Subject: Clarification Requests on I/O Cost Calculation in Assignment 3

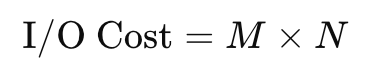
Dear TA,

I hope this message finds you well.

I am working on **Assignment 3, Section 2.2: Join Algorithms**, specifically addressing **Part (c)** related to the sort-merge join I/O cost analysis. While reviewing **Tutorial 9**, I encountered confusion regarding the I/O cost calculation for the sort-merge join in the worst-case scenario. I would greatly appreciate your assistance clarifying this matter to ensure my understanding aligns with the course’s teachings.

**## Background:**

In **Tutorial 9**, it was stated that in the worst-case scenario for a sort-merge join, the I/O cost of the merge phase is calculated as:



Where:

* **M** is the number of pages in the outer relation (**B**),
* **N** is the number of pages in the inner relation (**C**).

Given the problem parameters:

* **Total Buffer Pages (F):** 500
* **Reserved Pages:** One page for the output buffer. One page for the current input block of the inner relation (**C**).
* **Usable Buffer Pages:** F-2 = 498

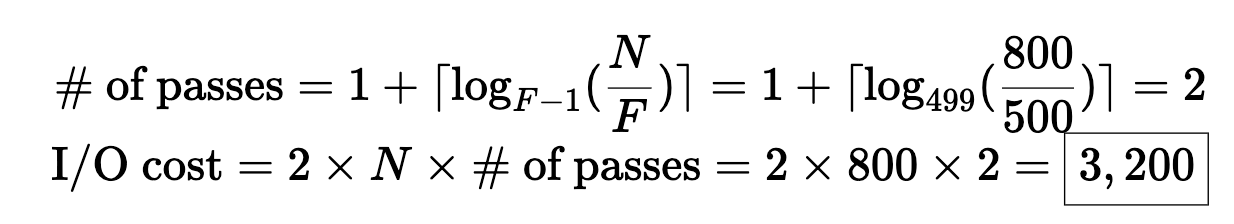
**Relations:**

* **Relation B:** 800 pages
* **Relation C:** 2,000 pages

**## My Calculations:**

**1. Sorting Phase:**

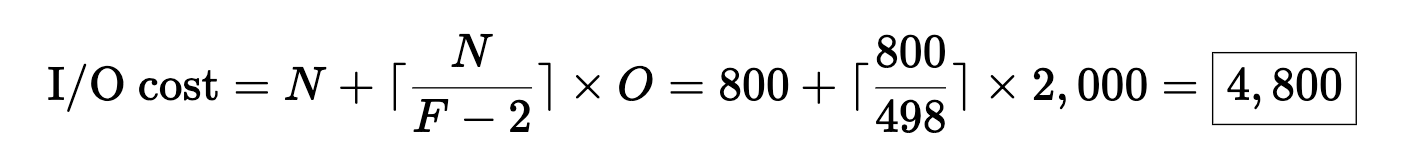
For sorting table **B**:



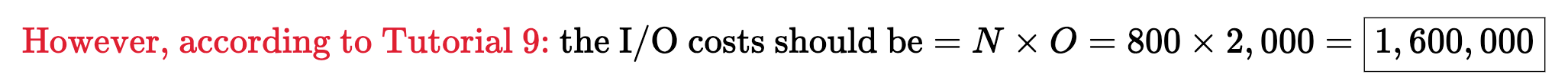
This phase is okay.

**2. Merge Phase:**

In the worst-case scenario where all join attributes are identical and sorted:



**Discrepancy with Tutorial 9:**



This calculation suggests that the entire inner relation C is scanned for each page in the outer relation B, leading to a significantly higher I/O cost. However, this seems **counterintuitive**, given the **buffer size F=500 constraints** provided.

**## Questions:**

1. **I/O Cost Calculation in Tutorial 9:**
   * **Assumption:** With 498 usable buffer pages, it’s possible to load multiple pages of the outer relation **B** into memory simultaneously, thereby reducing the number of times the inner relation **C** needs to be read.
   * **Impact:** This should lower the I/O cost from **M × N = 1,600,000 pages I/Os** to a value closer to **4,800 pages I/Os,** as per my calculations.
   * **Question:** Why does **Tutorial 9** suggest an I/O cost of **M × N** without considering the available buffer pages (i.e., 498)? Is there an aspect of the sort-merge join algorithm that requires re-reading the inner relation **C** for every page of the outer relation **B**, irrespective of the buffer size?
2. **Comparison with Hash Join Optimization:**
   * **Observation:** In the partitioned hash join algorithm, buffer pages are effectively utilized to minimize I/O costs by loading larger portions of the outer relation into memory during the probe phase.
   * **Question:** Why doesn’t a similar buffer optimization apply to the sort-merge join in the context provided by **Tutorial 9**? Shouldn’t the available buffers be used to reduce the number of inner relation scans?

Thank you very much for your time and assistance. I look forward to your guidance on this matter.

Best regards,

Chaoren