Problem 1. [10] Complete binary trees with $N$ inputs and $N$ outputs, where $N = 2^n$ for some $n \geq 0$, were described in class & Notes. In this problem we consider complete ternary trees with $N$ inputs and $N$ outputs, where $N = 3^n$. (So the root “switch-node” has degree 3, and the other switch-nodes each have degree 4.) The following figure shows a ternary tree with 3 inputs and 3 outputs.

(a) Find a closed-form expression for the diameter of the ternary tree.

Solution. The diameter is $2 \log_3 N + 1$.

(b) Prove that your expression is correct using induction. Hint: Your induction hypothesis should prove an expression for the length of a path from an input to the root node.

Solution. We proceed by induction on $n$. Let $P(n)$ be the proposition that, in a complete ternary tree with $N = 3^n$ inputs and outputs:

- the number of switches in the path from any input to the root node, or from the root node to an output is $\log_3 N + 1$, and
- the diameter in the tree is $2 \log_3 N + 1$. 

Copyright © 2004, Eric Lehman & F.T. Leighton. All rights reserved.
Base case \((n = 0)\): A complete ternary tree with \(N = 3^0 = 1\) input and 1 output consists of those two input/output nodes plus a single switch. The only path from the input to the root traverses 1 switch, which equals \(\log_3 N + 1 = 0 + 1\). Similarly, the only path from the root to the output traverses 1 switch. The only path from the input to the output traverses 1 switch, which equals \(2 \log_3 N + 1 = 0 + 1\). Thus \(P(0)\) is true.

Inductive step. Now assume \(P(n)\) for \(n \geq 0\) in order to prove \(P(n+1)\). A complete ternary tree \(T_0\) with \(3^{n+1}\) inputs and \(3^{n+1}\) outputs is constructed from three complete ternary trees \(T_1, T_2, T_3\) with \(3^n\) inputs and \(3^n\) outputs by connecting their root nodes to a single new root node \(r_0\).

By the inductive hypothesis, the length of a path from an input in \(T_1\) to its root \(r_1\) is \(\log_3 3^n + 1\). This root node is connected to the new root \(r_0\) by two edges \((r_0, r_1)\) and \((r_1, r_0)\). Thus the path from an input in \(T_1\) to \(r_0\) traverses \((\log_3 3^n + 1) + 1 = (n + 1) + 1 = \log_3 3^{n+1} + 1\) switches, as required. This same argument applies to the trees \(T_2\) and \(T_3\).

If an input and an output belong to \(T_i\) then, by the inductive hypothesis, the shortest path between them traverse at most \(2 \log_3 3^n + 1\) switches. For an input and output in different subtrees, say \(T_1\) and \(T_2\), the shortest path between them consists of the path from the input to \(r_1\), the edges \((r_1, r_0)\) and \((r_0, r_2)\), and the path from \(r_2\) to the output. This path traverses \((\log_3 3^n + 1) + 1 + (\log_3 3^n + 1) = 2(\log_3 3^n + 1) + 1 = 2 \log_3 3^{n+1} + 1\), as required. Thus the maximum distance from any input to any output is \(2 \log_3 3^{n+1} + 1\). This proves \(P(n+1)\).

By the principle of induction \(P(n)\) is true for all \(n \geq 0\).

Problem 2. [35] This problem explores some properties of butterfly networks that were mentioned in the lectures. Let \(B_n\) denote the butterfly network with \(N = 2^n\) inputs and \(N\) outputs.

(a) Let \(v\) be a vertex whose distance to the outputs is \(d\), that is, \(v\) is at level \(n - d\). Let \(b = b_1 \ldots b_n\) denote the binary number associated with vertex \(v\). Let \(T_v\) denote the set of all output vertices whose binary numbers are identical to \(b\) in the first \(n - d\) bits. Prove that:

1. \(T_v\) is the set of output vertices that are reachable by a path starting at \(v\).
2. The path from \(v\) to any vertex in \(T_v\) is unique.

Hint: Assume that \(n\) is fixed and use induction on \(d\).

Solution. By induction on \(d\). Let \(P(d)\) be the predicate that for any vertex \(v\) at distance \(d\) from the outputs,

1. there are paths from \(v\) to precisely the output vertices in \(T_v\), and
2. each such path is unique.
Problem Set 5

**Base case** \((d = 0)\): Let \(v\) be a vertex whose distance to the outputs is 0 and with binary number \(b = b_1 \ldots b_n\). There are no edges outbound from vertex \(v\) so the only output reachable from \(v\) is itself. The set \(T_v\) consists of all output vertices whose binary numbers are identical to \(b\) in all \(n\) bits. Since all output vertices have unique binary numbers, \(T_v\) consists only of vertex \(v\) itself. Since there is a length 0 path from \(v\) to itself, claim 1 is proven. Since this length 0 path is unique, claim 2 is proven. Thus \(P(0)\) is true.

