CSC 370

Activity Worksheet:
IO Model of Computation

Dr. Sean Chester

Fall 2022

Notes

In this worksheet, you will practice calculating the cost of an algorithm within the I/O model. In each question, you are provided with the pseudocode for an algorithm and important statistics. You should determine the number of blocks input and output by the algorithm. If relevant, distinguish between the best and worst case. The first question has been answered already as an exemplar.

Questions

1. Assume that a block is 4KB and that you have 4MB available in main memory. You are given a relation R that contains ten thousand contiguously-stored tuples that each require 92B. What is the cost of the algorithm below, which performs a selection on R?

```
1. for each block b \in R:
2.     for each tuple t \in b:
3.         if t.x < 100:
4.             add t to output buffer, o
5.             if o is full:
6.                 output o
7. if o is not empty:
8.     output o
```

Solution: This is a single-pass, tuple-at-a-time algorithm; so, the size of main memory is only important to establish that there is space to read in one block and maintain one block for an output buffer.

Because tuples should not be split across blocks and because blocks are atomically input and output, we need to be careful when calculating the number of blocks occupied by R, B(R):

$$B(R) = \lceil \frac{T(R)}{\lfloor \frac{4KB}{92B} \rfloor} \rceil = \lceil \frac{10000}{44} \rceil = \lceil \frac{T(R)}{\lfloor \frac{4096}{92} \rfloor} \rceil = 228$$

We see that Line 1 is the only line that inputs data from disk and it reads each block of R exactly once. Therefore, the total number of input blocks for this algorithm is exactly B(R)=228.

We see that Line 6 and Line 8 are the only lines that output data to disk. Line 6 outputs each full block of output and Line 8 outputs the partially-full block at the end of the of the algorithm, if there is one. We see that each tuple of R is added to one of these buffers either zero or one time(s) and the entire 92B tuple is output. Therefore, the total number of output blocks, B(O), for this algorithm ranges from a minimum of zero blocks to a maximum of B(R)=228 blocks.

2. Assume that a block is 4KB and that you have 4MB available in main memory. You are given a relation R that contains ten thousand contiguously-stored tuples that each require 92B, including the 4B attribute x. What is the cost of the algorithm below, which performs a selection and a projection on R?

```
1. for each block b \in R:
2.     for each tuple t \in b:
3.         if t.x < 100:
4.             add t.x to output buffer, o
5.             if o is full:
6.                 output o
7. if o is not empty:
8.     output o
```

Solution:

3. Assume that a block is 4KB and that you have 4MB available in main memory. You are given a relation R that contains ten thousand contiguously-stored tuples that each require 92B. What is the cost of the algorithm below, which performs a simplistic duplicate elimination on R? What if you only have 400KB available in main memory?

```
1. for each block bi \in R:
2.     for each tuple tj \in b:
3.        save tj in main memory array index a[44i + j]
4. for each tuple ti in a:
5.     if \nexists tj such that j<i and ti=tj:
6.             add ti to output buffer, o
7.             if o is full:
8.                 output o
9. if o is not empty:
A.    output o
```

Solution:

4. Assume that a block is 4KB and that you have 4MB available in main memory. You are given a relation R that contains ten thousand contiguously-stored tuples that each require 92B. You are also given a relation S that contains one hundred contiguously-stored tuples that each require 48B. What is the cost of the algorithm below, which performs a cross product between R and S?

```
1. for each block bs \in S:
2.     for each block br \in R:
3.         for each tuple ts \in bs:
4.             for each tuple tr \in br:
5.                 add ts \bowtie tr to output buffer, o
6.                 if o is full:
7.                     output o
8. if o is not empty:
9.     output o
```

Solution:

Solutions

Question 1

This is a single-pass, tuple-at-a-time algorithm; so, the size of main memory is only important to establish that there is space to read in one block and maintain one block for an output buffer.

Because tuples should not be split across blocks and because blocks are atomically input and output, we need to be careful when calculating the number of blocks occupied by R, B(R):

$$B(R) = \lceil \frac{T(R)}{\lfloor \frac{4KB}{92B} \rfloor} \rceil = \lceil \frac{10000}{44} \rceil = \lceil \frac{T(R)}{\lfloor \frac{4096}{92} \rfloor} \rceil = 228$$

We see that Line 1 is the only line that inputs data from disk and it reads each block of R exactly once. Therefore, the total number of input blocks for this algorithm is exactly B(R)=228.

