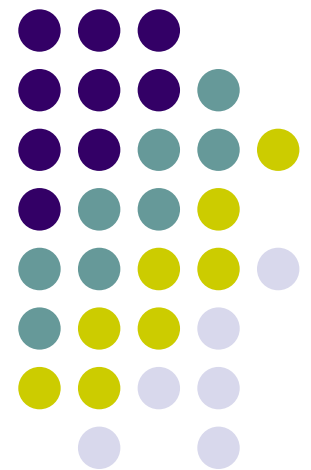


# CS159 Introduction to Computational Complexity

---

## Space-Time Tradeoffs I

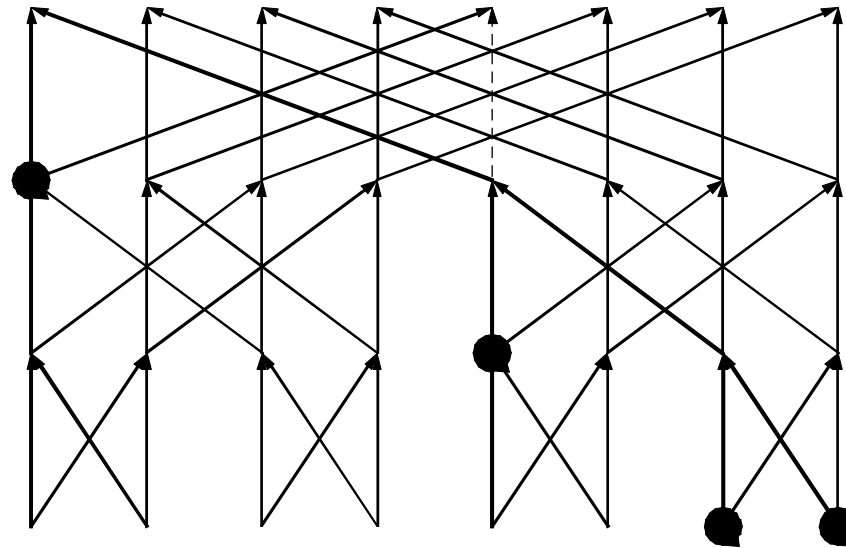


# Rules of the Red Pebble Game



- (**Initialization**) A pebble can be placed on an input vertex at any time.
- (**Computation Step**) A pebble can be *placed on or moved to* any non-input vertex if all of its immediate predecessors carry pebbles.
- (**Pebble Deletion**) A pebble can be removed at any time.
- (**Goal**) Each output vertex must be pebbled at least once.

# Pebbling the FFT Graph on Eight Inputs

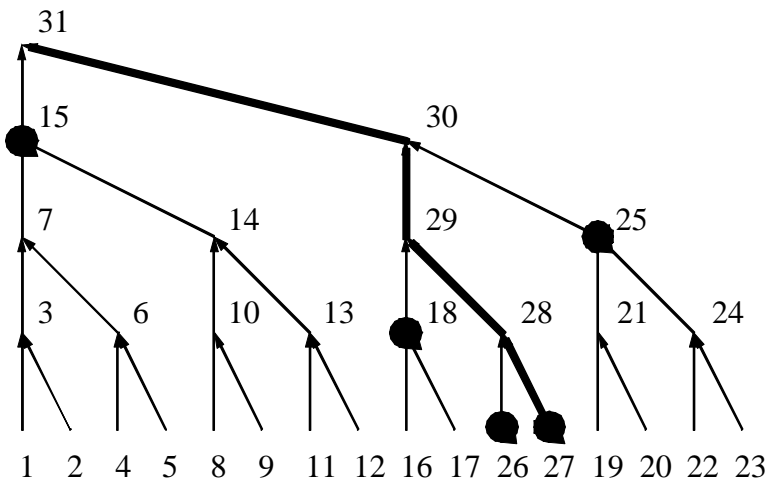
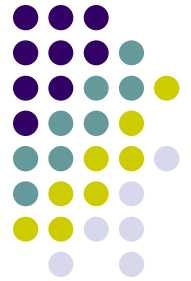




