Perimeter Coverage Using Backward and Forward Greedy Algorithm

S. Renugadevi
SENSE, VIT University
Vellore, Tamil Nadu, India-632014

Aayush S. Rawat
SENSE, VIT University
Vellore, Tamil Nadu, India-632014

Suraj Swaminathan
SENSE, VIT University
Vellore, Tamil Nadu, India-632014

P. Arulmozhivarman
SENSE, VIT University
Vellore, Tamil Nadu, India-632014

R. Selvakumar
SAS, VIT University
Vellore, Tamil Nadu, India-632014

Copyright © 2013 S. Renugadevi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

In today’s world there is much need for monitoring of object(s). Depending upon the area of coverage, many sensors might be used to cover the entire area to be monitored. In such extensive areas, there is a need for finding minimum number
of sensor nodes to activate among all nodes in wireless sensor networks. Greedy forward algorithms are currently efficient and effective algorithms that help find the minimum set of sensors required to monitor an area. Battery conservation has been a long term problem that many try to address. Our algorithm is basically an improvement over the present greedy forward algorithm in terms of reduced time complexity by parallel computation and also tries to extend the network lifetime of the network.

**Keywords:** Cover Range, Minimum Cover, Greedy Forward and Greedy Backward Neighbor, Greedy Neighbor

### 1 Introduction

Wireless sensor networks find varied usage from military applications—landmine monitoring to health monitoring and as early warning system in case of geo-hazards. In our paper, we explore a scenario in which a wireless sensor network is used for landslide monitoring on a target area. A number of sensors are distributed spatially on the area to be monitored as early detection warning system lest the event of a landslide were to occur. Thus there is a need to have updated and efficient algorithms which are able to provide early detection of such calamities at the earliest. The underlying principle of placement and cover ranges of the sensors is taken from [5] which explains the how cover ranges of sensors are obtained even though on terms of Visual Sensors but the underlying principle can be used for the purpose of finding Neighbors.

Practically, each sensor has a limited power bank. Thus it is in our interests that the power be conserved so that the network can work or sense for a longer time. Our algorithm thus aims at minimizing the number of sensors activated. Also, it is in our interests to find multiple minimum covers (MCs) which can act as backups [6] in case a set of nodes were to fail due to some reason.

Our algorithm is an improvement over the distributed greedy forward algorithm as seen in [6]. We use both the greedy forward and greedy backward neighbor techniques simultaneously and by doing so reduce the number of hops needed to find minimum cover drastically. We were thus successful in reducing the time convergence of the algorithm. We provide a proof of the same by comparing it to the algorithm in [6] in terms of convergence time.

The paper is organized in the following fashion- In section 2 we explain the reasons behind our research in this field and we will also explain the algorithm in [6] so as to provide a background for our work. Later on, we explain in detail our proposed algorithm. We conclude by providing an example to help understand our algorithm and proof as to how our algorithm is an improvement over previous algorithm in [6].
2 Related Work

Our main motivation came from [3], [4], and [8] which discuss the challenges today’s sensor networks face in terms of energy consumption and network longevity, the requirements for deploying sensor networks on a large scale and also help us get an overview of the various research areas that this field offers. We wanted to devise algorithms that address problems related to energy consumption and network longevity. The paper [10] helped us understand how power gets consumed during sensing, processing and transmission. The authors of [8] and [9] helped us get an insight as to why there is a need for energy consumption techniques and also talk about energy harvesting for environment monitoring applications. The nodes consume a lot of energy in communicating wirelessly. As rightly pointed out by [2], the energy utilized in transmission exceeds far more than that used in processing the data. Hence, it is in our best interests to devise algorithms that use minimum sensor nodes to monitor the entire target and thereby save energy as well as improve the network lifetime.

The authors in [1] talk about the need to position or the nodes appropriate places and distances for proper functioning of the network so that there is connectivity among the nodes and that the network is connected in spite of failure of a few nodes. We, too, talk about the placement of sensors and have considered random distribution. It would be beneficial to consider some of the ideas [1] illustrates. The sensors should be close enough so that there is proper wireless communication between them. Thus, the above papers have contributed in some or the other way motivating us to come up with new and improved algorithms so as to address the problems the sensor networks face.

Further we talk about the Greedy Forward Algorithm that is explained in [6] so as to help you understand the basic principle of our algorithm.

