CS186 Discussion 6 – Joins

Tables:
Companies: (company_id, industry, ipo_date)
Nyse: (company_id, date, trade, quantity)

We have 20 pages of memory, and we want to join two tables Companies and NYSE on C.company_id = N.company_id
Attribute company_id is the primary key for Companies.
For every tuple in Companies, assume there are 4 matching tuples in NYSE.

NYSE contains [N] = 100 pages, NYSE holds pN = 100 tuples per page.
Companies contains [C] = 50 pages, C holds pC = 50 tuples per page.
These are unclustered B+ indexes on C.company_id and N.company_id
For both indexes, assume it takes 2 I/Os to access a leaf.

1. How many disk I/Os are needed to perform a simple nested loops join?
   Using Companies as the outer relation yields the lower I/O count
   [C] + pC*[C]*[N] = 50 + 50*50*100 = 50 + 50*50*100 = 250,050 I/Os
2. Index nested loop join?
   With C as the outer relation (meaning we probe on the index on N):
   [C] + [C] * pC * (cost to find matching NYSE tuples)
   = 50 + 50 * 50 * (2 + 4) = 15,050 I/Os
3. How about block nested loops join?
   (# pages in smaller relation) + ceil[(# pages in smaller relation) / (# pages in memory – 2)] * (# pages in larger relation)
   = 50 + ceil(50/18) * 100 = 350 I/Os
4. How about sort merge join? (Assume that both tables are sorted in 2 passes and that we join during the merge pass of sorting i.e. we are using the optimized version of sort merge join)
   3[C] + 3[N] = 450 I/Os
5. How about a hash join? (Assume no recursive partitioning and ignore output costs)
   Partitioning phase: 2([C] + [N])
   Matching phase: [C] + [N]
   3[C] + 3[N] = 450 I/Os
6. Now assume the index on NYSE.company_id is *clustered*. What is the cost of an index nested loops join using companies as the outer relation?
   [C] + [C]*pC * (cost to find matching NYSE tuples)
   = 50 + 50 * 50 * (2 + $\lceil 4/100 \rceil$) = 7,550 I/Os