**Inductive step.** Now assume \(P(d)\) for \(d \geq 0\) in order to prove \(P(d + 1)\).

Let \(v\) be a vertex whose distance to the outputs is \(d + 1\) and with binary number \(b = b_1 \ldots b_n\). Since \(v\) is in level \(n - d - 1\), the definition of the butterfly network implies that \(v\) has outbound edges to two vertices, say \(w\) and \(x\), in level \(n - d\). The vertices \(w\) and \(x\) have binary numbers

\[
    b_1 b_2 \ldots b_{n-d-1} 0 b_{n-d+1} \ldots b_n \quad \text{and} \quad b_1 b_2 \ldots b_{n-d-1} 1 b_{n-d+1} \ldots b_n,
\]

respectively.

Since the distance from both \(w\) and \(x\) to the outputs is \(d\), by the inductive hypothesis they can reach the outputs \(T_w\) and \(T_x\) respectively. The set \(T_w\) consists of all output vertices whose binary numbers are equal to \(b\) in the first \(n - d - 1\) bits and whose \(n - d\)th bit is a 0. Similarly, the set \(T_x\) consists of all output vertices whose binary numbers are equal to \(b\) in the first \(n - d - 1\) bits and whose \(n - d\)th bit is a 1. The set \(T_w \cup T_x\) consists of all output vertices whose binary numbers are identical to \(b\) in the first \(n - (d + 1)\) bits, i.e., \(T_v = T_w \cup T_x\). Since the output vertices that are reachable from \(v\) are precisely those that are reachable from \(w\) or \(x\), we have shown that the outputs reachable from \(v\) are precisely those in \(T_v\). Furthermore, \(T_w \cap T_x = \emptyset\), so the only path from \(v\) to an output in \(T_w\) must start with the edge \((v, w)\); a similar claim holds for outputs in \(T_x\). By the inductive hypothesis, the paths from \(w\) and \(x\) to the outputs are unique, and thus the paths from \(v\) to the outputs are unique. Thus \(P(d + 1)\) is true.

**(b)** Let \(v\) be a vertex at level \(i\) of the butterfly network. Let \(b = b_0 \ldots b_n\) denote the binary number associated with vertex \(v\). Let \(S_v\) be the set of all input vertices whose binary numbers are equal to \(b\) in the last \(n - i\) digits. Prove that \(S_v\) is the set of input vertices that have a path to \(v\).

**Solution.** By induction on \(i\). Let \(P(i)\) be the predicate that for any vertex \(v\) at level \(i\), there are paths from precisely the input vertices in \(S_v\) to vertex \(v\).

**Base case** \(i = 0\): Let \(v\) be a vertex at level 0 and with binary number \(b = b_1 \ldots b_n\). Then \(v\) is an input vertex. There are no edges inbound to vertex \(v\) so the only input reachable from \(v\) is itself. The set \(S_v\) consists of all output vertices whose binary numbers are identical to \(b\) in all \(n\) bits. Since all input vertices have unique binary numbers, \(S_v\) consists only of vertex \(v\) itself. Since there is a length 0 path from \(v\) to itself, \(P(0)\) is true.
**Inductive step.** Now assume $P(i)$ for $i \geq 0$ in order to prove $P(i + 1)$. Let $v$ be a vertex at level $i + 1$ and with binary number $b = b_1 \ldots b_n$. Since $v$ is in level $i + 1$, the definition of the butterfly network implies that $v$ has inbound edges from two vertices, say $w$ and $x$, in level $i$. The vertices $w$ and $x$ have binary numbers

$$b_1 b_2 \ldots b_{i-1} 0 b_{i+1} \ldots b_n \quad \text{and} \quad b_1 b_2 \ldots b_{i-1} 1 b_{i+1} \ldots b_n,$$

respectively.

