The Red Blue Pebble Game

The red-blue pebble game is played on a DAG.

- **(initialization)** A blue pebble can be placed on any input vertex at any time.
- **(computation)** A red pebble can be placed on (or moved to) any vertex all of whose predecessors have level-1 pebbles.
- **(pebble deletion)** A pebble can be deleted from a vertex at any time.
- **(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 ≤ k ≤ L
- **(output from level-l)** For 2 ≤ k ≤ L, level-k pebble can be placed on vertex carrying

Hong-Kung Lower Bound

$T_k^{(L)}(p,G,P)$ is no. of level-k I/O ops used to pebble G with strategy $P$. $T_k^{(L)}(p,G) = \min_{P} T_k^{(L)}(p,G,P)$ is the I/O complexity of G.

**Definition S-span** of DAG G, $\rho(S,G)$, is the max 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.

**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

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

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 \( q-I \) 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 \( t=1 \).

We now derive an upper bound on \( p \).

Matrix-Matrix Multiplication

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

**Proof** Let \( A = \{a_{i,j}\} \), \( B = \{b_{i,j}\} \), \( C = \{c_{i,j}\} \) for \( 1 \leq i,j \leq n \)
\[ c_{i,j} = \sum_k a_{i,k} b_{k,j} \]
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 \( c_{i,j} \)
Consider an initial placement of \( S \leq n^2 \) 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 \( \pi = 2p+r-1 \) vertices pebbled.

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

Let \( A \) and \( B \) have \( a \) and \( b \) 1’s, respectively. There are at most \( a/\alpha \) rows containing at least \( \alpha \) 1’s. At most \( ab/\alpha \) 1’s in \( C \) can be formed by inner products with these rows. \( B \) has \( b \) 1’s. At most \( S \) inner products can be formed with other rows and these contribute at most \( aS \) 1’s to \( p \). \( A \) has \( a \) 1’s. Hence, \( p = ab/\alpha + aS \).
Since \( \alpha \) is unknown, we choose it to maximize \( p \), which occurs when \( \alpha = \sqrt{ab/S} \). Thus, \( p \leq \sqrt{S} (S-r) \) 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 $S \geq 3$. For every graph $G$ in $F_n$ computing $n \times n$ 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.

Proof (cont.) Assume that $r = \sqrt{(S/3)}$ divides $n$. Represent each matrix $A$, $B$, and $C$ as an $n/r \times n/r$ 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 $r \times r$ blocks. Do the same with columns of $B$.

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