Exercise 5.5

Friday, 29 September 2023 18:21

Given that m is much larger than M, we can conclude that going over one row or going over one column requires reading and evicting a significant part of that row/column. Hence, we can state that (in the limit as $\frac{m}{M} \to \infty$), it must be the case that (asymptotically), the number of entries which is not stored in memory before the row/column is starting to be read, goes to m. In other words, upon the start of reading a row/column, we might as well assume memory to be empty

(i) We need to fill m^2 entries of matrix Z. Given that Z is stored in row-major order, and we fill Z in a row-by-row manner, this requires $O\left(\frac{m^2}{B}\right)$ I/Os.

For filling each entry of Z, we need to read for each of the m^2 entries of Z:

one row of X, which given that X is stored in row-major order takes $O\left(\frac{m}{B}\right)$ I/Os.

one column of Y, which given that Y is stored in row-major order takes $O(m)$ I/Os.

In total, this takes $O\left(\frac{m^2}{B} + m^2\left(\frac{m}{B} + m\right)\right) = O(m^3)$ I/Os.

(ii) We need to fill m^2 entries of matrix Z. Given that Z is stored in row-major order, and we fill Z in a row-by-row manner, this requires $O\left(\frac{m^2}{B}\right)$ I/Os.

For filling each entry of Z, we need to read for each of the m^2 entries of Z:

one row of X, which given that X is stored in row-major order takes $O\left(\frac{m}{B}\right)$ I/Os.

one column of Y, which given that Y is stored in column-major order takes $O\left(\frac{m}{B}\right)$ I/Os.

In total, this takes $O\left(\frac{m^2}{B} + m^2\left(\frac{m}{B} + \frac{m}{B}\right)\right) = O\left(\frac{m^3}{B}\right)$ I/Os.

(iii)

$$T_{IO}(t) = \begin{cases} O\left(\frac{M}{B}\right) & \text{if the subproblem of size } t \text{ fits in memory} \\ 8\, T_{IO}\left(\frac{t}{2}\right) + O\left(\frac{t^2}{B}\right) & \end{cases}$$

matrices are square

a.k.a. if $3t^2 + 2(B-1) \leq M$

3 matrices need to fit

probably $\frac{12\left(\frac{t}{2}\right)^2}{B}$

number of subproblems into which original problem is split

size of subproblem

overhead of I/Os outside recursive calls; here: adding smaller matrices

$$\frac{m^2}{B}$$

$$\frac{\left(\frac{m}{2}\right)^2}{B} \quad \frac{\left(\frac{m}{2}\right)^2}{B} \cdots 8\times \quad \frac{\left(\frac{m}{2}\right)^2}{B} \quad \frac{\left(\frac{m}{2}\right)^2}{B} \qquad 8\frac{m^2}{4B} = \frac{2m^2}{B}$$

$$\frac{\left(\frac{m}{4}\right)^2}{B} \quad \cdots 64\times \cdots \quad \frac{\left(\frac{m}{4}\right)^2}{B} \qquad 64\frac{m^2}{16B} = \frac{4m^2}{B}$$

$\vdots$

overhead on level i: $2^i \frac{m^2}{B}$

size on level i: $\frac{m}{2^i} \to 3\left(\frac{m}{2^i}\right)^2 \cancel{+2(B-1)} \leq M$

ignore under tall cache assumption

at base level: $t = \frac{m}{2^k}$ and $t = \sqrt{\frac{M}{3}}$

$\frac{m}{2^k} = \sqrt{\frac{M}{3}}$

$2^k = \frac{m}{\sqrt{\frac{M}{3}}}$

$3\left(\frac{m}{2^i}\right)^2 = M$

$\frac{3m^2}{M} = 2^{2i}$

$2i = \log_2\left(\frac{3m^2}{M}\right)$

$i = \frac{1}{2}\log_2\left(\frac{3m^2}{M}\right) = O\left(\log_2\left(\sqrt{\frac{m^2}{M}}\right)\right)$

$\to O\left(\log_2\left(\frac{m}{\sqrt{M}}\right)\right)$

$\frac{M}{3} = \left(\frac{m}{2^i}\right)^2$

$\frac{m}{2^i} = \sqrt{\frac{M}{3}}$

$2^i = \frac{m}{\sqrt{M/3}}$

makes sense: we need to split problem of size m^2 into $\frac{m^2}{M}$ smaller problems of size M. This gives the given tree height

total overhead $= \sum_{i=0}^{k-1} (2^i)\frac{m^2}{B}$

to-do: learn this sum

$(2^k - 1)\left(\frac{m^2}{B}\right)$

basecases $= 8^k = (2^k)^3$

I/Os = # base cases $\cdot\, O\left(\frac{M}{B}\right)$ + total overhead

$= (2^k)^3\, O\left(\frac{M}{B}\right) + (2^k - 1)\frac{m^2}{B}$

$= \left(\frac{m}{\sqrt{M/3}}\right)^3 O\left(\frac{M}{B}\right) + \left(\frac{m}{\sqrt{M/3}} - 1\right)\frac{m^2}{B}$

$= O\left(\frac{m^3 M}{M^{\frac{3}{2}} B} + \frac{m^3}{\sqrt{M} B}\right)$

$= O\left(\frac{m^3}{B\sqrt{M}}\right) = O\left(\frac{n\sqrt{n}}{B\sqrt{M}}\right)$

4. The spatial locality remains the same in both cases; in both cases, once a block is loaded, its contents will be used in their entirety.
The temporal locality has been improved in the recursive algorithm, however. All accesses to the same data element are done in the same base case of the recursive procedure in the recursive algorithm, whereas the row-by-row algorithm still has to load blocks multiple times.