Since the level of both $w$ and $x$ is $i$, by the inductive hypothesis they are reachable from the inputs $S_w$ and $S_x$ respectively. The set $S_w$ consists of all output vertices whose binary numbers are equal to $b$ in the last $n - i - 1$ bits and whose $i$th bit is a 0. Similarly, the set $S_x$ consists of all output vertices whose binary numbers are equal to $b$ in the last $n - i - 1$ bits and whose $i$th bit is a 1. The set $S_w \cup S_x$ consists of all input vertices whose binary numbers are identical to $b$ in the last $n - (i + 1)$ bits, i.e., $S_v = S_w \cup S_x$. Since the input vertices that can reach $v$ are precisely those that can reach $w$ or $x$, we have shown that the inputs that can reach $v$ are precisely those in $S_v$. Furthermore, $S_w \cap S_x = \emptyset$, so the only path from an input in $S_w$ to $v$ must end with the edge $(w, v)$; a similar claim holds for inputs in $S_x$. Thus $P(i + 1)$ is true.

(c) Let $v$ be a vertex at level $i$ of the butterfly network. Prove that there is a path from exactly $2^i$ input vertices to $v$ and a path from $v$ to exactly $2^{n-i}$ output vertices.

**Solution.** Let $b$ be the binary number for vertex $v$. By parts (a) and (b), the set of input vertices that can reach $v$ is $S_v$ and the set of output vertices reachable from $v$ is $T_v$. The set $S_v$ consists of all input vertices whose binary numbers match $b$ in the last $n - i$ bits, but the first $i$ bits can be arbitrary. There are $2^i$ ways to set these bits, so there are $2^i$ inputs in $S_v$. The set $T_v$ consists of all output vertices whose binary numbers match $b$ in the first $i$ bits, but the last $n - i$ bits can be arbitrary. There are $2^{n-i}$ ways to set these bits, so there are $2^{n-i}$ outputs in $T_v$.

(d) Let $n$ be an arbitrary positive integer and let $N = 2^n$. Show that the congestion of an $N$-input $N$-output butterfly network is at most $\sqrt{N}$ if $n$ is even and $\sqrt{N/2}$ if $n$ is odd.

**Solution.** For the butterfly network, there is a unique path from each input to each output, so the congestion is: the maximum number of messages passing through a vertex for any matching of inputs to outputs.

Let $v$ be an arbitrary vertex at some level $i$. Let $S_v$ be the set of inputs that can reach vertex $v$. Let $T_v$ be the set of outputs that are reachable from vertex $v$.

By part (c), we have $|S_v| = 2^i$ and $|T_v| = 2^{n-i}$. The number of inputs in $S_v$ that are matched with outputs in $T_v$ is at most $\min \{2^i, 2^{n-i}\}$. To obtain an upper-bound on the congestion of the network, we need to find the maximum value of $\min \{2^i, 2^{n-i}\}$, where the maximum is taken over all $i$. The maximum value is achieved when $2^i$ and $2^{n-i}$ are as equal as possible. If $n$ is even, these two quantities are equal when $i = n/2$, hence the maximum congestion is

$$2^{n/2} = N^{1/2} = \sqrt{N}.$$
If \( n \) is odd, the maximum value is achieved when \( i = (n - 1)/2 \), hence the maximum congestion is
\[
2^{(n-1)/2} = (2^{n-1})^{1/2} = (N/2)^{1/2} = \sqrt{N/2}.
\]

(e) Let \( \pi \) be the permutation of \( \{0, 1, \ldots, 2^n - 1\} \) where \( \pi(k) \) is defined to be the integer whose \( n \)-bit binary representation is the reversal of the \( n \)-bit binary representation of \( k \). (For example, if \( n = 4 \), then \( \pi(14) = 7 \) because the binary representation of 14 is 1110 and its reversal, 0111 is the binary representation of 7.) For simplicity, assume \( n \) is even, and identify a switch that has \( \sqrt{N} \) paths going through it when the paths are chosen to achieve input-output mapping \( \pi \). Conclude that the congestion of \( B_n \) is exactly \( \sqrt{N} \) when \( n \) is even.

**Solution.** From part d), the congestion of \( \sqrt{N} \) can be achieved only at a vertex at level \( n/2 \).

For a binary number \( x \), Let \( x^r \) denote reversal of \( x \). Now consider a vertex \( v \) at level \( n/2 \) and let \( k_1k_2 \) be the binary number associated with the vertex, where \( k_1 \) is the first \( n/2 \) bits and \( k_2 \) is the last \( n/2 \) bits. Furthermore, let \( S_v \) and \( T_v \) be the sets of inputs and outputs that \( v \) is connected to respectively. From parts b) and c), we have:

1. \( S_v = \{w | w \) is an input and last \( n/2 \) bits of \( w \) is \( k_2 \} \).
2. \( T_v = \{w | w \) is an output and first \( n/2 \) bits of \( w \) is \( k_1 \} \).