For the convenience of the algorithm, we assume a set of sensors \( n \) that cover the whole target area that is required to be monitored. Each sensor is assumed to have a cover range which has a starting and an ending angle. The direction of the cover range is taken to be clockwise. So, the sending angle would be to the right of the starting angle.

Now, let \( S_0 \) be a set of sensors whose cover ranges pass through \( 0^\circ \). Out of these sensors, we chose one such sensor (termed as the start sensor) which has the maximum ending angle, to be the sensor that collects and forwards the information of the remaining sensors’ greedy forward neighbors (GFNs). The information it receives from the sensors is in the form \( \langle sq,q,GFN(q) \rangle \) where \( q \) is the sensor belonging to the set \( S_0 \) including the start sensor, \( sq \) is the starting angle of the sensor which is required for the purpose of pruning in case of redundant Greedy Forward Neighbors and \( GFN(q) \) is the Greedy forward neighbor of the sensor \( (q) \). Once the start sensor receives the information, it forwards the information to its GFN.

The GFN now prunes the information and requests the remaining GFNs to find their respective GFNs (i.e. \( GFN^2(q) \)) and inform the start sensor’s GFN. One can
refer to [6] in order to completely understand the process of pruning and combining as well as finding GFNs. Once the GFN(Start Sensor) receives the information from its neighbors (i.e. GFNs of S0) it forwards this to its GFN and this process continues in the same manner. We define a Hop to be the step where a node prunes the redundant nodes and informs the remaining nodes and receives the information regarding the GFN from these nodes. At each Hop, k, the node checks to see whether there is overlapping of the GFN\(^k\)(q) with q. When this occurs the algorithm terminates and the minimum cover is found. Our algorithm runs on a similar notion but instead of moving only in the clockwise direction throughout the algorithm, we plan to move in both directions at the same time using the same principles explained in [6], but with the added complexity of computing simultaneously at different nodes.

3 Backward and Forward Algorithm

Our algorithm is based on the principles of the previously discussed algorithm.

a) Finding S0, Start Sensors and Greedy Neighbors

Instead of moving or finding the minimum cover in one direction, we can move in both directions and thereby find the minimum cover quicker than the above algorithm. Similar to [6], let S0 be the set of sensors having their cover ranges passing through 0\(^\circ\) and the clockwise direction as the forward direction. Out of these, select two such sensors, one having maximum ending angle and the other having minimum starting angle among them. Let them be termed as Start Sensor-Forward and Start Sensor-Backward respectively. Once the start sensors are selected, each sensor in S0 finds its GFNs and GBNs and forwards them to the respective start sensors.

Note that we will use the terms Greedy neighbors for nodes that receive the bulk of information or in other words are Greedy forward or backward neighbors of the start sensors.

For example, if S0={1,2,3,4,5} and 1 and 5 are the Start Sensor-Backward and Start Sensor-Forward, then the sensors in S0 will forward their GFNs and GBNs to 5 and 1 respectively. The start sensors will in turn forward this information to their respective greedy neighbors. So, the start sensor greedy backward neighbor will receive the information of all GBNs of the sensors in S0 and similarly, start sensor greedy forward neighbor will receive that of all GFNs of the sensors in S0. Thus, GBN(1) will receive the information in the form \(<s1,t1,1,gbn(1)>, <s2,t2,2,gbn(2)>, …\) while GFN(5) will receive the information in the form \(<s1,t1,1,gfn(1)>, <s2,t2,2,gfn(2)>, …\).
Pruning, Combining and Overlapping

As mentioned in [6], the usual Pruning and Combining takes place where nodes with the same greedy forward or backward neighbors would be pruned based on the sensors starting and ending angles. Once this is done, the Greedy neighbors will request the respective GFNs and GBNs from the received information to find their GFN and GBN i.e. \( \text{GFN}(2) \) and \( \text{GBN}(2) \); such that the ones requested by the Start Sensor GBN will find their GBNs while the ones requested by the Start Sensor GFN will find their GFNs. This continues for as many hops and the search will stop based on the terminating condition.

Note that Hop, here signifies the completion of forwarding the information from one node to its greedy neighbor and the completion of pruning and combining of the information. So, at the first hop, the nodes forward the information to the start sensors’ greedy neighbors and the nodes with redundant neighbors are pruned. At each Hop, the start sensors’ greedy neighbors check to see whether they overlap with each other.

