1. Time Vs. Space

A Turing machine can visit at most \( t(n) \) cells in \( t(n) \) time. This implies that any deterministic \( t(n) \)-time bounded machine can be simulated by a \( t(n) \)-space bounded machine. Can we improve this space bound asymptotically? In other words, is there a language in \( \text{DTIME}(t(n)) \) that takes less than \( t(n) \) space? We will show the following theorem.

**Theorem 1.** \( \text{DTIME}(t(n)) \subseteq \text{DSPACE}(t(n)/\log t(n)) \).

Using space hierarchy, we have the following straightforward corollary.

**Corollary 1.** \( \text{DTIME}(t) \subseteq \text{DSPACE}(t) \)

**Proof.** We know by space hierarchy theorem that \( \text{DSPACE}(t(n)/\log t(n)) \subseteq \text{DSPACE}(t(n)) \), which implies, \( \text{DTIME}(t(n)) \subseteq \text{DSPACE}(t/\log t) \subseteq \text{DSPACE}(t) \). \( \square \)

To prove the above theorem, we will consider block respecting Turing machines. Suppose \( M \) is a \( t(n) \) time bounded \( k \)-tape Turing machine. Let \( 1 \leq b(n) \leq t(n)/2 \) and \( a(n) = t(n)/b(n) \). Divide the computation of \( M \) into \( a(n) \) time segments where each segment has \( b(n) \) steps. Since \( M \) os \( t(n) \) time bounded, it visits atmost \( t(n) \)-cells on each tape. Now divide each tape into \( a(n) \) segments and thus each segment has \( b(n) \) cells.

**Definition 1.** \( M \) is \( b(n) \)-block respecting if every tape head of \( M \) crosses a block boundary at time \( c.b(n) \), where \( c \) is an integer.

We will use the following claim without proof.

**Claim 1.** If \( M \) is \( t(n) \) time bounded \( k \)-tape Turing machine and \( 1 \leq b(n) \leq t(n)/2 \), then there is \( (k+1) \)-tape, \( b(n) \) block respecting machine \( M' \) that runs in time \( O(t(n)) \), and \( L(M) = L(M') \).

Let \( L \) be a language in \( \text{DTIME}(t(n)) \). Set \( b(n) = t^{2/3}(n) \). Let \( M \) be a \( b(n) \) block respecting Turing machine that decides \( L \) in time \( t(n) \). Set \( a(n) = t^{1/3}(n) \). Define a computation graph \( G \) of \( M \) in the following way. \( G = (V, E) \) where \( V = \{1, 2, \ldots, a(n)\} \). For every \( i \geq 1 \), \( (i, i + 1) \in E \). Further, \( (i, j) \in E \) if for some tape, the tape head is in some block \( l \) during time segment \( i \) and then first time that tape head is again in block \( l \) in time segment \( j \). Note that \( j \) could be \( i + 1 \). This yields a a multigraph and the maximum in-degree of a vertex is \( (k + 1) \). We add some dummy tape heads to make in-degree uniform \( (d = k + 1) \) for each node.

**Pebbling Game.** We now describe a pebbling game that will eventually be used in our space efficient simulation of \( M \). Suppose we have a directed acyclic graph \( G = (V, E) \). We want to place a pebble on a special vertex \( w \in V \) with the minimum number of pebbles. The rules for the game are as follows:

- We can remove a pebble from a vertex at any time.
- We can place a pebble at a vertex if all its predecessors are pebbles.
We can always pebble a graph with $n$ nodes with $n$ pebbles. Is it possible to pebble with much less than $n$ pebbles? Let $G$ be a graph with in degree $d$ and $G_d$ be the set of directed acyclic graphs (DAG) with in-degree $d$.

**Theorem 2.** Any graph $G$ in $G_d$ can be pebbled with $(c'n/\log n)$ pebbles for some constant $c$, where $n$ is the number of vertices in $G$.

We say that a graph requires $P$ pebbles, if there is a vertex $w \in G$ and $w$ can not be pebbled with $P - 1$ pebbles. Let $R(P) = \min\{\text{card}(E(G)) \mid G \in G_d \text{ and } G \text{ requires } P \text{ pebbles}\}$.

We will show that $R(P) > c'P\log P$ for some $c' > 0$. Because of the bound on in degree this is same as: $\min\{\text{card} V(G) : G \in G_d \text{ and } G \text{ requires } P \text{ pebbles}\} > (c'/d)P\log P$. This implies that for graphs with $(c'/dP)\log P$ vertices, $P$ pebbles are sufficient. This in turn implies the theorem.

To prove that $R(P) \geq c'P\log P$, it is sufficient to show that $R(P) = 2R(P/2 - d) + P/4d$.

We will first show that every graph that requires $P$ pebbles can be divided the graph into two sub graphs each requiring $P/2 - d$ pebbles.

