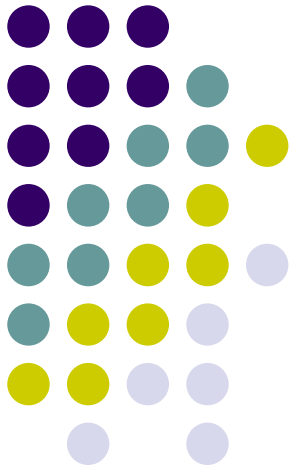
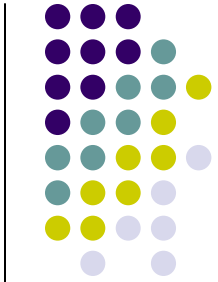
CS256
Applied Theory of Computation

Memory Hierarchy Tradeoffs II

John E Savage

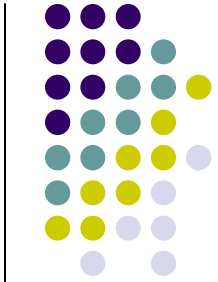


Overview



- Review of the Memory Hierarchy Game
- Review of the Hong-Kung Lower Bound
- Application of the Hong-Kung Bound to Matrix Multiplication

The Memory Hierarchy Game

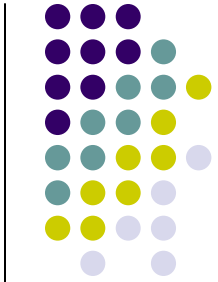


- The game is played on a DAG. The number of pebbles at level $1 \le l \le$ L-1is p_l. p_L unlimited
- (initialization) A level-L pebble can be placed on any input vertex at any time.
- (computation) Level-1 pebble can be placed on (or moved to) vertex whose predecessors have level-1 pebbles.
- (pebble deletion) A pebble can be deleted from a vertex at any time.

The Memory Hierarchy Game

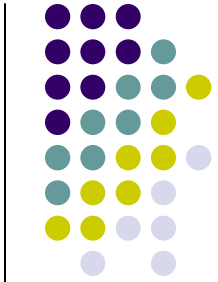
- (goal) A level-L pebble must be on each output vertex at the end of the game.
- (input from level-k) Level-(k-1) pebble can be placed on vertex carrying level-k pebble, $2 \leq k \leq L$
- (output from level-k) Level-(k+1) pebble can be placed on vertex carrying level-k pebble, $1 \leq k \leq L-1$.

The Memory Hierarchy Game



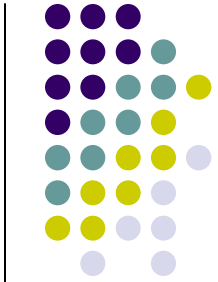
- In the **I/O-limited version** game level-L pebbles are allowed only on inputs and outputs. When L=2, this is the red pebble game; blue pebbles are allowed only on inputs and outputs.
- **Resource vector (RV) $\boldsymbol{p}$** = $(p_1, p_2, \ldots, p_{L-1})$ specifies the amount of *space* used at each level. $T_k^{(L)}(\boldsymbol{p},G,P)$ is the I/O time at level k with RV $\boldsymbol{p}$ on the DAG G using pebbling strategy P for $2 \le k \le L$. The computation time $T_1^{(L)}(\boldsymbol{p},G,P)$ is the number of times vertices are pebbled with level-1 pebbles.

The Memory Hierarchy Game



- A **minimal pebbling strategy** minimizes the number of I/O ops at level L, then at level L-1, all the way down to level 2. Finally it minimizes $T_1^{(L)}(\boldsymbol{p},G,P)$.