The vertex \( v \) has load \( \sqrt{N} \) if and only if all packets from \( S_v \) have destinations in \( T_v \). Equivalently, the necessary and sufficient condition is:

\[
\forall x \in S_v, \quad \pi(x) \in T_v \quad \Leftrightarrow \quad x^r \in T_v \quad \Leftrightarrow \quad k_2 = k_1^r
\]

The last step is true because all \( x \in S_v \) end with \( k_2 \) and all \( x \in T_v \) start with \( k_1 \).

Therefore, the set of vertices with load \( \sqrt{N} \) are those vertices \( v \) at level \( n/2 \) with the last \( n/2 \) bits being the reversal of its first \( n/2 \) bits. For example, consider the vertex \( v \) at level \( n/2 \) with binary index \( \underbrace{100\ldots0}_n \underbrace{0\ldots001}_{n/2} \). Any packet from input in the form \( \underbrace{z\ldots0}_n \underbrace{0\ldots001}_{n/2} \) has destination \( \underbrace{100\ldots0}_n z^r \), where \( z \) is any \( n/2 \)-bit number. So all such packets will pass through \( v \), and there are \( 2^{n/2} = \sqrt{N} \) of them, giving \( v \) load \( \sqrt{N} \).

As a sanity check, there are a total of \( \sqrt{N} \) \( n \)-bit numbers in the form \( kk^r \). So there are \( \sqrt{N} \) vertices at level \( n/2 \) with load \( \sqrt{N} \), giving us a total of \( N \) packets.
Problem 3. [15] Which of the following relations are equivalence relations? For those that are equivalence relations, what are the equivalence classes? For those that are not, give an example of how they fail.

(a) \( R := \{(x, y) \in W \times W \text{ s.t. the words } x \text{ and } y \text{ start with the same letter}\} \) where \( W \) is the set of all words in the 2001 edition of the Oxford English dictionary.

Solution. \( R \) is an equivalence relation since it is reflexive, symmetric, and transitive. The equivalence class of \( x \) with respect to \( R \) is the set \([x]_R = \text{the set of words } y \text{ such that } y \text{ has the same first letter as } x\). There are 26 equivalence classes, one for each letter of the English alphabet.

(b) \( S := \{(x, y) \in W \times W \text{ s.t. the words } x \text{ and } y \text{ have at least one letter in common}\} \).

Solution. \( S \) is reflexive and symmetric, but it is not transitive. Therefore, \( S \) is not an equivalence relation. For example, let \( w_1 \) be the word “scream,” let \( w_2 \) be the word “and,” and let \( w_3 \) be the word “shout.” Then \( w_3Sw_1 \), and \( w_1Sw_2 \), but it is not the case that \( w_3Sw_2 \).

(c) \( R := \{(x, y) \in \mathbb{R} \times \mathbb{R} \text{ s.t. } \exists n \in \mathbb{Z} y = 10^n x\} \).

Solution. \( R \) is reflexive, since we can set \( n = 0 \). \( R \) is symmetric because if \( y = 10^k x \), for some \( k \in \mathbb{Z} \), then \( x = 10^{-k} y \), so if \( (x, y) \in R \), then \( (y, x) \in R \). We can show that \( R \) is transitive as follows: consider pairs \( (x, y) \in R \) and \( (y, z) \in R \). So \( y = 10^m x \) and \( z = 10^m y \), for \( m, n \in \mathbb{Z} \). Then, substituting, we have that \( z = 10^m(10^n x) = 10^{m+n} x \). Since \( m + n \in \mathbb{Z} \), we conclude that \( (x, z) \in R \).

The equivalence class for \( x \) is

\[ [x]_R = \{10^n x \text{ s.t. } n \in \mathbb{Z}\} \]

There is a distinct equivalence class for every real number in the intervals \((-10, -1) \cup \{0\} \cup [1, 10)\).

Problem 4. [10] Let \( G \) be a graph. Prove that there is a path from every vertex of odd degree to some other vertex of odd degree in \( G \).

Solution.

**Proof.** Note that there is a path between two vertices iff there is a walk between them, because the shortest walk between them must be a path. So we need only show that from every vertex of odd degree there is a walk to some other odd-degree vertex.

