,2,1) (M,3,1,3) (M,3,2,4)
N: (N,1,1,1) (N,1,2,2) (N,2,1,1) (N,2,2,3)

**2** **Job 1 — Map: use the shared index j as the key.** For $m_{ij}$ emit (j, (M, i, $m_{ij}$)); for $n_{jk}$ emit (j, (N, k, $n_{jk}$)).

| Key j | Values from M (i, $m_{ij}$) | Values from N (k, $n_{jk}$) |
|---|---|---|
| j = 1 | (M,1,1) (M,2,2) (M,3,3) | (N,1,1) (N,2,2) |
| j = 2 | (M,1,2) (M,2,1) (M,3,4) | (N,1,1) (N,2,3) |

**3** **Job 1 — Reduce: multiply every M value with every N value for the same j** and emit ((i, k), $m_{ij}$·$n_{jk}$).

| (i, k) | from j = 1 | from j = 2 |
|---|---|---|
| (1,1) | 1 × 1 = 1 | 2 × 1 = 2 |
| (1,2) | 1 × 2 = 2 | 2 × 3 = 6 |
| (2,1) | 2 × 1 = 2 | 1 × 1 = 1 |
| (2,2) | 2 × 2 = 4 | 1 × 3 = 3 |
| (3,1) | 3 × 1 = 3 | 4 × 1 = 4 |
| (3,2) | 3 × 2 = 6 | 4 × 3 = 12 |

**4** **Job 2 — Map: identity. Reduce: sum the products for each key (i, k).**
`p11 = 1 + 2 = 3` · `p12 = 2 + 6 = 8` · `p21 = 2 + 1 = 3` · `p22 = 4 + 3 = 7` · `p31 = 3 + 4 = 7` · `p32 = 6 + 12 = 18`

**5** **Matrix–vector view (one column at a time, v in memory).** Map emits (i, $m_{ij}$·$v_j$); reduce sums per row i.
$v_1$ = [1, 1]: row 1 → 1·1 + 2·1 = 3; row 2 → 2·1 + 1·1 = 3; row 3 → 3·1 + 4·1 = 7 → x = [3, 3, 7]
$v_2$ = [2, 3]: row 1 → 1·2 + 2·3 = 8; row 2 → 2·2 + 1·3 = 7; row 3 → 3·2 + 4·3 = 18 → x = [8, 7, 18]

**ANSWER**

$P = M \times N = \begin{pmatrix}3&8\\3&7\\7&18\end{pmatrix}$ (the two columns are exactly the two matrix–vector results [3, 3, 7] and [8, 7, 18]).

**PSEUDO CODE TO WRITE**

`map(M,i,j,m): emit(j,(M,i,m)) · map(N,j,k,n): emit(j,(N,k,n)) · reduce1(j,L): for (M,i,m) in L, for (N,k,n) in L: emit((i,k), m·n) · reduce2((i,k),vals): emit((i,k), sum(vals))`

## Q60 · Grouping and aggregation using MapReduce (10 marks)

Two map workers each hold File 1 and File 2 of a relation R(A, B, C, D). Compute $\gamma_{A, B, SUM(C)}(R)$, i.e. `SELECT A, B, SUM(C) FROM R GROUP BY A, B`.

**ASSUMPTION**

The question's figure gives only the tables. The standard example (also used in the reference article) groups by (A, B) and sums C, so that is solved here. For any other choice, keep the same steps and change the key/value columns.

**Map worker 1**

| File | A | B | C | D |
|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 1 |
| 1 | 2 | 2 | 3 | 2 |
| 1 | 1 | 2 | 1 | 3 |
| 2 | 4 | 2 | 1 | 3 |
| 2 | 6 | 8 | 4 | 4 |
| 2 | 3 | 2 | 2 | 4 |

**Map worker 2**

| File | A | B | C | D |
|---|---|---|---|---|
| 1 | 1 | 2 | 5 | 2 |
| 1 | 2 | 3 | 2 | 4 |
| 1 | 1 | 3 | 1 | 3 |
| 2 | 3 | 2 | 5 | 2 |
| 2 | 2 | 3 | 9 | 2 |
| 2 | 3 | 4 | 2 | 1 |

