CS3223 Tutorial 3: Sorting & Selection

Wk 5, Sem 2, 2022/23

Questions to be discussed: 2, 4 & 5

1. (Exercise 13.1, R&G)

   Answer the following questions (a to d) for each of the scenarios (i to iii), assuming that the external merge sort algorithm is used:

   (i) A file with 10,000 pages and three available buffer pages.

   (ii) A file with 20,000 pages and five available buffer pages.

   (iii) A file with 2,000,000 pages and 17 available buffer pages.

   (a) How many runs will you produce in the first pass?

   (b) How many passes will it take to sort the file completely?

   (c) What is the total I/O cost of sorting the file?

   (d) How many buffer pages do you need to sort the file completely in just two passes?

Solution:

(a) In the first pass (Pass 0), $\lceil N/B \rceil$ runs of B pages each are produced, where N is the number of file pages and B is the number of available buffer pages:

   (i) $\lceil 10000/3 \rceil = 3334$ sorted runs.

   (ii) $\lceil 20000/5 \rceil = 4000$ sorted runs.

   (iii) $\lceil 2000000/17 \rceil = 117648$ sorted runs.

(b) The number of passes required to sort the file completely, including the initial sorting pass, is $\log_{B-1}(N_0) + 1$, where $N_0 = \lceil N/B \rceil$ is the number of runs produced by pass 0:

   (i) $\lceil \log_2(3334) \rceil + 1 = 13$ passes.

   (ii) $\lceil \log_4(4000) \rceil + 1 = 7$ passes.

   (iii) $\lceil \log_{16}(117648) \rceil + 1 = 6$ passes.

(c) Since each page is read and written once per pass, the total number of page I/Os for sorting the file is $2 \times N \times (\#passes)$:

   (i) $2 \times 10000 \times 13 = 260000$.

   (ii) $2 \times 20000 \times 7 = 280000$.

   (iii) $2 \times 2000000 \times 6 = 24000000$.

(d) In Pass 0, $\lceil N/B \rceil$ runs are produced. In Pass 1, we must be able to merge this many runs; i.e., $B - 1 \geq \lceil N/B \rceil$. This implies that B must at least be large enough to satisfy $B \times (B-1) \geq N$; this can be used to guess at B, and the guess must be validated by checking the first inequality. Thus:

   (i) With 10000 pages in the file, B = 101 satisfies both inequalities, B = 100 does not, so we need 101 buffer pages.

   (ii) With 20000 pages in the file, B = 142 satisfies both inequalities, B = 141 does not, so we need 142 buffer pages.

   (iii) With 2000000 pages in the file, B = 1415 satisfies both inequalities, B = 1414 does not, so we need 1415 buffer pages.

2. (Exercise 13.4, R&G) Consider a disk with an average seek time of 10ms, average rotational delay of 5ms, and a transfer time of 1ms for a 4096-byte page. Assume that the cost of reading/writing a page is the sum of these values (i.e., 16ms) unless a sequence of pages is read/written. In this case, the cost is the average seek time plus the average rotational delay (to find the first page in the sequence) plus 1ms per page (to transfer data). You are given 320 buffer pages and asked to sort a file with 10,000,000 pages. Assume that you begin by creating sorted runs of 320 pages each in the first pass. Evaluate the cost of the following approaches for the subsequent merging passes:

   (i) Do 319-way merges.

   (ii) Create 256 input buffers of 1 page each, create an output buffer of 64 pages, and do 256-way merges.

   (iii) Create 16 input buffers of 16 pages each, create an output buffer of 64 pages, and do 16-way merges.

   (iv) Create eight input buffers of 32 pages each, create an output buffer of 64 pages, and do eight-way merges.

   (v) Create four input buffers of 64 pages each, create an output buffer of 64 pages, and do four-way merges.

   Complete the following table.

	Merging Factor	Input block size	Output block size	# of Merging passes	Total merging time (secs)
i	319	1	1		
ii	256	1	64		
iii	16	16	64		
iv	8	32	64		
v	4	64	64		

Solution: In Pass 0, 31250 sorted runs of 320 pages each are created. The calculations below are slightly simplified by neglecting the effect of a final read/written block that is slightly smaller than the earlier blocks.