I/O Time Relationships



- Let P be strategy to pebble DAG G with RV $\boldsymbol{p}$. Let In(G) and Out(G) be G's inputs & outputs. Then,

$$T_k^{(L)}(\boldsymbol{p},G,P) \geq |In(G)| + |Out(G)| \text{ for } 2 \leq k \leq L$$

$$T_1^{(L)}(\boldsymbol{p},G,P) \geq |V| - |In(G)|$$

Theorem Let $s_k = p_1+p_2+\ldots+p_{k-1}$. Let $T_1^{(2)}(S,G,P)$ & $T_2^{(2)}(S,G,P)$ be the computation and I/O times for a minimal red-blue pebbling of G with S red pebbles.

$$T_k^{(L)}(\boldsymbol{p},G,P)) \geq T_2^{(2)}(s_{k-1},G,P) \text{ for } 2 \leq k \leq L$$

$$T_1^{(L)}(\boldsymbol{p},G,P) \geq T_1^{(2)}(s_{k-1},G,P) \text{ for } 2 \leq k \leq L$$

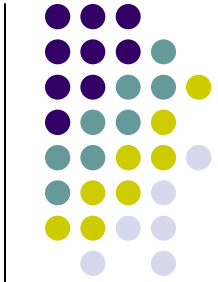
Hong-Kung Lower Bound

Definition The **S-span** of DAG G, ρ(S,G), is the maximum number of vertices of G that can be pebbled with S red pebbles in red pebble game maximized over all initial placements of S red pebbles. (Initialization rule is disallowed.)

Theorem For every pebbling P of G = (V,E) in the red-blue pebble game with S red pebbles, the I/O time used, $T_2^{(2)}(S,G,P)$ satisfies

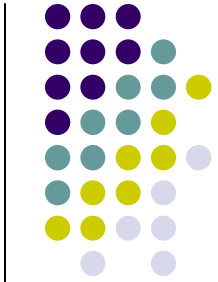
$$\left\lceil T_2^{(2)}(S,G,P)/S \right\rceil \rho(2S,G) \geq |V| - |In(G)|$$

Hong-Kung Lower Bound



- The Hong-Kung bound applies to individual DAGs or types of DAGs.
- Unlike lower bounds for the red pebble game, we can't yet derive I/O complexity lower bounds that apply to all DAGs for a function.
- We now derive a lower bound on I/O complexity for the family of algorithms F_n based on the standard algorithm for matrix-matrix multiplication in which two-input adders are used.
- These algorithms form inner products but don't specify the order in which the additions of inner products are done.

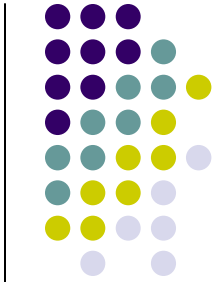
Matrix-Matrix Multiplication



Lemma For every graph in the family F_n of nxn matrix multiplication algorithms computing C = AB, the S-span satisfies $\rho(S,G) \leq 2S^{3/2}$ for $S \leq n^2$.

Proof Let A =$\{a_{i,j}\}$, B =$\{b_{i,j}\}$, C =$\{c_{i,j}\}$ for $1 \leq i,j \leq n$ $c_{i,j} = \Sigma_k\, a_{i,k}\, b_{k,j}$ is associated with the root of an inner product tree. G in F_n has product vertices $a_{i,k}\, b_{k,j}$ and 2-input addition vertices uniquely associated with $c_{i,j}$

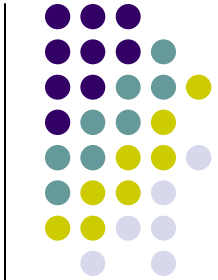
Matrix-Matrix Multiplication



Proof (cont.) Consider an initial placement of S ≤ n^2 pebbles on G of which *r* are on addition or product vertices and S - *r* are on inputs in A or B, which are common to multiple inner product trees. Let *p* be the max no. of product vertices that can be pebbled from the inputs.

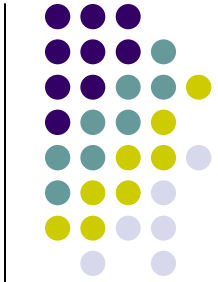
We show that at most *p+r-1* vertices in the addition trees can be pebbled for a total of at most π = *2p+r-1* vertices pebbled.