# Pebbling Strategy

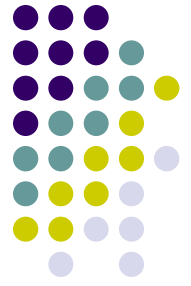
- A **pebbling strategy** determines sequence of rules invoked on vertices of a graph. A strategy uses **space**  $S$  if it uses at most  $S$  pebbles. It uses **time**  $T$  if  $T$  initialization and computation steps are done.
- The **minimum space**  $S_{min}$  to pebble a graph  $G$  is the smallest space of any strategy that pebbles  $G$ .
- The FFT graph exhibits a tradeoff between space and time. The time required when the minimum space is used is strictly more than that required when more space is available.

# Pebbling Balanced Binary Tree



**Theorem** The complete balanced binary on  $n = 2^k$  inputs has  $S_{min} = k+1 = \log_2 n + 1$ . It can be pebbled in  $T = 2n-1$  steps, but no fewer.

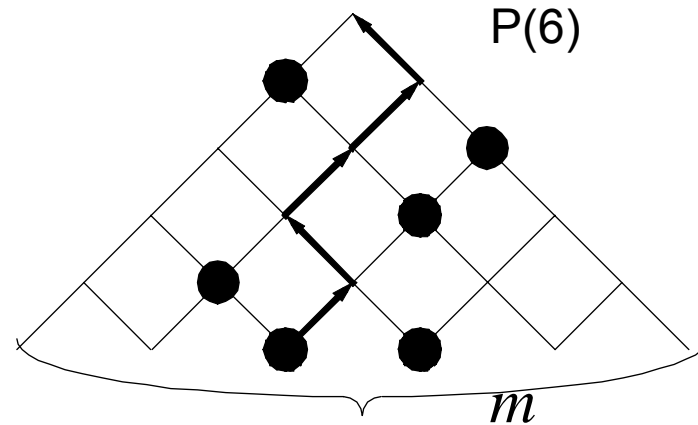
# Pebbling Balanced Binary Tree



**Proof** Initially each path from an input to the output is free of pebbles. Finally, (a pebble is on the output), all paths contain a pebble. There is a last time at which a path is open. Each path has  $k+1$  vertices. When placing a pebble on last input, all paths from other inputs to vertices on the path carry 1 pebble. Thus,  $S_{min} \geq k+1$ .  $T = 2n-1$ ,  $S_{min} = k+1$  by induction. Do left subtree in  $2(n/2)-1$  steps, leave pebble at its root, do right subtree with  $k$  pebbles in same time.

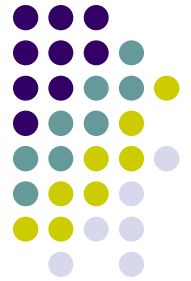


# Pebbling the Pyramid Graph



**Theorem**  $S_{min} = m$  for the pyramid graph  $P(m)$  on  $m$  inputs. It can be pebbled in  $n = m(m+1)/2$  steps where  $n$  is the number of vertices in the graph with  $S_{min}$  pebbles.