The number of merging passes is given by $\lceil log_f(31250) \rceil$, where f denote the merging factor.

	Merging Factor	Input block size	Output block size	# of Merging passes	Total merging time (secs)
i	319	1	1	2	640,000
ii	256	1	64	2	344,687.5
iii	16	16	64	4	126,875
iv	8	32	64	5	135,156.25
v	4	64	64	8	197,500

(i) For 319-way merges, only 2 merging passes are needed. The first pass will produce $\lceil 31250/319 \rceil = 98$ sorted runs; these can then be merged in the next pass. Every page is read and written individually, at a cost of 16ms per read or write, in each of these two passes. The cost of these merging passes is therefore $2 \times 2 \times 16 \times 10000000 = 640,000,000ms$. (The formula can be read as number of passes times cost of read and write per page times number of pages in file.)

(ii) With 256-way merges, only 2 merging passes are needed. Every page in the file is read and written in each pass, but the effect of blocking is different on reads and writes. For reading, each page is read individually at a cost of 16ms. Thus, the cost of reads (over both passes) is $2 \times 16 \times 10000000 = 320000000ms$. For writing, pages are written out in blocks of 64 pages. The I/O cost per block is $10 + 5 + 1 \times 64 = 79ms$. The number of blocks written out per pass is $10000000/64 = 156250$, and the cost per pass is $156250 \times 79 = 12343750ms$.

The cost of writes over both merging passes is therefore $2 \times 12343750 = 24687500ms$. The total cost of reads and writes for the two merging passes is $320000000 + 24687500 =$ 344,687,500ms.

(iii) With 16-way merges, 4 merging passes are needed. For reading, pages are read in blocks of 16 pages, at a cost per block of $10 + 5 + 1 \times 16 = 31ms$. In each pass, 10000000/16 = 625000 blocks are read. The cost of reading over the 4 merging passes is therefore $4 \times 625000 \times 31 = 77500000ms$. For writing, pages are written in 64-page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 4 merging passes is $4 \times 12343750 = 49375000ms$, and the total cost of the merging passes is $77500000+ 49375000 =$ 126,875,000ms.

(iv) With 8-way merges, 5 merging passes are needed. For reading, pages are read in blocks of 32 pages, at a cost per block of $10 + 5 + 1 \times 32 = 47ms$. In each pass, 10000000/32 = 312500 blocks are read. The cost of reading over the 5 merging passes is therefore $5 \times 312500 \times 47 = 73437500ms$. For writing, pages are written in 64-page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 5 merging passes is $5 \times 12343750 = 61718750ms$, and the total cost of the merging passes is $73437500+ 61718750 =$ 135,156,250ms.

(v) With 4-way merges, 8 merging passes are needed. For reading, pages are read in blocks of 64 pages, at a cost per block of $10 + 5 + 1 \times 64 = 79ms$. In each pass, 10000000/64 = 156250 blocks are read. The cost of reading over the 8 merging passes is therefore $8 \times 156250 \times 79 = 98750000ms$. For writing, pages are written in 64-page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 8 merging passes is $8 \times 12343750 = 98750000ms$, and the total cost of the merging passes is $98750000+ 98750000 =$ 197,500,000ms.

There are several lessons to be drawn from this (rather tedious) exercise. First, the cost of the merging phase varies from a low of 126,875,000ms to a high of 640,000,000ms. Second, the highest cost is associated with the option of maximizing fanout, choosing a buffer size of 1 page! Thus, the effect of blocked I/O is significant. However, as the block size increases, the number of passes increases slowly, and there is a trade-off to be considered: it does not pay to increase block size indefinitely. Finally, while this example uses a different block size for reads and writes, for the sake of illustration, in practice a single block size is used for both reads and writes.

3. (Exercise 12.4, R&G) Consider the following schema with the Sailors relation:

   Sailors (sid: integer, sname: string, rating: integer, age: real)

   For each of the following indexes, list whether the index matches the given selection conditions. If the index is not a match, list the primary conjuncts.