1. **Map:** for every tuple emit **key = (A, B)**, **value = C**. Column D is not needed and is dropped.
   MW1 emits: ((1,2),3) ((2,2),3) ((1,2),1) ((4,2),1) ((6,8),4) ((3,2),2)
   MW2 emits: ((1,2),5) ((2,3),2) ((1,3),1) ((3,2),5) ((2,3),9) ((3,4),2)
2. **Combiner (optional, allowed because SUM is associative):** each worker pre-adds its own values.
   MW1 → (1,2):3+1 = 4 · (2,2):3 · (4,2):1 · (6,8):4 · (3,2):2 | MW2 → (1,2):5 · (2,3):2+9 = 11 · (1,3):1 · (3,2):5 · (3,4):2
3. **Shuffle & sort — group by key**, then **Reduce: sum the list.**

| Key (A, B) | Values received (without combiner) | Reduce: SUM(C) |
|---|---|---|
| (1,2) | [3, 1, 5] | **3 + 1 + 5 = 9** |
| (1,3) | [1] | **1** |
| (2,2) | [3] | **3** |
| (2,3) | [2, 9] | **2 + 9 = 11** |
| (3,2) | [2, 5] | **2 + 5 = 7** |
| (3,4) | [2] | **2** |
| (4,2) | [1] | **1** |
| (6,8) | [4] | **4** |

With the combiner, (1,2) receives [4, 5] → 9 and (2,3) receives [11] → 11: the same result with fewer records shuffled (12 → 10).

**ANSWER — OUTPUT OF $\Gamma_{A,B,SUM(C)}$**

(1,2,9) · (1,3,1) · (2,2,3) · (2,3,11) · (3,2,7) · (3,4,2) · (4,2,1) · (6,8,4)

**PSEUDO CODE**

`map(t): emit((t.A, t.B), t.C)` · `reduce(key, values): emit(key, sum(values))` — for AVG emit (sum, count) pairs so a combiner stays valid.

## Q63 · Union using MapReduce (10 marks)

Map worker 1 holds Table 1 {(1,2), (2,3), (5,6)} and Table 2 {(2,3), (4,4), (6,1)}; Map worker 2 holds Table 1 {(6,1), (6,3), (7,6)} and Table 2 {(9,8), (3,3), (0,1)}, all with schema (A, B). Find Table1 ∪ Table2.

**RULE**

**Map:** for every tuple t of either table emit (t, t). **Reduce:** for each key t output t **once** (one or two values may arrive) — duplicates disappear.

1. **Map outputs**
   MW1: ((1,2),(1,2)) ((2,3),(2,3)) ((5,6),(5,6)) ((2,3),(2,3)) ((4,4),(4,4)) ((6,1),(6,1))
   MW2: ((6,1),(6,1)) ((6,3),(6,3)) ((7,6),(7,6)) ((9,8),(9,8)) ((3,3),(3,3)) ((0,1),(0,1)) → 12 records

**2** **Shuffle & group by key, then reduce**

| Key t | Values received | Came from | Reduce output |
|---|---|---|---|
| (0,1) | [(0,1)] | MW2 Table 2 | (0,1) |
| (1,2) | [(1,2)] | MW1 Table 1 | (1,2) |
| (2,3) | [(2,3), (2,3)] | MW1 Table 1 and Table 2 | **(2,3) — once** |
| (3,3) | [(3,3)] | MW2 Table 2 | (3,3) |
| (4,4) | [(4,4)] | MW1 Table 2 | (4,4) |
| (5,6) | [(5,6)] | MW1 Table 1 | (5,6) |
| (6,1) | [(6,1), (6,1)] | MW1 Table 2 and MW2 Table 1 | **(6,1) — once** |
| (6,3) | [(6,3)] | MW2 Table 1 | (6,3) |
| (7,6) | [(7,6)] | MW2 Table 1 | (7,6) |
| (9,8) | [(9,8)] | MW2 Table 2 | (9,8) |

**ANSWER**
Table1 ∪ Table2 = {(0,1), (1,2), (2,3), (3,3), (4,4), (5,6), (6,1), (6,3), (7,6), (9,8)} — **10 distinct tuples** from 12 input tuples; the duplicates (2,3) and (6,1) appear once.

**CONTRAST FOR FULL MARKS**
Intersection on the same data would emit (t, table-tag) and output only keys with both tags → {(2,3), (6,1)}.

## Q79 · Flajolet–Martin (10 marks)

Stream = 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1 · h(x) = (6x + 1) mod 5. Estimate the number of distinct elements.

