See course website for instructions and due date.

1. In a recent class you completed a worksheet analyzing the performance of external merge sort. (Solutions to this worksheet are on moodle.) These two questions reference the pseudocode algorithm shown on the worksheet.

   (a) Suppose in the sort phase, only one page is read at a time and thus each page is initially assigned to its own separate run file. What is the *total cost* of this modified external merge sort? (In other words, I'm not just asking about the cost of the sort phase but asking you to consider how this modification in the sort phase may also affect the cost of later phases.)

   > **Solution:** The output of the sort phase would be N run files (one per page) rather than $\left\lceil \frac{N}{B} \right\rceil$ run files. Thus, the total cost would be: $2N(1 + \lceil \log_{B-1} N \rceil)$
   >
   > In more practical terms, if we only sort individual pages in the sort phase, it costs one round additional round of merging in the merge phase: $\log_{B-1} N \approx 1 + \log_{B-1} \left\lceil \frac{N}{B} \right\rceil$.

   (b) Suppose the sort phase is as described on the worksheet but in the merge phase, only two run files are merged at a time. What is the *total cost* of this modified external merge sort?

   > **Solution:** Total cost: $2N(1 + \left\lceil \log_2 \left\lceil \frac{N}{B} \right\rceil \right\rceil)$

   (c) Suppose B is fixed to some constant (say, 100) but N is very, very large. Which of the above variants has lower I/O cost?

   > **Solution:** The first one. The comparison is between roughly $(1 + \left\lceil \log_{B-1} \left\lceil \frac{N}{B} \right\rceil \right\rceil)$ vs $\left\lceil \log_2 \left\lceil \frac{N}{B} \right\rceil \right\rceil)$ and the former is much smaller for $B = 100$ and large N.

   (d) Assume the algorithm is as described on the worksheet. Given a table with N pages, how much memory is necessary to sort the page in two passes? By two passes, I mean the first pass is a sort and the second pass consists of one round of merging. Your answer should be a function of N, such as $\frac{N}{2}$, $\sqrt{N}$, $\log N$, and it can be approximate.

   > **Solution:** There are k run files after the sort phase. We require that all k can be merged into a single file. Thus $\frac{k}{B-1} \leq 1$ and since $k = \left\lceil \frac{N}{B} \right\rceil$, that implies $N \leq B(B-1)$. In other words, to sort a table of size N in two passes, you need roughly $\sqrt{N}$ memory.

2. Consider the join $R \bowtie_{R.a=S.b} S$, given the following information about the relations to be joined. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be ignored – however, have at least one buffer page to store the output!

   - Relation R contains 20,000 tuples and has 10 tuples per page. Relation S contains 5,000

tuples and also has 10 tuples per page. Attribute S.b is the primary key for S.

- Both relations are stored as simple heap files, and neither relation has any indexes.
- Unless otherwise specified, 42 buffer pages are available.

(a) What is the cost of (simple) nested loops join? Assume R is the outer relation.

> **Solution:** Let $N_R = 2000$ be the number of pages in R, $N_S = 500$ be the number of pages in S.
>
> Cost is lower when the smaller relation, S, is the outer relation.
>
> Cost is $N_R + N_R * p_R * N_S = 2000 + (2000 * 10 * 500) = 10,002,000$.

(b) What is the cost of block nested loops join? *Choose as the outer relation whichever relation leads to lower cost.*

> **Solution:** Cost of option two is cheaper with S as outer relation: $N_S + \left\lceil \frac{N_S}{M-2} \right\rceil * N_R = 500 + 13 * 2000 = 26,500$.

(c) Consider the merge phase of the sort-merge join. In class, we discussed how the cost of merge is variable, and the worst-case cost is *much* higher than the best-case. Given that $S.b$ is a key for S, what is the worst-case cost for the merge phase? Briefly explain.

> **Solution:** The worst case cost of the merge is $N_R + N_S = 2500$.
>
> Because $S.b$ is a key, we know that each group of tuples in R matches at most one tuple of S. Therefore, in completing the merge step, we need only look at each tuple of each relation only once. (If $S.b$ were not a key, it might be necessary to perform a nested loops join to match multiple tuples of S with multiple tuples of R.) Since it's a key, we know that we can complete the merge in a single pass through each relation.

(d) Consider a join of R and S using hashing. Is the buffer large enough to perform in-memory joins following hashing? Assume the hash function is perfect (keys are evenly distributed), there is no data skew (distribution of key values in data is uniform), further assume that a hash table on k pages requires only k pages of space (in practice it's a little bigger).

> **Solution:** Yes!
>
> Number of hash partitions $H = B - 1$. Size of each partition assuming perfect hash: $\lceil N_R/B - 1 \rceil = \lceil 2000/41 \rceil = 49$. Thus, the partitions of R will be too big to fit in memory. However, the partitions of S can fit in memory: $\lceil N_S/B - 1 \rceil = \lceil 500/41 \rceil =$

13. When joining partition R_i with S_i, we only need one of the two partitions to fit in memory.

(e) For the given R and S, what is the minimum number of buffer pages needed to achieve a minimum cost hash join? Give an *exact* number. Make the same assumptions as in part (d).

Solution: Need partitions of smaller relation S to fit in memory. Let H be the number of hash partitions.

We have two constraints $H \leq B - 1$ to write out the partitions and $\lceil \frac{N_S}{H} \rceil \leq B - 2$ so that we can fit each partition of S into memory with a block for R and another block for the output.

We want H to be as large as possible (the more hash partitions, the better). So let's set $H = B - 1$. Now thinking about the constraint that $\lceil \frac{N_S}{H} \rceil \leq B - 2$, we can ignore the ceiling and we get $N_S \leq (B-2)(B-1) \approx (B-1)^2$. The smallest integer B that satisfies this is $B = 23$.

Since we were hand-wavy, double check that B is correct.

If $B = 22$, then $\lceil \frac{N_S}{H} \rceil = 24$ which is too big!

If $B = 23$, then $\lceil \frac{N_S}{H} \rceil = 23$ which is too big!

If $B = 24$, then $\lceil \frac{N_S}{H} \rceil = 22 \leq B - 2$, so it is just right.