Matrix-Matrix Multiplication



Proof (cont.) There is no loss in generality in assuming that we pebble the p product vertices before the addition vertices. Thus, we start with $p+r$ pebbles on the vertices of one or more addition trees associated with inner products. These pebbles allow us to pebble vertices in some number t of subtrees. If q pebbles are used to pebble one subtree, the number of vertices pebbled is maximized when the q pebbles are on inputs to the subtree. Since the subtrees are binary, at most $q-1$ vertices can be pebbled. Thus, the $p+r$ pebbles can pebble a most $p+r-t$ vertices in addition trees. This number is largest when $t=1$.

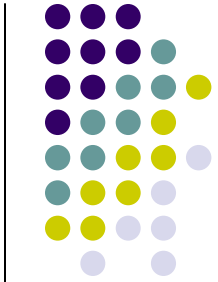
Matrix-Matrix Multiplication



Proof (cont.) We now derive an upper bound on p.

Let $\mathcal{A}$ be 0-1 matrix whose i,j entry is 1 if $a_{i,j}$ carries one of the S pebbles initially. Let $\mathcal{B}$ be defined the same way. Let $\mathcal{C}$ be matrix obtained by multiplying $\mathcal{A}$ and $\mathcal{B}$. Then $p = \Sigma_{i,j}\ \mathcal{C}_{i,j}$ is the number of product vertices that can be pebbled from initial placement of S pebbles. We show that $p \leq \sqrt{S}\ (S\text{-}r)$.

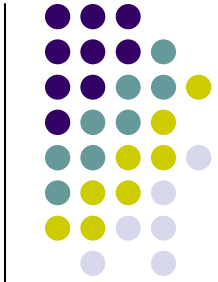
Matrix-Matrix Multiplication



Proof (cont.) Let $\mathcal{A}$ and $\mathcal{B}$ have a and b 1's, where $a+b = S-r$. There are at most a/α rows containing at least α 1's. Since $\mathcal{B}$ has b 1's, at most ab/α 1's in C can be formed by inner products with these rows.

At most S inner products can be formed with sparse rows. These contribute at most αS 1's to p. ($\mathcal{A}$ has a 1's.) Hence, $p = ab/\alpha + \alpha S$. Since α is unknown, we choose it to maximize p, i.e. $\alpha = \sqrt{ab}/S$. Thus, $p \leq 2\sqrt{abS} \leq \sqrt{S}\,(S-r)$ and $\pi = 2p+r-1 \leq 2\sqrt{S}\,(S-r) + r - 1 \leq 2\sqrt{S}\,S = 2\,S^{3/2}$. Q.E.D.

Matrix-Matrix Multiplication



Theorem Let S ≥ 3. For every graph G in Fn computing n x n matrix multiplication C = AB

$$T_1^{(2)}(S,G) = \Omega(n^3)$$

$$T_2^{(2)}(S,G) = \Omega(n^3/\sqrt{S})$$

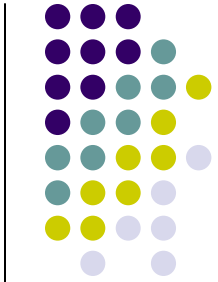
There is a pebbling strategy $\mathcal{P}$ for G satisfying both of the following bounds simultaneously.

