CS 145 Equations – v1.3 (Updated 9/1/25)

Many CS algorithms assume data fits in RAM, and focus on a CPUcost model (e.g., O(nlogn) for Sorting). "Big" data does not fit in RAM.
Our Goal: Model IOCosts for working with data in (slow) SSD/HDDs and RAM (versus fast CPUs)

[1] Basic IO Cost model for RAM, SSD, HDD

Assume: 1 (disk) Block = 1 (RAM) Page.

1. $numPages = \lceil \frac{numMBs}{pageSize} \rceil$
2. ReadCost per page (C_r) = 1 IO.
   WriteCost per page (C_w): k IOs
   ($1 \leq k \leq 10$ – Based on hardware! Default: $k = 1$)
3. Time per IO (secs): $accessLatency + \frac{pageSize}{scanSpeed}$

Example

E.g., pageSize = 64 MBs, numMBs = 630 MBs
SSD: accessTime = 10 us, scanSpeed = 5 GBps

- numPages = $\lceil \frac{630MB}{64MBs} \rceil = 10$
- $C_r = 1IO = 10us + \frac{64MB}{5GBps}$ secs
- Time to read 10 pages = $10 * C_r$
- Time to read 100 pages on 20 machines = $5 * C_r$

[2] HP and BigSort 'R' on column X

For table 'R', T(R) = # of rows in R. P(R) = # of pages in R. Buffer (B) = # of RAM pages.

1. **HashPartition**(R): IOCost= $(C_r + C_w) * P(R)$
2. **BigSort(R):** Let N = P(R). IOCost = $(C_r + C_w) * N * (1 + \lceil log_B \frac{N}{2B} \rceil)$

Special case for BigSort(R) ($C_r, C_w = 1$):

- (Small data) $N \leq 2B$? IOcost = $2N$
- If $2B < N \leq 2B^2$? $4N$; If $2B^2 < N \leq 2B^3$? $6N$

Example – $C_r, C_w = 1$

- P(R) = 1000, B = 100
- HP(R) = $2 * 1000 = 2000$ IOs
- BigSort(R) = $2*1000*(1+\lceil log_{100} \frac{1000}{2*100} \rceil) = 4000$ IOs

[3] Basic JOIN(R, S) on column X

Let OUT be the number of output pages for a JOIN depends on number of matching tuples and row size of JOIN result).

1. **BNLJ(R, S):**
   $C_r * (P(R) + P(S) * \lceil \frac{P(R)}{B} \rceil) + C_w * OUT$
   (Pick BNLJ(S, R), if cheaper – see Box[6]))

Example – $C_r, C_w = 1$

- $P(R) = 1000, P(S) = 2000, B = 100$
- $BNLJ(R, S) = 1000 + 2000 * \lceil \frac{1000}{100} \rceil + OUT$
- $BNLJ(S, R) = 2000 + 1000 * \lceil \frac{2000}{100} \rceil + OUT$

[4] HPJ(R, S) on column X

1. **HPJ(R, S):** $HP(R) + HP(S) + PJ(R, S) + C_w * OUT$
2. $HP(R) = (C_r + C_w) * P(R)$ // Ditto for $HP(S)$
3. $PJ(R, S) =$
   - ("Ideal"-case) $C_r * (P(R) + P(S))$
   - Skew? Try different hash.
   - Many duplicates? Try different algo!
4. HPJ(R, R). I.e., a self-join
   $HPJ(R, R) = HP(R) + C_r * P(R) + C_w * OUT$

Special case – $C_r, C_w = 1$

- $HPJ(R, S) = 3(P(R) + P(S)) + OUT$

Example – $C_r, C_w = 1$

- P(R) = 1000, P(S) = 2000, B = 100
- $HPJ(R, S) = 3 * (1000 + 2000) + OUT$
- $HPJ(R, R) = 3 * 1000 + OUT$

[5] SMJ(R, S) with JOIN on column X

1. **SMJ(R,S):** BigSort(R) + BigSort(S) + MergeJoin(R, S) + $C_w * OUT$
2. MergeJoin(R, S) with JOIN on column X:
   (a) (Best-case: No Backup) $C_r * (P(R) + P(S))$
   (b) (Worst-case: All Backup) $C_r * P(R) * P(S)$
3. SMJ(R, R). I.e., a self-join
   (a) (Best-case: No Backup) $BigSort(R) + C_r * P(R) + C_w * OUT$
   (b) (Worst-case: All Backup) $BigSort(R) + C_r * P(R) * P(R) + C_w * OUT$
4. If R is pre-sorted, BigSort(R) = 0. (Same as S.)

Example – $C_r = C_w = 1$

- P(R) = 1000, P(S) = 2000, B = 100
- SMJ(R, S) = 4000+8000+3000 + OUT IOs
  - BigSort(R) = $2 * 1000 * (1 + \lceil log_{100} \frac{1000}{2*100} \rceil) = 4000$
  - BigSort(S) = $2 * 2000 * (1 + \lceil log_{100} \frac{2000}{2*100} \rceil) = 8000$
  - MergeJoin(R, S) = 3000 IOs, no backup
  - MergeJoin(R, S) = 1000*2000 IOs, full backup

[6] Indexing: Hash Index and B+ Trees

- n: Number of data rows , N: Number of data pages
- d: Data row size, k: Key size, p: Pointer size
- M: Page size, h: index height

Data Pages:	$N = \lceil \frac{n \cdot d}{M} \rceil$
Index Fanout:	$f = \lfloor \frac{M}{k+p} \rfloor$
Hash Index (Unsorted): $P(index)$	$= \lceil \frac{n}{f} \rceil$
Hash Index (Sorted): $P(index)$	$= \lceil \frac{N}{f} \rceil$
B+ Tree Height (Unsorted): h	$\approx \lceil \log_f(n) \rceil$
B+ Tree Height (Sorted): h	$\approx \lceil \log_f(N) \rceil$
Query IOs: (index IO + data IO)	$\approx$ h + 1

Example Values:

- $M = 64MB$, $k = 8$ $bytes$, $p = 8$ $bytes$, $n = 10^{12}$, $d = 1024$ $bytes$
- $f = \lfloor \frac{64*1024*1024}{8+8} \rfloor = 4,194,304$
- $N = \lceil \frac{10^{12} \cdot 1024}{64*1024*1024} \rceil = 15258790$
- B+ Tree height (unsorted): $\lceil \log_{4194304}(10^9) \rceil = 2$
- B+ Tree height (sorted): $\lceil \log_{4194304}(15258790) \rceil = 2$