From the above example, at the first hop, GBN(1) and GFN(5) will check whether they overlap with each other. In other words, GFN(5) will check to see whether GBN(1) is its forward neighbor. If this is not the case, the nodes forward the information to their respective GFN and GBN and the process continues. But if after some hop the nodes overlap, they share the information. Sharing of information takes place from one of the greedy neighbors to the other. Let us assume that the Greedy Backward neighbor relays the information to the corresponding Greedy Forward neighbor. This happens when the GFN informs all the Forward neighbors of its appointment as the Start sensor GFN\(^k\) node, where \( k \) is the hop where the node is found to be the GFN. On receiving the information, the GFN will compare the shared information with the information it received from its predecessor.

Common and Uncommon Nodes

Now, from the information, there arises a case where certain nodes are common to both the Greedy neighbors. In such a situation, we find our Minimum cover. The minimum cover is such that, since the greedy neighbors overlap, they will check to see whether the GFNs or the GBNs in the common nodes overlap with each other. If they don’t, using the properties explained in [6], either the GFN of the start sensor or the GBN of the other start sensor would overlap with those of the common nodes. Thus, the minimum cover would be found.

But, if there are no nodes common among the shared and the received information, the greedy neighbor (in case of our example, gfn\(^k(5)\)) will check for the overlap of the GFN and GBN of the uncommon nodes and if they overlap, which they are bound to, it checks the starting angles of the uncommon nodes.
The minimum cover is found when the starting angle of the sensor present in the received information (Information received from the Start Sensor’s Predecessor) is less than that present in the shared information and the respective nodes are informed of their presence in the Minimum Cover. Since there would be uncommon many nodes, we would have to check for each of the nodes in the forward neighbor.

Another situation that is possible is that the GFNs and GBNs of the common nodes would be the same. In such a situation the minimum cover would be less than the usual minimum cover obtained at the same hop. For example, usually in the 2nd hop we get a total of 5 sensors including that in S0. But if the GBN and GFN are the same, we would get one less sensor in the Minimum Cover due to the common neighbor. We would have to keep this in mind while programming the processors to run the algorithm. The above algorithm would be clear once the example in the next section is explained.

d) Costs to find the Active Minimum Cover

Once the multiple covers are found, we can easily find which minimum cover to be activated so that the others can for the time being remain idle. This can be done by assigning costs to each sensor based on the range that it covers. The cost would be assigned to only the sensor nodes in S0. Calculation of the cost can be done by either of the start sensors which would be relayed back to them. The cost assigned would be in a manner where the lowest cost assigned to the one with the maximum range and the highest to the one with the least range. One of the reasons the sensors with the maximum range should be active is, with larger ranges the size of the minimum covers decrease [6] and as a result we would require lesser sensors to cover the entire object. Obviously, with multiple minimum covers, the size would remain the same. But, the node with greater range would be able to sense a greater area than the one with a smaller range.

So, when the minimum covers are found and the respective sensors are let known of their presence in the minimum cover which can be done by the procedure in [5] and [6], the one with the cost ‘1’ would start monitoring the target area while the others with greater costs would go into sleep mode waiting for failure of any of the active nodes. Another possibility of assigning the costs would be based on the battery levels of the nodes in the minimum covers. But, we would stick to the range for now as it seems to be a better option than the latter. The following example explains the algorithm minus the cost factor but the readers are free to input the cost factor and find the minimum cover.

One must note that while this algorithm explains on a similar ground as the one explained in [6], we have not considered the possibility of a default member. A default member is the node removal of which will result in the failure of the entire network. In other words, one can consider a node to have only one Forward and Backward neighbor or a node which has No neighbors. Such nodes have to be
considered, as they are the only nodes covering that particular area of the target. We must make sure that every node has more than one neighbor and that there are no default members in the scenarios considered for the use of our algorithm. The main reason behind this condition is that the failure of the default member will result in failure of the entire network. There would be no point of convergence of this algorithm and thus does not solve the purpose of proposing the algorithm in the first place. Even in the example considered below, it would help to consider that no node is a default member as the below example is used to explain the algorithm. So node 8 would seem like a default member but it would be convenient to assume that it is supplied with constant power supply and that there is no possibility of its failure.