**METHOD**
(1) hash each element · (2) write h(x) in binary · (3) r = number of trailing zeros · (4) R = max r · (5) estimate = $2^R$. Hash values lie in 0–4, so 3 bits are enough.

**1** **Hash every element of the stream** (repeats give the same hash, so they never change R).

| Position | x | 6x + 1 | h(x) = (6x+1) mod 5 | Binary (3 bits) | Trailing zeros r |
|---|---|---|---|---|---|
| 1 | 1 | 7 | 2 | 010 | 1 |
| 2 | 3 | 19 | 4 | 100 | **2** |
| 3 | 2 | 13 | 3 | 011 | 0 |
| 4 | 1 | 7 | 2 | 010 | 1 |
| 5 | 2 | 13 | 3 | 011 | 0 |
| 6 | 3 | 19 | 4 | 100 | **2** |
| 7 | 4 | 25 | 0 | 000 | 0* |
| 8 | 3 | 19 | 4 | 100 | **2** |
| 9 | 1 | 7 | 2 | 010 | 1 |
| 10 | 2 | 13 | 3 | 011 | 0 |
| 11 | 3 | 19 | 4 | 100 | **2** |
| 12 | 1 | 7 | 2 | 010 | 1 |

* h = 0 is the all-zero string with no '1' bit; by the usual exam convention it is given r = 0 (ignored). If your teacher counts it as 3 trailing zeros, R becomes 3 and the estimate 8.

**2** **R = max(1, 2, 0, 1, 0, 2, 0, 2, 1, 0, 2, 1) = 2.**

**3** **Estimate = $2^R = 2^2 = 4$.** Check: the distinct values are {1, 2, 3, 4} → 4 exactly.

**ANSWER**
R = 2, so the estimated number of distinct elements is **$2^2 = 4$** (matches the true count 1, 2, 3, 4).

## Q80 · Flajolet–Martin with two hash functions (10 marks)

Stream = 4, 2, 5, 9, 1, 6, 3, 7 · (a) h(x) = (3x + 1) mod 32 · (b) h(x) = (x + 6) mod 32. Values are 0–31, so use 5-bit binary.

**(a) h(x) = (3x + 1) mod 32**

| x | 3x+1 | h | binary | r |
|---|---|---|---|---|
| 4 | 13 | 13 | 01101 | 0 |
| 2 | 7 | 7 | 00111 | 0 |
| 5 | 16 | 16 | 10000 | **4** |
| 9 | 28 | 28 | 11100 | 2 |
| 1 | 4 | 4 | 00100 | 2 |
| 6 | 19 | 19 | 10011 | 0 |
| 3 | 10 | 10 | 01010 | 1 |
| 7 | 22 | 22 | 10110 | 1 |

R = max(0,0,4,2,2,0,1,1) = **4** → estimate = $2^4$ = **16**

**(b) h(x) = (x + 6) mod 32**

| x | x+6 | h | binary | r |
|---|---|---|---|---|
| 4 | 10 | 10 | 01010 | 1 |
| 2 | 8 | 8 | 01000 | **3** |
| 5 | 11 | 11 | 01011 | 0 |
| 9 | 15 | 15 | 01111 | 0 |
| 1 | 7 | 7 | 00111 | 0 |
| 6 | 12 | 12 | 01100 | 2 |
| 3 | 9 | 9 | 01001 | 0 |
| 7 | 13 | 13 | 01101 | 0 |

R = max(1,3,0,0,0,2,0,0) = **3** → estimate = $2^3$ = **8**

**Compare with the truth:** all 8 stream values are different, so the true count is 8. Hash (b) is exact; hash (a) overestimates because one value (h(5) = 16 = 10000) happened to have 4 trailing zeros. This is why FM uses many hash functions: the average of these two is (16 + 8)/2 = 12, and with more hashes you take the **average within groups, then the median across groups**.

**ANSWER**
(a) R = 4 → **16 distinct** · (b) R = 3 → **8 distinct** (true value 8). Combining several hash functions reduces the error of a single estimate.

## Q84 (B) · Bloom filter false-positive probability (part of a 10-mark question)

A Bloom filter has m = 1000 cells (bits), stores n = 100 items and uses k = 4 hash functions. Find the false-positive probability.

**FORMULA (LETTERS AS IN THIS QUESTION)**
$P_{fp} = (1 - e^{-kn/m})^k$ [exact form: $(1 - (1 - 1/m)^{kn})^k$]
Note: Part 1 writes the same formula with n = bits and m = items. Only the meaning matters: exponent = (hash functions × items) ÷ bits.