   (a) A B$^+$-tree index on the search key `(Sailors.sid)`.
      (i) $\sigma_{Sailors.sid<50{,}000}$(Sailors)
      (ii) $\sigma_{Sailors.sid=50{,}000}$(Sailors)

   (b) A hash index on the search key `(Sailors.sid)`.
      (i) $\sigma_{Sailors.sid<50{,}000}$(Sailors)
      (ii) $\sigma_{Sailors.sid=50{,}000}$(Sailors)

   (c) A B$^+$-tree index on the search key `(Sailors.sid, Sailors.age)`.
      (i) $\sigma_{Sailors.sid<50{,}000 \,\wedge\, Sailors.age=21}$(Sailors)
      (ii) $\sigma_{Sailors.sid=50{,}000 \,\wedge\, Sailors.age>21}$(Sailors)
      (iii) $\sigma_{Sailors.sid=50{,}000}$(Sailors)
      (iv) $\sigma_{Sailors.age=21}$(Sailors)

   (d) A hash index on the search key `(Sailors.sid, Sailors.age)`.
      (i) $\sigma_{Sailors.sid=50{,}000 \,\wedge Sailors.age=21}$(Sailors)
      (ii) $\sigma_{Sailors.sid=50{,}000 \,\wedge\, Sailors.age>21}$(Sailors)
      (iii) $\sigma_{Sailors.sid=50{,}000}$(Sailors)
      (iv) $\sigma_{Sailors.age=21}$(Sailors)

Solution:

(a) (i) Matches index.
    (ii) Matches index.

(b) (i) Does not match index. Hash indexes are not used for range queries. No primary conjuncts.
    (ii) Matches index.

(c) (i) Does not match index. Primary conjunct is: Sailors.sid $<$ 50,000.
    (ii) Matches index.
    (iii) Matches index.
    (iv) Does not match index. No primary conjuncts.

(d) (i) Matches index.
    (ii) Does not match index. No primary conjuncts.
    (iii) Does not match index. No primary conjuncts.
    (iv) Does not match index. No primary conjuncts.

4. (Exercise 12.5, R&G) Consider again the schema with the Sailors relation:

   Sailors (<u>sid</u>: integer, sname: string, rating: integer, age: real)

   where `sid` is the primary key of `Sailors`. Assume that each tuple of Sailors is 50 bytes long, that a page can hold 80 Sailors tuples, and that we have 500 pages of such tuples. For each of the following selection conditions, estimate the number of pages retrieved, given the catalog information in the question.

   (a) Assume that we have a B$^+$-tree index T on the search key `(Sailors.sid)`, and assume the following:
   - the number of internal (ie non-leaf) levels in $T = 4$,
   - the number of leaf pages in $T = 50$, and
   - $Sailors.sid \in [1, 100, 000]$

   (i) $\sigma_{Sailors.sid<50,000}$(Sailors)
   (ii) $\sigma_{Sailors.sid=50,000}$(Sailors)

   (b) Assume that we have a hash index T on the search key `(Sailors.sid)`, and assume the following:
   - the number of primary and overflow pages in $T = 50$, and
   - $Sailors.sid \in [1, 100, 000]$

   (i) $\sigma_{Sailors.sid<50,000}$(Sailors)
   (ii) $\sigma_{Sailors.sid=50,000}$(Sailors)

Solution: We have $|Sailors| = 500$.

(a) (i) Assuming uniform distribution, around half of the tuples in `Sailors` ($\frac{500\times 80}{2} = 20,000$) will match the selection condition. Since T has only 50 leaf pages, T is not a format-1 index. The cost of retrieving each matching tuple could be as high as 20,000 I/Os (one I/O for each matching tuple), leading to a total cost of $4 + \frac{50}{2} + 20,000 = 20,029$ I/Os. Thus, doing a table scan is most likely the preferred method with a cost of 500 I/Os. However, if the index was a clustered index, the total cost of an index scan followed by RID lookup would be $4 + \frac{50}{2} + \frac{500}{2} = 279$ I/Os which is more efficient than a full table scan.

(ii) Since sid is a primary key for the relation, we expect at most one matching tuple; therefore, the cost is just the cost of navigating the B$^+$-tree index (5 I/Os) plus the cost of reading the qualifying tuples page (1 I/O) which adds up to be 6 I/Os.