4 Example

The following section will explain the above algorithm using an example. We consider a target area that requires a set of 13 sensors to entirely cover it. In order to understand how a particular sensor identifies its cover range, one can read [5]. From the figure, we can easily find out the set $S_0$

$$S_0=\{1, 2, 3\}$$

Obviously, there cannot be a minimum cover without any of these sensors. Based on the principles explained, 1 is the start sensor (backward) and 3 is the start sensor (forward). Now all the starts sensors find their greedy Backward and Forward neighbors.

<table>
<thead>
<tr>
<th>Backward-GBN</th>
<th>Forward-GFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start sensor-1</td>
<td>Start sensor-3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Now in forward direction, sensor 1 has 3 as GFN but both of them exist in start sensor $S_0$ (same set) this causes sensor 3 to think that the loop is complete and unless this is pruned we won’t be able to proceed to find the minimum MC.

In backward direction, sensors 2 and 3 both have 13 as their GBN. Hence 2 gets pruned (3 has greater ending angle than 2).
After pruning:

<table>
<thead>
<tr>
<th>Backward- GBN</th>
<th>Forward- GFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start sensor -1</td>
<td>Start sensor-3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
</tbody>
</table>

Now it checks whether 12 and 6 (GBN of 1 and GFN of 3) overlap. Since they don’t, the unpruned GBNs and GFNs find their greedy neighbors as shown and the respective information is forwarded to 9 and 8 respectively.

Here no scope of any further pruning is possible hence 8 and 9 check whether they are overlapping with each other. Since they do, they will share information with each other. Now they check whether any common node exists between the two. In this case sensor 3 is common to both. It thus finds the minimum cover in two hops the MC being- {3,6,8,10,13}.

![Spatial arrangement of 13 sensors](image)

Figure 1: Spatial arrangement of 13 sensors

<table>
<thead>
<tr>
<th>Backward-Start sensor -1</th>
<th>GBN</th>
<th>GBN²</th>
<th>Forward-Start sensor-3</th>
<th>GFN</th>
<th>GFN²</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>10</td>
<td>3</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>
Perimeter coverage

Here no scope of any further pruning is possible hence 8 and 9 check whether they are overlapping with each other. Since they do, they will share information with each other. Now they check whether any common node exists between the two. In this case sensor 3 is common to both. It thus finds the minimum cover in two hops the MC being- \{3,6,8,10,13\}.

Now it goes on to find remaining minimum MC’s using the uncommon nodes. First GFN\(^2\)(2) i.e. 7 will see if it overlaps with GBN\(^2\)(1) i.e. 9. Since they don’t now GFN\(^2\)(2) i.e. 7 will check if it overlaps with GBN\(^2\)(3) i.e. 10 which also is not the case. Hence now GFN\(^2\)(3) i.e. 8 will check whether it overlaps with GBN\(^2\)(1) i.e. 9 which is the case, as evident from figure 2. Now it checks whether 3 has a starting angle lesser than 1. This is to eliminate 1 from the MC so found, if this be the case (as it will then completely overlap 1 and also overlap with GBN(1)). However due to the above condition not holding the only MC possible in the above topology is \{3,6,8,10,13\}

Note that if we were to find the same using algorithm in [6], it would have taken us 4 hops instead of 2.

If in the above example, 8 had a larger cover range then 7 and if it were to overlap with 2 then GFN\(^2\)(2) = GFN\(^2\)(3) = 8 as a result of which 3 would get pruned and we would arrive at a situation where there are no common nodes in the shared and the received information. In this case we should check for overlapping of the GFN’s and GBNs of the uncommon nodes. Here GFN\(^2\)(2) overlaps with GBN\(^2\)(3) and GBN\(^2\)(1), next we would check the starting angles of the nodes.

S2 < S3 and S2 > S1 (S Ë starting angle)

From the above equation we may infer that the nodes overlapping with 3 at the backward end will definitely have to overlap with 2 which isn’t the case with 1. Hence our minimum cover is found.

5 Simulation and Results

In order to prove that our algorithm fared better in terms of faster Time of Convergence, we assumed a few sensor values ranging from 15 sensors to 40 sensors (the values of which are shown at the end of the paper) that cover a target perimeter and tested both our algorithm as well as the algorithm in [6]. It can be seen from the graph below that in each of the 4 examples our algorithm gave the minimum cover in lesser hops and in most of the cases it would amount to half of the number of hops that the Greedy Forward Algorithm would provide. Moreover, we can easily obtain multiple minimum covers since both common and uncommon nodes are compared and based on the starting angles of the uncommon
nodes the minimum cover can be found. Obviously, our algorithm only explains how minimum covers can be found but the actual procedure of informing the nodes of their presence in the minimum cover is done in the same manner as explained in [6].

