Query Processing (Sample Q)

Q1. Natural Join of the relations,

Student (Name, Project),
Professor (Name, Project) } not sorted.

on Student.Project = Professor.Project.

B(Student) = 2,000
B(Professor) = 1,000
M = 201 : Available blocks in memory buffers.

Both B(S) > M, B(P) > M, so use "external mem. Join"

$M^2 = 201^2 = 40{,}401$

$$B(S) + B(P) \leq M^2$$

$2{,}000 + 1{,}000 \leq 40{,}401$

∴ We can use "Improved Sort-Merge" Join

→ cost : 3B(S) + 3 (P)

Q2. worst case scenario of Q1

N_s	Proj
A	1
B	1
C	1
D	1

⋈

N_p	Proj
a	1
b	1
c	1
d	1

→

Proj	N_s	N_p
1	A	a
1	A	b
1	A	c
1	A	d
⋮		

When every student and professor works on the same project,
the result of JOIN is essentially like "Improved Nested Loop" where
the cost is close to B(S) × B(P).
↳ Cartesian Product

Q3.

B(Student) = 10,000
B (Insurance) = 300 } No index.

M = 12,000

B (S) < M
B (I) < M } → Internal Memory Join

1. Nested Loop Join
2. Sort-Merge Join
3. Hash Join

⇓

∴ cost : B(S) + B(I) = 10,300 disk I/O

↳ what if M = 20? ⇒ External Memory Join

$$B(I) \leq M^2$$

min (B(I), B(S)) = $B(I) \leq M^2$ ∴ Hash Join

∴ cost : 3B(S) + 3B(I) = 3(10,300)
= 30,900

Q4. Sort a relation, R.

a. B (R) > M : External Memory Sort

- Two-pass Multi-way Merge Sort : 3B(R)
- General Multi-way Merge Sort : $B(R) \cdot \log B(R)$

b. Given M = 200,

what's the requirement for using Two-pass Multi-way Merge Sort on R?

Since the "two-pass multi-way Merge Sort" can be used when $B(R) \leq M^2$,
B(R) should be smaller than or equal to M^2(= 40,000).

Accordingly, the maximum cost of 2PMMS is 3B(R) = 3 × 40,000 = 120,000.

Query Processing (HW4)

Q1 $B(R) = 80{,}000$, $B(S) = 20{,}000$ → no index.

(a) $M = 10.$ → External Mem. Sort

Block-based Improved Nest Loop Join (Baseline)

- Cost: $B(R) + \frac{B(R)B(S)}{M-2} \approx \frac{B(R)B(S)}{M}$
- Requirement: 4 → 2 for reading R, S; 1 for join, 1 for output

↳ Read the entire outer (smaller) relation, Read the larger relation in blocks, Join them :

(b) $M = 350,\ M^2 = 122{,}500.$

$B(R) + B(S) \leq M^2$
$100{,}000 \leq 122{,}500$
→ External Optimized Sort-Merge

∴ cost: $3B(R) + 3B(S)$
$= 300{,}000$

(c) $M = 200,\ M^2 = 40{,}000$

$\min(B(R), B(S)) \leq M^2$
$20{,}000 \leq 40{,}000$
→ External Hash Join

∴ cost : $3B(R) + 3B(S)$
$= 300{,}000$

Q2. a) Given that

- R can fit into main memory,
- S is too large to fit in main memory.

Use "External" Block-based Improved Nested Loop Join.

First, read the entire relation (the smaller one), R and scan S block by block ⇒ $B(R) + \underbrace{B(S)}_{\text{Dominant!}}$

Clustered index on relations will not change the choice of merge algorithm, since it doesn't reduce I/O access.

b) B(S) and B(R) are both too large. to get a desired # of tuples after join, we can use "Improved Block-based Nested Loop" Join and stop when it returns the desired number of tuples. On the other hand, Sort-Merge Join and Hash Join is not recommended as these algorithms require pre-processing.