We see that Line 6 and Line 8 are the only lines that output data to disk. Line 6 outputs each full block of output and Line 8 outputs the partially-full block at the end of the of the algorithm, if there is one. We see that each tuple of R is added to one of these buffers either zero or one time(s) and the entire 92B tuple is output. Therefore, the total number of output blocks, B(O), for this algorithm ranges from a minimum of zero blocks to a maximum of B(R)=228 blocks.

Question 2

This is also a single-pass, tuple-at-a-time algorithm; so, the size of main memory is only important to establish that there is space to read in one block and maintain one block for an output buffer.

As in question 1, because tuples should not be split across blocks and because blocks are atomically input and output, we need to be careful when calculating the number of blocks occupied by R, B(R):

$$B(R) = \lceil \frac{T(R)}{\lfloor \frac{4KB}{92B} \rfloor} \rceil = \lceil \frac{10000}{44} \rceil = \lceil \frac{T(R)}{\lfloor \frac{4096}{92} \rfloor} \rceil = 228$$

We see that Line 1 is the only line that inputs data from disk and it reads each block of R exactly once. Therefore, the total number of input blocks for this algorithm, like in Question 1, is exactly B(R)=228.

We see that Line 6 and Line 8 are the only lines that output data to disk. However, we do not need to copy entire tuples to the output buffer, only the 4B attribute x. Therefore, we can fit

$$\lfloor \frac{4KB}{4B} \rfloor = 1034$$

output tuples on each output block. Since each tuple appears exactly zero or one time(s) in the output, the number of output blocks ranges from a minimum of zero to a maximum of

$$\lceil \frac{T(R)}{1024} \rceil = \lceil \frac{10000}{1024} \rceil = 10$$

blocks.

Question 3

This is also a dual-pass algorithm, but the second pass over the data occurs in main memory; so, the response critically depends on whether the relation fits in memory or not. We can calculate that main memory has a capacity for

$$\lfloor \frac{4MB}{4KB} \rfloor = 1024$$

blocks. We now from the previous two questions that B(R) = 228; so, R fits in main memory.

As before, we see that Line 1 is the only line that inputs data from disk and it reads each block of R exactly once. Although the data is read multiple times, it is only once input from disk. Therefore, the total number of input blocks for this algorithm, like in Question 1 and Question2, is exactly B(R)=228.

Moreover, the output tuples are identical to the tuples in R. The duplicate elimination operator could, in principle, return as few as 1 tuple or as many as T(R) tuples. Therefore, the output size, B(O), is between a minimum of 1 and a maximum of B(R) = 228.

In the case that main memory has only 400KB available, it only has capacity for 10 blocks. Therefore, the array a will not fit in memory. A basic solution is to remark that this algorithm will fail. An advanced solution could assume virtual memory and paging, in which case the writes on line 3 are additional output blocks and the reads on lines 4 and 5 are additional input blocks. Hence, there would be an additional B(R) blocks of output and an additional

$$B(R) + \sum_{i=1}^{T(R)-1} \lceil \frac{i}{44} rceil$$

blocks of input. Clearly, we can see that we would want a different (disk-aware) algorithm if B(R) exceeds the capacity of main memory.

Question 4

This is also a block-nested loop algorithm, i.e., it iterates two relations in a nested loop that jumps one block at a time. It treats each pair of tuples independently; so, the size of main memory is only important to establish that there is space to read in one block of R, read in one block of S, and maintain one block for an output buffer.

As in previous questions, because tuples should not be split across blocks and because blocks are atomically input and output, we need to be careful when calculating the number of blocks occupied by R, B(R):

$$B(R) = \lceil \frac{T(R)}{\lfloor \frac{4KB}{92B} \rfloor} \rceil = \lceil \frac{10000}{44} \rceil = \lceil \frac{T(R)}{\lfloor \frac{4096}{92} \rfloor} \rceil = 228$$

.

Similarly, we can calculate B(S):

$$B(S) = \lceil \frac{T(S)}{\lfloor \frac{4KB}{48B} \rfloor} \rceil = \lceil \frac{100}{85} \rceil = 2$$

.

We can therefore calculate that Lines 1 and 2 together input 2(228) = 456 blocks.

We see that Line 6 and Line 8 are the only lines that output data to disk. However, we need to copy concatenated tuples to the output buffer, which will have size 92B+48B. Therefore, we can fit

$$\lfloor \frac{4096}{(92 + 48)} \rfloor = 29$$

output tuples on each output block. Since each tuple of R appears exactly once with every tuple of R in the output, the number of output blocks is exactly

$$B(O) = \lceil \frac{T(R) * T(S)}{29} \rceil = \lceil \frac{10000 * 100}{29} \rceil = 34483$$

blocks.