1. **Exponent:** kn/m = (4 × 100) / 1000 = **0.4**
2. **Probability a given bit is still 0:** $e^{-0.4}$ = **0.67032**
3. **Probability a given bit is 1:** 1 − 0.67032 = **0.32968**
4. **All k = 4 bits are 1 by chance:** $0.32968^4 = (0.32968^2)^2 = 0.108689^2$ = **0.011813**

**ANSWER**
$P_{fp}$ ≈ **0.0118 ≈ 1.18%** (the exact form gives 0.01183 — the same to three decimals). About 12 in every 1000 absent items would wrongly be reported as present.

**EXTRA MARK**
Optimal k = (m/n)·ln 2 = 10 × 0.693 ≈ 7 → $P_{fp} = (1 - e^{-0.7})^7 \approx 0.0082$ (0.82%), so 7 hash functions would be slightly better than 4.

## Q101 · PCY algorithm with buckets and MapReduce (10 marks)

Threshold (support) s = 2 · hash h(i, j) = (i × j) mod 10 · transactions: T1 {1,2,3} T2 {2,3,4} T3 {3,4,5} T4 {4,5,6} T5 {1,3,5} T6 {2,4,6} T7 {1,3,4} T8 {2,4,5} T9 {3,5,6} T10 {1,2,4} T11 {2,3,5} T12 {3,4,6}. Find the candidate set.

1. **Pass 1(a) — count single items.**

| Item | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Count | 4 | 6 | 8 | 8 | 6 | 4 |
| Frequent (≥ 2)? | yes | yes | yes | yes | yes | yes |

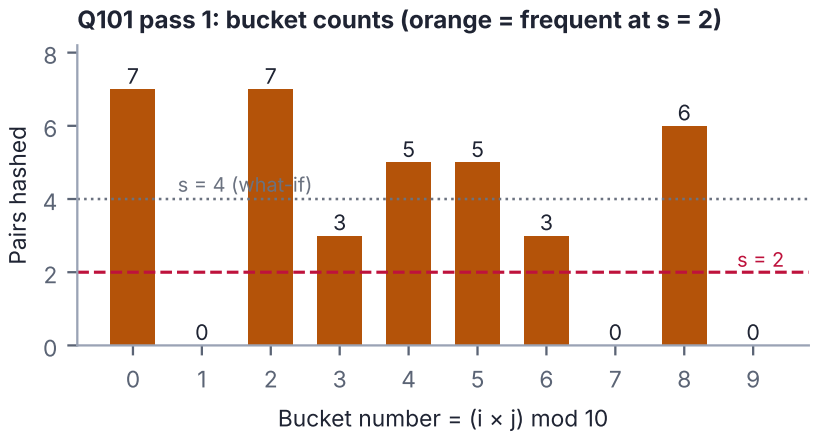
All six items are frequent.

**2** **Pass 1(b) — hash every pair of every transaction** into bucket (i × j) mod 10.

| T | Pairs → bucket | T | Pairs → bucket |
|---|---|---|---|
| T1 | {1,2}→2 · {1,3}→3 · {2,3}→6 | T7 | {1,3}→3 · {1,4}→4 · {3,4}→2 |
| T2 | {2,3}→6 · {2,4}→8 · {3,4}→2 | T8 | {2,4}→8 · {2,5}→0 · {4,5}→0 |
| T3 | {3,4}→2 · {3,5}→5 · {4,5}→0 | T9 | {3,5}→5 · {3,6}→8 · {5,6}→0 |
| T4 | {4,5}→0 · {4,6}→4 · {5,6}→0 | T10 | {1,2}→2 · {1,4}→4 · {2,4}→8 |
| T5 | {1,3}→3 · {1,5}→5 · {3,5}→5 | T11 | {2,3}→6 · {2,5}→0 · {3,5}→5 |
| T6 | {2,4}→8 · {2,6}→2 · {4,6}→4 | T12 | {3,4}→2 · {3,6}→8 · {4,6}→4 |

**3** **Bucket counts** (36 pair occurrences in total):

| Bucket | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Pairs in it | {2,5}{4,5}{5,6} | — | {1,2}{2,6}{3,4} | {1,3} | {1,4}{4,6} | {1,5}{3,5} | {2,3} | — | {2,4}{3,6} | — |
| Count | 7 | 0 | 7 | 3 | 5 | 5 | 3 | 0 | 6 | 0 |
| Frequent (≥ 2) | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |




**Bitmap** (buckets 0–9) = **1 0 1 1 1 1 1 1 0 1 0**

**4** **Candidate set $C_2$:** a pair {i, j} is a candidate if **i is frequent, j is frequent, and h(i, j) is a frequent bucket**. Check all 15 pairs of the 6 frequent items:

| Pair | h | Bucket frequent? | Candidate? | Pair | h | Bucket frequent? | Candidate? |
|---|---|---|---|---|---|---|---|
| {1,2} | 2 | yes | ✓ | {2,6} | 2 | yes | ✓ |
| {1,3} | 3 | yes | ✓ | {3,4} | 2 | yes | ✓ |
| {1,4} | 4 | yes | ✓ | {3,5} | 5 | yes | ✓ |
| {1,5} | 5 | yes | ✓ | {3,6} | 8 | yes | ✓ |
| {1,6} | 6 | yes | ✓ | {4,5} | 0 | yes | ✓ |
| {2,3} | 6 | yes | ✓ | {4,6} | 4 | yes | ✓ |
| {2,4} | 8 | yes | ✓ | {5,6} | 0 | yes | ✓ |
| {2,5} | 0 | yes | ✓ | | | | |

With s = 2 every pair lands in a frequent bucket, so **all 15 pairs are candidates**. Note {1,6} never occurs together, but it is still a candidate because it shares bucket 6 with {2,3} (a collision) — PCY removes such pairs only in pass 2.

**5** **Pass 2 — count the candidates** and keep those with count ≥ 2.

| Pair | {1,2} | {1,3} | {1,4} | {1,5} | {1,6} | {2,3} | {2,4} | {2,5} |
|---|---|---|---|---|---|---|---|---|
| Count | 2 | 3 | 2 | 1 | 0 | 3 | 4 | 2 |
| **Pair** | **{2,6}** | **{3,4}** | **{3,5}** | **{3,6}** | **{4,5}** | **{4,6}** | **{5,6}** | |
| Count | 1 | 4 | 4 | 2 | 3 | 3 | 2 | |

**6** **Using MapReduce.** *Job 1* — Map (each mapper gets a chunk of transactions): emit (item, 1) for every item and (bucket h(i,j), 1) for every pair; Combiner adds locally; Reduce sums → item counts and bucket counts → frequent items + bitmap. *Job 2* — Map (bitmap and frequent items broadcast to every mapper): for each pair in a transaction that passes the PCY test emit (pair, 1); Reduce sums and keeps count ≥ s.

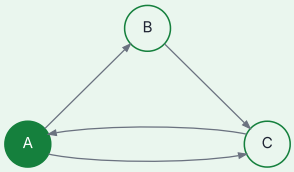
**ANSWER**
**Candidate set $C_2$ (all 15 pairs):** {1,2} {1,3} {1,4} {1,5} {1,6} {2,3} {2,4} {2,5} {2,6} {3,4} {3,5} {3,6} {4,5} {4,6} {5,6}.
**Frequent pairs after pass 2 (count ≥ 2), 12 pairs:** {1,2} {1,3} {1,4} {2,3} {2,4} {2,5} {3,4} {3,5} {3,6} {4,5} {4,6} {5,6}.
Rejected: {1,5}, {1,6}, {2,6}.

**WHY PCY HELPS — SAME DATA WITH S = 4 (WHAT-IF)**

Frequent buckets become 0, 2, 4, 5, 8 (bitmap 1010110010). Pairs hashing to buckets 3 or 6 — {1,3}, {2,3}, {1,6} — are pruned **without counting**, leaving 12 candidates; pass 2 finds {2,4}, {3,4}, {3,5} (count 4 each). Mention this to show you understand the pruning.

## Q120 · PageRank after 3 iterations (10 marks)

Graph: A → B, A → C, B → C, C → A. Use damping β = 0.85 and initial PR = 1 for every page.
Out-links: L(A) = 2, L(B) = 1, L(C) = 1. In-links: A ← C; B ← A; C ← A, B.




