PhD Open lecture: Data Aware Algorithms Autumn 2023

Homework

Exercise 1 Pebble Game for Pyramid Graphs

We focus on computations described by pyramid graphs as depicted in Figure 1. A pyramid graph is obtained by slicing a 2D $n \times n$ grid along its diagonal and orienting edges towards the corner facing the diagonal. We consider the (black) pebble game with the rules seen in the lecture; in particular, one may put a (new) pebble on a node only if all its predecessors are already pebbled.

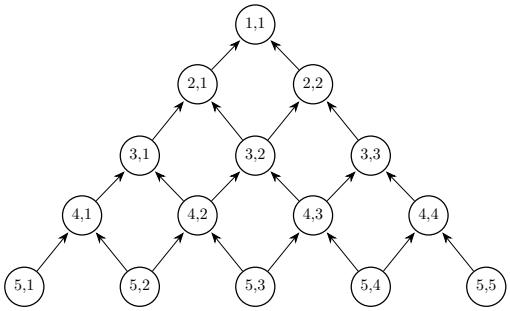


Figure 1: Pyramid graph with $n = 5$ levels

Question 1. Describe a strategy that pebbles the n-level pyramid using only $n + 1$ pebbles (for $n \geq 2$).

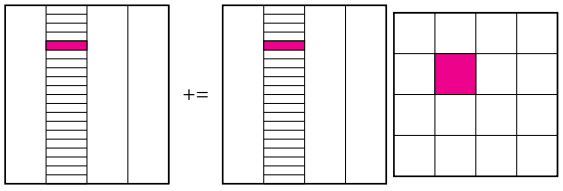
Question 2. Prove that any pebbling strategy in the pebble game for the n-level pyramid uses at least $n + 1$ pebbles (for $n \geq 2$).

Exercise 2 I/O-Efficient Matrix Multiplication

We consider here a computing system with a fast memory of size M and an unlimited (but slower) disk. All data initially reside on disk, but we can only compute on the data held in fast memory. We analyse the performance of an algorithms in terms of data read and written from/to the disk.

To compute the product of two square $n \times n$ matrices, we propose the following partition of the matrices, illustrated below:

- Matrix C is partitioned in block-rows of size $(\sqrt{M} - 1) \times 1$
- Matrix A is partitioned in block-rows of size $(\sqrt{M} - 1) \times 1$
- Matrix B is partitioned in square blocks of size $(\sqrt{M} - 1) \times (\sqrt{M} - 1)$



The sketch of the algorithm is the following. We load a square block B_p of B. We successively load each pair of blocks of C_p and A_p so that we compute the contribution $C_p = C_p + A_p \times B_p$. Only when all computations involving B_p are completed, we remove B_p from the memory and start over with another block of B.

Question 1. Write a proper algorithm following these directions.

Question 2. What is the number of read operations for this algorithm ? Of write operations? Is this algorithm I/O-optimal?

Question 3. In modern systems, read and write operations can happen concurrently, so that a better metric for the I/O cost is $M_{I/O} = \max(reads, writes)$. Do you think this algorithm is optimal for the problem of minimizing $M_{I/O}$? (only simple justification expected)

Exercise 3 Cache Oblivious Matrix Transposition

We consider a simple algorithm to compute the transposition of a $n \times n$ matrix ($B = A^T$):

Algorithm 1: MatrixTanspose(A)

```
for i=1, ..., n do
    for j=1, ..., n do
        B_{i,j} ← A_{j,i}
return B
```

Both matrices A and B are stored in row-major layout (elements in one row are store consecutively, rows are stored one after the other).

Question 1. Compute the I/O complexity of this algorithm in the external memory model, with cache size M and block size B.

Question 2. Design an efficient divide-and-conquer algorithm for this problem (when n is a power of two), and analyse its I/O complexity. Is it an optimal cache-oblivious algorithm?

Exercise 4 Memory-Aware DAG scheduling

We consider a computation modeled as a Directed Acyclic Graph G of tasks: vertices represent computations (=tasks), and an edge $u \to v$ represents a data dependency from task u to task v (u produces a data used as input by v). Edges have weight representing the size of the corresponding data. We use the simple model of computation seen in class: when a task is processed, its inputs are replaced by its outputs. A schedule σ of the graph is defined by $\sigma(u) = t$ if task u is processed at step t.

Question 1. Consider a graph G, and assume we know a schedule σ that minimizes the peak memory for G. We define $\overline{G}$ as the DAG obtained from G by reversing all edges $u \to v$ into $v \to u$. Exhibit an optimal schedule for $\overline{G}$.

Question 2. Consider a tree T of tasks where all edges are oriented towards the root. We assume now that all edge weights are identical and equal to m. Prove that there exists a postorder schedule with minimal memory peak. Give a recursive formula $M(T)$ of the minimum memory peak of the tree T.