# Pebbling the Pyramid Graph

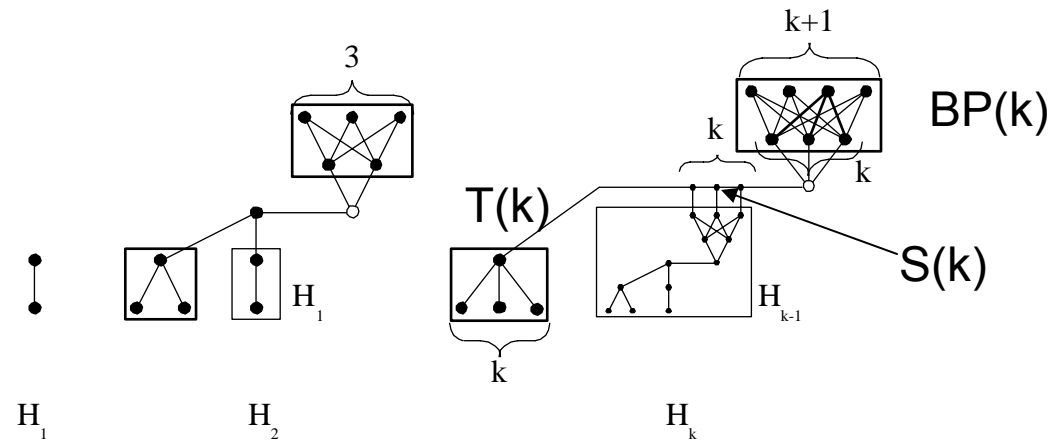


**Proof** The last open path argument can be used to show that  $S_{min} \geq m$ . To pebble  $P(m)$  with  $m$  pebbles, place pebbles on all inputs. Move leftmost pebble up one level. Now all vertices one level up can be pebbled using  $m-1$  pebbles. Repeat at subsequent levels. Each vertex is pebbled once.

**Note:**  $S_{min}$  is about the square root of its number of vertices,  $n$ , much larger than for binary tree.



# Extreme Tradeoffs



- $H_k$  is constructed from a copy of  $H_{k-1}$ , a  $k$ -input tree,  $T(k)$ , a "spine",  $S(k)$ , of  $k$  vertices connected to the  $k$  outputs of  $H_{k-1}$ , an "open" vertex, and a bipartite graph,  $BP(k)$ , with  $k$  inputs and  $k+1$  outputs.  $S_{min}(H_i) = i$  for  $i = 1, 2$ .



# Extreme Tradeoffs

**Lemma**  $S_{min}(H_k) = k$  for  $k \geq 3$ .

**Proof** Since tree  $T(k)$  needs  $k$  pebbles,  $S_{min} \geq k$ . We show  $k$  pebbles suffice assuming outputs of  $H_{k-1}$  can be pebbled in succession with  $k-1$  pebbles.

Advance pebble to tree output, use  $k-1$  pebbles on  $H_{k-1}$  to pebble outputs and advance pebbles along spine. Advance pebble to open vertex. Put  $k$  pebbles on  $BP(k)$ , pebble one output. Repeat for each additional output.



# Extreme Tradeoffs

**Lemma**  $H_k$  has  $N(k) = 2k^2 + 5k - 6$  vertices,  $k \geq 2$

**Proof Basis:**  $N(2) = 12 = 2(2)^2 + 5(2) - 6$

$$\begin{aligned} N(k) &= N(k-1) + (k+1) + k + 1 + (k + k+1) \\ &= N(k-1) + 4k + 3 \end{aligned}$$

$$N(k) = 2(k-1)^2 + 5(k-1) - 6 + 4k + 3 = 2k^2 + 5k - 6$$



# Extreme Tradeoffs

**Lemma**  $H_k$  requires  $T(k) \geq (k+1)!$  steps to pebble with  $S_{\min}(H_k) = k$  pebbles but can be pebbled in  $N(k)$  steps with  $k+1$  pebbles.

**Proof** When  $S_{\min}(H_k) = k$ , re-pebble  $H_{k-1}$   $(k+1)$  times. (Pebbling  $BP(k)$ , removes all pebbles from  $H_{k-1}$ .) Thus,

$$T(k) \geq (k+1)T(k-1) \geq (k+1)(k)(k-1)\dots(3)T(1) = (k+1)!$$

*Inductive Hypothesis:* When  $k+1$  pebbles used, assume all outputs of  $H_{k-1}$  can be pebbled in succession using  $k+1$  pebbles without re-pebbling any vertices.

We advance  $k$  pebbles to the inputs of the  $BP(k)$  without re-pebbling any vertices. The remaining pebble is used to pebble outputs of the  $BP(k)$  in succession.

- **Note:**  $(k+1)!$  exponential in  $N(k)$ . **Extreme tradeoff!**