**FORMULA USED**

$PR(p) = (1 - \beta) + \beta \cdot \Sigma_{q \to p}\, PR(q)/L(q) = 0.15 + 0.85 \cdot \Sigma\, PR(q)/L(q)$

PR(A) = 0.15 + 0.85·PR(C)/1 · PR(B) = 0.15 + 0.85·PR(A)/2 · PR(C) = 0.15 + 0.85·[PR(A)/2 + PR(B)/1]

**Method 1 — every page uses the previous iteration's values (standard power iteration)**

| Iteration | PR(A) | PR(B) | PR(C) | Sum |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0.15 + 0.85(1) = **1.0000** | 0.15 + 0.85(1/2) = **0.5750** | 0.15 + 0.85(1/2 + 1) = **1.4250** | 3 |
| 2 | 0.15 + 0.85(1.425) = **1.3613** | 0.15 + 0.85(1.0/2) = **0.5750** | 0.15 + 0.85(0.5 + 0.575) = **1.0638** | 3 |
| 3 | **0.15 + 0.85(1.0638) = 1.0542** | **0.15 + 0.85(1.3613/2) = 0.7285** | **0.15 + 0.85(0.6806 + 0.575) = 1.2173** | 3 |

**Method 2 — use a value as soon as it is updated (Gauss–Seidel; common in textbook solutions)**

| Iteration | PR(A) | PR(B) uses new A | PR(C) uses new A, B |
|---|---|---|---|
| 1 | 0.15 + 0.85(1) = 1.0000 | 0.15 + 0.85(1.0/2) = 0.5750 | 0.15 + 0.85(0.5 + 0.575) = 1.0638 |
| 2 | 0.15 + 0.85(1.0638) = 1.0542 | 0.15 + 0.85(1.0542/2) = 0.5980 | 0.15 + 0.85(0.5271 + 0.5980) = 1.1064 |
| 3 | **0.15 + 0.85(1.1064) = 1.0904** | **0.15 + 0.85(1.0904/2) = 0.6134** | **0.15 + 0.85(0.5452 + 0.6134) = 1.1348** |

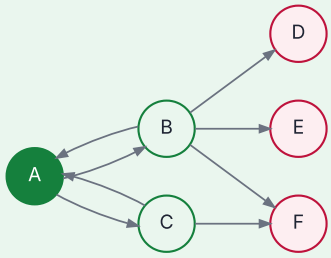
Both methods converge to the same values: **PR(A) ≈ 1.1634, PR(B) ≈ 0.6444, PR(C) ≈ 1.1922** (sum = 3, because there are no dead ends).

**ANSWER**

After 3 iterations (Method 1): **PR(A) = 1.0542, PR(B) = 0.7285, PR(C) = 1.2173** · (Method 2: A = 1.0904, B = 0.6134, C = 1.1348).
Ranking: **C > A > B** — C receives links from both A and B; B receives only half of A's vote.

## Q125 · PageRank of a 6-page graph at iteration 2 (10 marks)

A → B, C · B → A, D, E, F · C → A, F. $PR_0(m) = 1$, β = 0.85.
Out-links: L(A) = 2, L(B) = 4, L(C) = 2; D, E, F have none (dead ends).




**EQUATIONS — PR(P) = 0.15 + 0.85 · Σ PR(Q)/L(Q)**

PR(A) = 0.15 + 0.85[PR(B)/4 + PR(C)/2] · PR(B) = 0.15 + 0.85[PR(A)/2] · PR(C) = 0.15 + 0.85[PR(A)/2]
PR(D) = PR(E) = 0.15 + 0.85[PR(B)/4] · PR(F) = 0.15 + 0.85[PR(B)/4 + PR(C)/2]

| Page | Iteration 1 (from $PR_0 = 1$) | Iteration 2 (from iteration 1) |
|---|---|---|
| A | 0.15 + 0.85(0.25 + 0.5) = 0.15 + 0.6375 = **0.7875** | **0.15 + 0.85(0.575/4 + 0.575/2) = 0.15 + 0.85(0.14375 + 0.2875) = 0.5166** |
| B | 0.15 + 0.85(0.5) = **0.5750** | **0.15 + 0.85(0.7875/2) = 0.15 + 0.3347 = 0.4847** |
| C | 0.15 + 0.85(0.5) = **0.5750** | **0.15 + 0.85(0.7875/2) = 0.4847** |
| D | 0.15 + 0.85(0.25) = **0.3625** | **0.15 + 0.85(0.575/4) = 0.15 + 0.1222 = 0.2722** |
| E | 0.15 + 0.85(0.25) = **0.3625** | **0.15 + 0.85(0.575/4) = 0.2722** |
| F | 0.15 + 0.85(0.25 + 0.5) = **0.7875** | **0.15 + 0.85(0.14375 + 0.2875) = 0.5166** |
| **Sum** | 3.45 | 2.547 |

**ANSWER — ITERATION 2**

PR(A) = PR(F) = **0.5166** · PR(B) = PR(C) = **0.4847** · PR(D) = PR(E) = **0.2722**. Ranking: A = F > B = C > D = E.