$$T_1^{(2)}(S,G,\mathcal{P}) = O(n^3)$$

$$T_2^{(2)}(S,G,\mathcal{P}) = O(n^3/\sqrt{S})$$

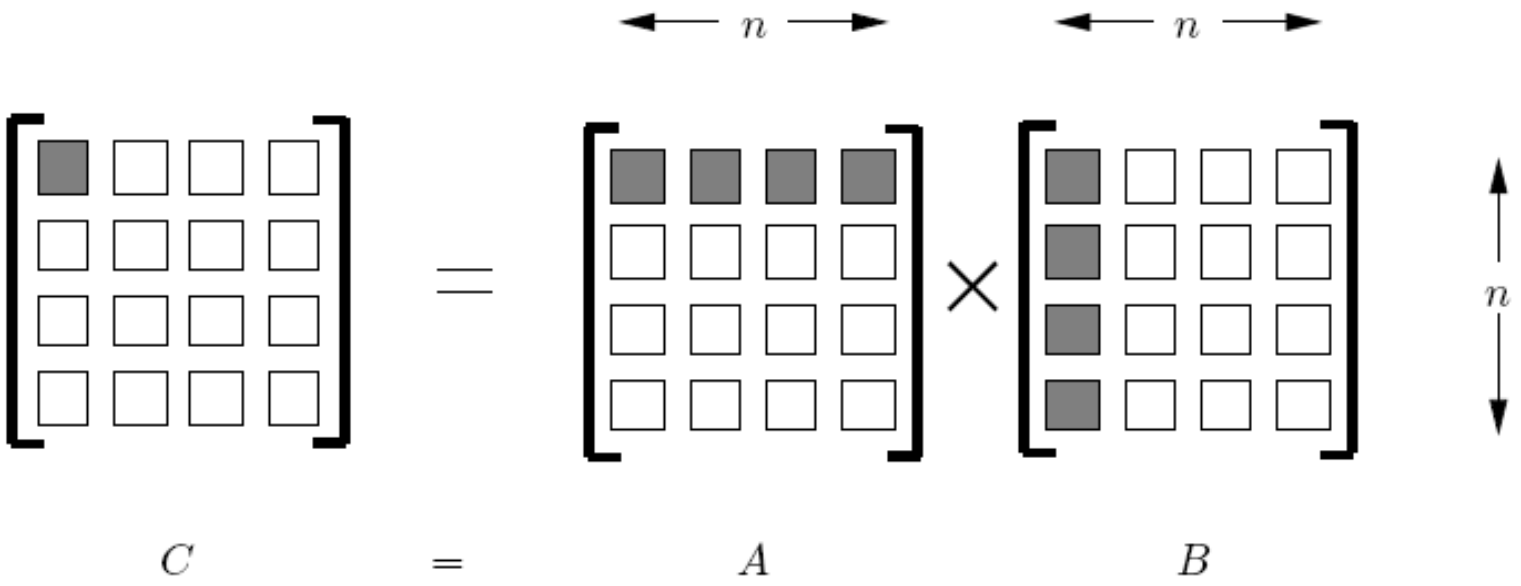
Proof The lower bound follows directly from the Hong-Kung bound. We give an algorithm that achieves the upper bound.

Matrix-Matrix Multiplication

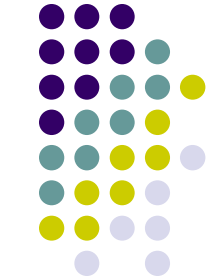


Proof (cont.) Assume that $r = \sqrt{(S/3)}$ divides n. Represent each matrix A, B, and C as an n/r x n/r matrix X, Y, and Z. To compute the product C = AB, form the product Z = XY. This involves inner products of rows of X with columns of Y. Each row of X corresponds to r rows of A and each column of Y corresponds to r columns of B. Treat each set of r rows of A as a row of r x r blocks. Do the same with columns of B.

Matrix-Matrix Multiplication



Matrix-Matrix Multiplication



- **Proof (cont.)** Each block of A or B has at most $r^2 \leq S/3$ entries. To form an inner product of a row of X with a column of Y, read in one block from A and one from B, form their product and store the result if the first such product or add it to the previous result, if not. At most S pebbles is used for this purpose. Since each block of A is involved in *n/r* inner products, each element of A (and B) is involved in *n/r* I/O ops, giving the desired result. Q.E.D