Now let \( v \) be a vertex of odd degree. Consider a longest walk starting at \( v \) in which each edge occurs at most once. Then every edge leaving the final vertex on the walk must already be on the walk. We will show that this final vertex of the walk is another vertex of odd degree.

Since \( v \) has odd degree, the walk contains at least one edge. Furthermore, the final vertex of the walk cannot be \( v \), since the edges incident to \( v \) would then consist of the first
and last edge of the walk plus two edges for each time that the walk crosses \( v \) — an even total. Therefore, the edges incident to the final vertex of the walk consist of the final edge of the walk plus two edges for each time that the walk crosses the final vertex — an odd total, as claimed.

\[ \square \]

**Proof.** An alternative proof begins with the observation from lecture that the sum of the degrees of the vertices in any connected component is twice the number of edges in the component. So there must be an even number of odd-degree vertices in any connected component. In particular, if there is a vertex of odd degree in the component, there must be an odd number of additional odd-degree vertices connected to it.

\[ \square \]

**Problem 5.** [30] We consider DAG’s where each vertex represents a task to be completed. If there is a path from one vertex, \( v \), to another vertex, \( w \), then the \( v \) task must be completed before the \( w \) task. Assuming all tasks take unit time to complete, we showed in the Notes that the minimum time schedule to complete all the tasks is the size (number of vertices), \( t \), of the longest path (chain) in the DAG.

Formally, a *schedule* for a DAG is a partition of the vertices. Each block of the partition is supposed to correspond to a set of tasks that are to be performed simultaneously. The number of processors required by a schedule is the maximum number of tasks that are scheduled to be performed simultaneously.

(a) Describe purely in terms of graph, partition, and partial order properties (no informal descriptions in terms of “jobs,” “parallel processing,” etc.):

- exactly the properties a vertex partition of a DAG must satisfy in order to represent a possible schedule for the vertex tasks,
- the total time required to complete a schedule,
- the number of processors required by a schedule.

**Solution.**

- A schedule for a DAG, \( G \), is a partition of the edges of \( G \) into a sequence of blocks, \( B_1, B_2, \ldots, B_k \) such that if \( a \in B_i \), \( b \in B_j \), and \( a < b \) (that is, there is a path of positive length from vertex \( a \) to vertex \( b \)), then \( i < j \). Another way to say this is that the blocks are anti-chains, and the sequence consisting of the elements in \( B_1 \) in any order, followed by the elements of \( B_2 \) in any order, through the elements of \( B_k \), is a topological sort of the partial order defined by \( G \).
- The total time required to complete a schedule is the number, \( k \), of blocks it has.
- The number of processors required by a schedule is the size of the largest block.
(b) Give a small example of a DAG with more than one minimum time schedule.

**Solution.** \( V = \{1, 2, 3\} , E = \{1 \rightarrow 2\} \). There are two minimum time schedules: \(\{\{1, 3\} \{2\}\}\) and \(\{\{1\} \{2, 3\}\}\).

(c) Explain why any schedule that requires only \(p\) processors to complete \(n\) tasks must take time at least \(\lceil n/p \rceil\).

**Solution.** If there are \(k < \lceil n/p \rceil\), then the integer \(k\) is less than \(n/p\). So if there are \(k\) blocks and each block has at most \(p\) vertices, the total number of vertices is \(\leq kp < (n/p) \cdot p = n\), a contradiction.

(d) Let \(D_{n,t}\) be the DAG with \(n\) vertices that consists of a directed path of \(t - 1\) vertices ending with edges from the final, \((t - 1)\)st, vertex on the path directly to each of the remaining \(n - (t - 1)\) vertices, as in the following figure:

![Diagram of DAG with \(D_{n,t}\)]

What is the minimum time schedule for \(D_{n,t}\)? Explain why it is unique. How many processors does it require?

**Solution.** There's no choice but to schedule each of the \(t - 1\) vertices on the path one at a time in order. A minimum time schedule then does all the remaining \(n - (t - 1)\) vertices at the \(t\)th time interval. The number of processors required is therefore \(n - t + 1\). The time is \(t\), the number of vertices on the longest chain in the graph.

(e) Describe a minimum time \(p\)-processor schedule for \(D_{n,t}\). Write a simple formula for this minimum time, \(M(n, t, p)\).