**POINT TO MENTION**

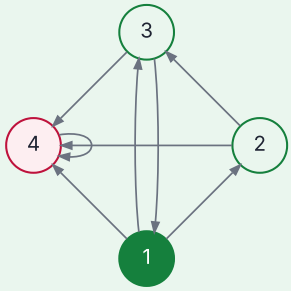
The total falls from 6 → 3.45 → 2.55 because D, E and F are **dead ends**: rank flows into them and never comes out. Fix by teleporting from dead ends to all pages or by pruning them first.

## Q124 · HITS hub and authority scores, k = 3 (10 marks)

Adjacency matrix L (row i → column j means page i links to page j):

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Links: 1 → 2, 3, 4 · 2 → 3, 4 · 3 → 1, 4 · 4 → 4




**RULES**

**a = $L^T$·h** — authority of page j = sum of hub scores of pages linking *to* j (column sums weighted by h).
**h = L·a** — hub of page i = sum of authority scores of pages i links *to* (row sums weighted by a).
Start $h_0$ = [1, 1, 1, 1]; after each update divide the vector by its largest value (max-normalisation).
In-links: page 1 ← 3 · page 2 ← 1 · page 3 ← 1, 2 · page 4 ← 1, 2, 3, 4.

### Iteration k = 1

| Node | a = $L^T$h (h = 1,1,1,1) | a normalised (÷ 4) | h = L·a | h normalised (÷ 1.75) |
|---|---|---|---|---|
| 1 | $h_3$ = 1 | 0.25 | $a_2 + a_3 + a_4$ = 0.25 + 0.5 + 1 = 1.75 | 1.000 |
| 2 | $h_1$ = 1 | 0.25 | $a_3 + a_4$ = 0.5 + 1 = 1.50 | 0.857 |
| 3 | $h_1 + h_2$ = 2 | 0.50 | $a_1 + a_4$ = 0.25 + 1 = 1.25 | 0.714 |
| 4 | $h_1 + h_2 + h_3 + h_4$ = 4 | 1.00 | $a_4$ = 1 | 0.571 |

### Iteration k = 2

| Node | a = $L^T$h | a normalised (÷ 3.143) | h = L·a | h normalised (÷ 1.909) |
|---|---|---|---|---|
| 1 | $h_3$ = 0.714 | 0.227 | 0.318 + 0.591 + 1 = 1.909 | 1.000 |
| 2 | $h_1$ = 1 | 0.318 | 0.591 + 1 = 1.591 | 0.833 |
| 3 | 1 + 0.857 = 1.857 | 0.591 | 0.227 + 1 = 1.227 | 0.643 |
| 4 | 1 + 0.857 + 0.714 + 0.571 = 3.143 | 1.000 | 1 | 0.524 |

### Iteration k = 3

| Node | a = $L^T$h | a normalised (÷ 3.0) | h = L·a | h normalised (÷ 1.944) |
|---|---|---|---|---|
| 1 | $h_3$ = 0.643 | 0.214 | 0.333 + 0.611 + 1 = 1.944 | **1.000** |
| 2 | $h_1$ = 1 | 0.333 | 0.611 + 1 = 1.611 | 0.829 |
| 3 | 1 + 0.833 = 1.833 | 0.611 | 0.214 + 1 = 1.214 | 0.625 |
| 4 | 1 + 0.833 + 0.643 + 0.524 = 3.000 | **1.000** | 1 | 0.514 |

**ANSWER (K = 3)**

Authority a = [0.214, 0.333, 0.611, **1.000**] → **best authority = node 4** (every page, including itself, links to it).
Hub h = [**1.000**, 0.829, 0.625, 0.514] → **best hub = node 1** (it links to pages 2, 3 and 4, which hold the most authority).

**IF YOUR TEACHER NORMALISES DIFFERENTLY**

Some solutions divide by √(sum of squares) instead of the maximum. The numbers change but the ranking does not: node 4 remains the top authority and node 1 the top hub.