(b) (i) Since a hash index on sid cannot help us for range queries, the index is useless, and therefore we must do a table scan at a cost of 500 pages I/Os (the cost of reading the entire relation).

(ii) Since sid is a primary key for the relation, we expect at most one matching tuple; therefore, the cost of using the hash index is the sum of the cost of reading the hashed bucket (between 1 I/O and 50 I/Os depending on the length of overflow chain), and the cost of reading the qualifying tuples page (1 I/O). Thus, the cost is between 2 and 51 I/Os.

5. Consider a relation with the schema `Hotel` (<u>id</u>, name, address, phone, price, rating) and the following query

   SELECT name FROM Hotel WHERE price $>$ 500 AND rating $>$ 4

   Assume the following for this question:

   1. The `Hotel` relation contains 10,000 pages with each page containing 20 records.
   2. There are five unclustered indexes on the `Hotel` relation:
      - I_p: a B$^+$-tree index on (price),
      - I_r: a B$^+$-tree index on (rating),
      - I_{pr}: a B$^+$-tree index on (price, rating),
      - I_{rp}: a B$^+$-tree index on (rating, price), and
      - I_{npr}: a B$^+$-tree index on (name, price, rating).
   3. For each of I_p and I_r, it has 2000 leaf pages with at most 100 data entries in each leaf page.
   4. For each of I_{pr} and I_{rp}, it has 2500 leaf pages with at most 80 data entries in each leaf page.
   5. For I_{npr}, it has 4000 leaf pages with at most 50 data entries in each leaf page.
   6. Each of the indexes has 2 levels of internal nodes.
   7. Only 10% of `Hotel` records satisfy the condition "price $>$ 500".
   8. Only 20% of `Hotel` records satisfy the condition "rating $>$ 4".
   9. Only 1% of `Hotel` records satisfy both the conditions "price $>$ 500" and "rating $>$ 4".

   What is the best evaluation plan for the query? What is its cost?

Solution:

- Number of records satisfying condition on price: $0.1 \times 10,000 \times 20 = 20,000$
- Number of records satisfying condition on rating: $0.2 \times 10,000 \times 20 = 40,000$
- Number of records satisfying both conditions: $0.01 \times 10,000 \times 20 = 2,000$

1. Plan using table scan: cost = 10,000 pages.
2. Plan using index scan of I_{npr}: $2 + 4000 = 4002$. As I_{npr} is a covering index for the query, no RID lookup is required. However, as there is no primary conjunct in the predicate that matches I_{npr}, all the leaf pages in I_{npr} have to be scanned.
3. Plan using index scan of I_{pr} with RID lookup: $2 + (2500 \times 0.1) + 2000 = 2252$. Note that the only primary conjunct in the predicate that matches I_{pr} is "price $>$ 500".
4. Plan using index scan of I_{rp} with RID lookup: $2 + (2500 \times 0.2) + 2000 = 2502$. Note that the only primary conjunct in the predicate that matches I_{rp} is "rating $>$ 4".
5. Plan using index scan of I_p with RID lookup: $2 + (2000 \times 0.1) + 20,000 = 20,202$. Note that the only primary conjunct in the predicate that matches I_p is "price $>$ 500".
6. Plan using index scan of I_r with RID lookup: $2 + (2000 \times 0.2) + 40,000 = 40,402$. Note that the only primary conjunct in the predicate that matches I_r is "rating $>$ 4".
7. Plan using index intersection of I_p & I_r with RID lookup: Assuming that the retrieved RIDs from the two index scans could fit in main memory and the intersection of the RIDs could be performed without incurring any I/O, the lower bound cost of this approach is $(2+(2000\times0.1)) + (2+(2000\times0.2))+2000 = 2604$. Since the lower bound cost is already higher than the cost of the plan using I_{pr}, this plan can't be the optimal plan.

In the above discussion, we assume a RID-lookup cost of one page I/O for each matching data entry. For plans 5 and 6, even if we assume that the buffer pool is large enough (or that the RIDs are sorted before the data records are retrieved) so that no page is retrieved more than once, the cost of RID lookups for these plans is still very high ($10,000$ page I/Os in the worst case).

The cheapest plan (with a cost of 2252) is using index scan of I_{pr} with RID lookup.