Let $G = (V, E)$ that requires $P$ pebbles. Let $V_1 = \text{set of all vertices that can be pebbled with } \leq P/2 \text{ pebbles}$. Let $G_1$ be the graph induced by $V_1$. $G_1 = (V_1, E_1)$. Let $V_2 = V - V_1$, and $G_2 = (V_2, E_2)$ is the subgraph induced by $V_2$. Let $E_3 = E - (E_1 \cup E_2)$. By definition, every vertex in $V_2$ requires $P/2$ pebbles.

**Claim 2.** If we play the pebbling game in $G_2$, then we require $(P/2 - d)$ pebbles.
Proof. We will prove this by contradiction by proving that if every vertex of\( G_2 \) requires less than \((P/2 - d)\) pebbles (Let us say at most \((P/2 - d - 1)\) pebbles), then \( P - 1 \) pebbles will be sufficient for \( G \) which will be in contradiction to our assumption that \( P \) pebbles are required for \( G \). Let \( S \) be a strategy that can pebble very vertex in \( G_2 \) using \( P/2 - d - 1 \) pebbles. By the definition of \( G_1 \), there is a strategy \( S' \) that can pebble any vertex in \( G_1 \) using \( P/2 \) pebbles. Consider the following strategy to pebble \( G \): This strategy divides its pebbles set into two parts. First part contains \( P/2 + d \) pebbles and the second part contains \( P/2 - d - 1 \) pebbles. If we have to pebble a vertex from \( G_1 \), then by definition \( P/2 \) pebbles are sufficient. Suppose the target vertex is in \( G \). Now follow the strategy \( S \) using pebbles from the second part. Say this strategy places a pebble in \( v \in V_2 \). We may not be able to place a pebble on \( v \) immediately because \( v \) may contain some predecessors in \( V_1 \). As the indegree is \( d \), \( v \) has \( d' \leq d \) predecessors on \( V_1 \). Let \( \{w_1, w_2, \ldots, w_{d'}\} \) are predecessors of \( v \) that are in \( V_1 \). First pebble \( w_1 \) using strategy \( S' \) and using pebbles from the first part. For this we need at most \( P/2 \) pebbles. Keep the pebble on \( w_1 \) and remove all pebbles. Now pebble \( w_2 \) using strategy \( S' \) using the pebbles from the first part. This can be done using at most \( P/2 \) pebbles. Since we can not use the pebble that is already placed on \( w_1 \), by now we may have used \( P/2 + 1 \) pebbles and placed pebbles on both \( w_1 \) and \( w_2 \). Now keep the pebble on \( w_2 \) and pebble \( w_3 \) using strategy \( S' \). Repeat this process. At the end of this each of \( w_1, w_2, \ldots, w_{d'} \) has a pebble on it and we have used at most \( P/2 + d \) pebbles. Now we can place a pebble on \( v \) as per the strategy \( S \) and remove all pebbles from \( w_1, \ldots, w_{d'} \). Since strategy \( S \) uses at most \( P/2 - d - 1 \) pebbles, we can pebble any vertex in the graph with \( P/2 - 1 \) pebbles. This is a contradiction.

Claim 3. If we only pebble \( G_1 \), we require \((P/2 - d)\) pebbles.

Proof. We will first show that there exist a vertex \( v \) in \( V_2 \) such that all its predecessors are in \( V_1 \). Suppose not. Consider a vertex \( w \). By our assumption, it has a predecessor \( w' \) in \( V_2 \). Now mark \( w \) and consider \( w' \). By the assumption, at has a predecessor \( w'' \) in \( V_2 \). Mark \( w' \) and consider \( w'' \). Repeat this process. Since there are finitely many vertices in \( V_2 \), this process must return to a vertex that is already marked. This means that \( G \) has a cycle. However, \( G \) is a DAG.

Assume that \( G_1 \) requires \((P/2 - d - 1)\) pebbles. Let \( v \) be a vertex from \( V_2 \) that has all its predecessors in \( V_1 \). Assume that its predecessors are \( w_1, w_2, \ldots, w_l \) \((l \leq d)\). Consider the following strategy to pebble \( v \): Pebble \( w_1 \) using \( P/2 - d - 1 \) pebbles. Keep the pebble on \( w_1 \) and remove all other pebbles. Now pebble \( w_2 \). Repeat this till all of \( w_1, \ldots, w_l \) have pebbles. Total number of pebbles used for this is \( P/2 - d + l - 2 \). Now place a pebble on \( v \). Thus we can pebble \( v \) using \( P/2 - d + l - 1 \) pebbles. Since \( l \leq d \), we can pebble \( v \) using \( P/2 - 1 \) pebbles. However, by the definition of \( V_2 \), every vertex in \( v_2 \) requires \( P/2 \) pebbles. This is a contradiction.

So for we have seen that \( R(P) \geq 2R(P/2 - d) \). In the next lecture we will prove prove that the inequality \( R(P) \geq 2R(P/2 - d) + P/4d \) holds.