**Solution.** As in part (d), there's no choice but to schedule each of the \(t - 1\) vertices on the path one at a time in order. A minimum time schedule then does all the remaining \(n - (t - 1)\) vertices \(p\) at a time, for a total time of

\[
M(n, t, p) := (t - 1) + \left\lceil \frac{n - (t - 1)}{p} \right\rceil.
\]
(f) Show that every DAG with $n$ vertices and maximum chain size, $t$, has a $p$-processor schedule that runs in time $M(n, t, p)$.

**Hint:** Induction – you decide on what variable. You may find it helpful to use the fact that if $a \geq b \geq 0$, then

$$[a - b] \leq 1 + [a] - [b]$$

for all real numbers $a, b$.

**Solution.**

**Proof.** Induction on $t$. Induction hypothesis:

$$P(t) := \forall \text{DAGs } G, \forall n, p \in \mathbb{N}^+, \text{if } G \text{ has } n \text{ vertices and maximum chain size } t, \text{ then there is a } p \text{-processor schedule for } G \text{ that takes time } M(n, t, p).$$

**Base case** $t = 1$: In this case there are $n$ vertices and no edges between them. So any partition of the vertices into $\lceil n/p \rceil$ blocks of size at most $p$ will be a $p$-processor schedule taking time $\lceil n/p \rceil = 0 + \lceil (n - 0)/p \rceil = M(n, 1, p)$.

**Inductive step:** Assume $P(t)$ and conclude $P(t + 1)$ where $t \geq 1$.

Let $G$ be any DAG with $n$ vertices and maximum chain size $t + 1$. Suppose $k$ vertices are endpoints of maximum-size chains in $G$. Note that no edge can leave any of these endpoint vertices, for otherwise there would be a chain of length one more than the maximum chain size. Let $H$ be the subgraph of $G$ obtained by removing these $k$ vertices.

Now $H$ is a DAG with $n - k$ vertices and maximum chain size $t$, so by Induction Hypothesis, there is a $p$-processor schedule for $H$ taking time $M(n - k, t, p)$.

This $p$-processor schedule for $H$ can be extended to one for $G$ by adding $\lceil k/p \rceil$ disjoint blocks of the endpoints, all of size $\leq p$. So the time for this schedule for $G$ is

$$M(n - k, t, p) + \left\lceil \frac{k}{p} \right\rceil = (t - 1) + \left\lceil \frac{n - k - (t - 1)}{p} \right\rceil + \left\lceil \frac{k}{p} \right\rceil \quad (\text{def of } M)$$

$$= (t - 1) + \left\lceil \frac{n - t - k - 1}{p} \right\rceil + \left\lceil \frac{k}{p} \right\rceil \quad (3)$$

We complete the proof by showing that the expression (3) is $\leq M(n, t + 1, p)$. To do this, we consider two cases:

- **Case 1** ($k - 1$ is not a multiple of $p$): We have

$$\left\lceil \frac{k - 1}{p} \right\rceil = \left\lceil \frac{k}{p} \right\rceil \quad (4)$$
so

\[(3) \leq (t - 1) + \left(1 + \left\lfloor \frac{n - t}{p} \right\rfloor - \left\lfloor \frac{k - 1}{p} \right\rfloor + \left\lfloor \frac{k}{p} \right\rfloor \right) + \left\lfloor \frac{k}{p} \right\rfloor \quad \text{(by (2))}\]

\[= (t - 1) + \left(1 + \left\lfloor \frac{n - t}{p} \right\rfloor - \left\lfloor \frac{k}{p} \right\rfloor + \left\lfloor \frac{k}{p} \right\rfloor \right) \quad \text{(by (4))}\]

\[= t + \left\lfloor \frac{n - t}{p} \right\rfloor \]

\[= M(n, t, p). \quad \text{(def of } M\text{)}\]

- **Case 2** \((k - 1)\) is a multiple of \(p\): Now we have

\[\left\lfloor \frac{k}{p} \right\rfloor = 1 + \frac{k - 1}{p}, \quad \text{(5)}\]

so

\[(3) = (t - 1) + \left(\left\lfloor \frac{n - t}{p} \right\rfloor - \frac{k - 1}{p} \right) + \left\lfloor \frac{k}{p} \right\rfloor \quad \text{(since } (k - 1)/p \in \mathbb{Z}\text{)}\]

\[= (t - 1) + \left\lfloor \frac{n - t}{p} \right\rfloor - \frac{k - 1}{p} + \left(1 + \frac{k - 1}{p} \right) \quad \text{(by (5))}\]

\[= t + \left\lfloor \frac{n - t}{p} \right\rfloor \]

\[= M(n, t + 1, p). \quad \text{(def of } M\text{)}\]