<table>
<thead>
<tr>
<th>Number of Sensors</th>
<th>Total Minimum Covers</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1: Describes the Total Minimum Covers found for different set of Sensors.

6. Future Work and Conclusion

Thus, in this paper, we have discussed the existing Greedy Forward Algorithm and explained our Backward and Forward Algorithm. Our proposed algorithm is better in terms of Time Convergence and reduced number of hops but at the expense of Compiler complexity and message complexity. Also like in [6], our paper finds multiple MCs and they are found with much ease. This is done in order to have backup(s) in case of failure of one or more nodes as we can’t have a non functioning network due to the nature of the application like landslide monitoring.

For applications like these which requires constant sensing of all areas we recommend against the presence of any default members in our sensor network. We have also used an example for the ease of understanding of the algorithm.
The next step towards the use of this algorithm would be to implement it in a network simulator in order to observe the flow and count of messages so as to determine whether the proposed algorithm is as efficient and effective in real-time scenarios. The algorithm can be deployed in various monitoring applications like Early Detection of Landslides or other Geo-Hazard detection. One can use [9] as an added feature when nodes fail due to battery related problems.

Also, in order to realize this algorithm in real-time scenarios, one must introduce Cluster Heads which will act as databases and that the information of the Minimum covers would be present with them. So, when one of the nodes in Minimum Cover fails, the Cluster Heads can activate the other set of Minimum Cover nodes so that there is continuity in terms of Monitoring and that the nodes can again initiate the algorithm, only now, the failed node would not participate in the process and different set of Minimum covers would be obtained.

In applications like Landslide Monitoring where it is required that continuous monitoring of the Target land takes place, the Cluster Heads not only get the information pertaining to the Minimum Covers but also get the sensing information or data from the active sensors.

We have made an effort to contribute towards developing effective and efficient algorithms that reduce the overall energy consumption of all nodes, converge upon the minimum set of sensor, minimize the costs required to set up wireless networks and optimize the energy consumption and utilization of the current networks. It is the need of the hour to devise novel algorithms that address the aforementioned problems.
Appendix

The following are the sensors with their Start and End angle and the sensor number respectively.

20 sensors
s1(1)=310; s1(2)=20; s1(3)=300; s1(4)=350; s1(5)=55; s1(6)=75; s1(7)=100;
s1(8)=135; s1(9)=190; s1(10)=170; s1(11)=210; s1(12)=205; s1(13)=230; s1(14)=235;
s1(15)=260; s1(16)=250; s1(17)=285; s1(18)=295; s1(19)=310; s1(20)=345;
t1(1)=70; t1(2)=110; t1(3)=50; t1(4)=60; t1(5)=165; t1(6)=180; t1(7)=205; t1(8)=240;
t1(9)=275; t1(10)=260; t1(11)=305; t1(12)=290; t1(13)=305; t1(14)=310; t1(15)=340;
t1(16)=330; t1(17)=355; t1(18)=25; t1(19)=50; t1(20)=75;
sn(1)=1; sn(2)=2; sn(3)=3; sn(4)=4; sn(5)=5; sn(6)=6; sn(7)=7; sn(8)=8; sn(9)=9;
sn(10)=10; sn(11)=11; sn(12)=12; sn(13)=13; sn(14)=14; sn(15)=15; sn(16)=16;
sn(17)=17; sn(18)=18; sn(19)=19; sn(20)=20;

30 sensors
s1(1)=350; s1(2)=10; s1(3)=5; s1(4)=330; s1(5)=55; s1(6)=325; s1(7)=355; s1(8)=335;
s1(9)=60; s1(10)=75; s1(11)=80; s1(12)=105; s1(13)=120; s1(14)=145; s1(15)=150;
s1(16)=155; s1(17)=185; s1(18)=205; s1(19)=210; s1(20)=225; s1(21)=240;
s1(22)=245; s1(23)=250; s1(24)=260; s1(25)=275; s1(26)=280; s1(27)=285;
s1(28)=295; s1(29)=315; s1(30)=330;
t1(1)=90; t1(2)=115; t1(3)=95; t1(4)=75; t1(5)=165; t1(6)=60; t1(7)=85; t1(8)=45;
t1(9)=135; t1(10)=180; t1(11)=165; t1(12)=190; t1(13)=205; t1(14)=230; t1(15)=235;
t1(16)=230; t1(17)=255; t1(18)=285; t1(19)=295; t1(20)=305; t1(21)=310;
t1(22)=325; t1(23)=330; t1(24)=335; t1(25)=340; t1(26)=355; t1(27)=350; t1(28)=5;
t1(29)=30; t1(30)=65;
sn(1)=1; sn(2)=2; sn(3)=3; sn(4)=4; sn(5)=5; sn(6)=6; sn(7)=7; sn(8)=8; sn(9)=9;
sn(10)=10; sn(11)=11; sn(12)=12; sn(13)=13; sn(14)=14; sn(15)=15; sn(16)=16;
sn(17)=17; sn(18)=18; sn(19)=19; sn(20)=20; sn(21)=21; sn(22)=22; sn(23)=23;
sn(24)=24; sn(25)=25; sn(26)=26; sn(27)=27; sn(28)=28; sn(29)=29; sn(30)=30;

40 sensors
s1(1)=335; s1(2)=340; s1(3)=15; s1(4)=10; s1(5)=5; s1(6)=25; s1(7)=55; s1(8)=70;
s1(9)=60; s1(10)=85; s1(11)=100; s1(12)=105; s1(13)=115; s1(14)=130; s1(15)=135;
s1(16)=145; s1(17)=185; s1(18)=205; s1(19)=210; s1(20)=225; s1(21)=240;
s1(22)=245; s1(23)=250; s1(24)=260; s1(25)=275; s1(26)=280; s1(27)=285;
s1(28)=295; s1(29)=305; s1(30)=310; s1(31)=315; s1(32)=320; s1(33)=325;
s1(34)=335; s1(35)=345; s1(36)=350; s1(37)=355; s1(38)=20; s1(39)=35; s1(40)=45;
Perimeter coverage

\begin{align*}
t_1(1) &= 50; t_1(2) = 65; t_1(3) = 70; t_1(4) = 85; t_1(5) = 90; t_1(6) = 115; t_1(7) = 115; t_1(8) = 145; \\
t_1(9) &= 125; t_1(10) = 160; t_1(11) = 175; t_1(12) = 170; t_1(13) = 185; t_1(14) = 190; t_1(15) = 200; \\
t_1(16) &= 220; t_1(17) = 255; t_1(18) = 285; t_1(19) = 295; t_1(20) = 305; t_1(21) = 310; \\
t_1(22) &= 325; t_1(23) = 330; t_1(24) = 335; t_1(25) = 340; t_1(26) = 355; t_1(27) = 350; t_1(28) = 5; \\
t_1(29) &= 30; t_1(30) = 45; t_1(31) = 55; t_1(32) = 50; t_1(33) = 65; t_1(34) = 70; t_1(35) = 75; \\
t_1(36) &= 80; t_1(37) = 90; t_1(38) = 85; t_1(39) = 95; t_1(40) = 105; \\
\end{align*}

\begin{align*}
s_n(1) &= 1; s_n(2) = 2; s_n(3) = 3; s_n(4) = 4; s_n(5) = 5; s_n(6) = 6; s_n(7) = 7; s_n(8) = 8; s_n(9) = 9; \\
s_n(10) &= 10; s_n(11) = 11; s_n(12) = 12; s_n(13) = 13; s_n(14) = 14; s_n(15) = 15; s_n(16) = 16; \\
s_n(17) &= 17; s_n(18) = 18; s_n(19) = 19; s_n(20) = 20; s_n(21) = 21; s_n(22) = 22; s_n(23) = 23; \\
s_n(24) &= 24; s_n(25) = 25; s_n(26) = 26; s_n(27) = 27; s_n(28) = 28; s_n(29) = 29; s_n(30) = 30; \\
s_n(31) &= 31; s_n(32) = 32; s_n(33) = 33; s_n(34) = 34; s_n(35) = 35; s_n(36) = 36; s_n(37) = 37; \\
s_n(38) &= 38; s_n(39) = 39; s_n(40) = 40
\end{align*}

References


Received: April 3, 2013