

B.Tech. Project Report

**Enumerating Independent Sets In Trees And
Chordal Graphs**

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Technology

by
Rahul T. Shah
93005004

under the guidance of
Dr. A. A. Diwan

Department of Computer Science and Engineering
Indian Institute of Technology
Bombay
April 7, 1997

Abstract

This project deals with enumerating independent sets in trees and some particular kinds of chordal graphs. In enumeration, when we proceed from one set to another we try to get the newer set by minimum change in the previous one. So, the enumeration follows 'gray code' criteria. The problem may be seen equivalent to finding a Hamiltonian path in a graph. Each independent set can be seen as a vertex of this graph and edge in the graph indicates that two independent sets corresponding to it can be obtained from each other by minimum change. We show the existence of 'gray code' for various subproblems. The proofs use induction and are algorithmic.

Contents

1	Introduction	1
1.1	Problems Considered	1
2	Enumerating Maximum Independent Sets in Trees	2
2.1	Finding the Maximum Size	2
2.2	Labeling the Vertices	3
2.3	Example	7
2.4	Explanation of Merge	7
2.5	Enumerating Maximum Independent Sets in Chordal Graph	8
3	Enumerating all k-sized independent sets in path	10
3.1	Example	12
3.2	Successor Algorithm	13
3.3	Similarity between Path and Intervals	13
3.4	Extension	14
4	Enumerating k-sized Independent Sets in Interval Graphs	15
4.1	Representation of Set of Intervals	15
4.2	Example	17
5	Enumerating k-sized Independent Sets in Trees	18
5.1	Enumerating k-sized Independent Sets in Well Covered Trees	18
5.2	Enumerating k-sized shadows of maximum independent sets in trees .	26
5.3	Example	27
6	Conclusions	28

List of Figures

2.1	- vertex	6
2.2	+/- vertex	6
2.3	+ vertex	6
2.4	Example	7
2.5	Zigzag traversal for merged list	8
2.6	A counter example	8
3.1	An example of path with the first and the last independent set	11
3.2	Set of intervals corresponding to path	14
3.3	Extension of path	14
4.1	Representation of set of intervals	16
5.1	Example of a well covered tree	19
5.2	Lemma 5.1 and Lemma 5.2	21
5.3	Matrix formed by merge	22
5.4	Subcases 1.1 and 1.2 of Case 1 for Lemma 5.1	23
5.5	Case 2 for lemma 5.1	25
5.6	Sizes of various sets in the branches	26

Chapter 1

Introduction

In this B.Tech project we attempt to solve some enumeration problems. We have attempted to solve the problems concerning enumeration of independent sets in trees and some particular cases of chordal graphs. While listing we start with some independent set and we try to get the next independent set which differs the least (in some sense) from the previous one. We try to enumerate all the independent sets, possibly having some properties, without repetitions. Each independent set can be represented by a vertex in a graph. And there exists an edge between two vertices if independent sets corresponding to them differ minimally. So problem is equivalent to finding a Hamiltonian Path in this graph.

1.1 Problems Considered

We have solved the following enumeration problems.

1. Maximum Independent sets in trees (chapter 2).
2. k-sized independent sets in a path (chapter 3).
3. k-sized independent sets in proper interval graphs(chapter 4).
4. k-sized independent sets in a subclass of trees(chapter 5).
5. k-sized shadows of maximum independent sets in trees(chapter 5).

Chapter 2

Enumerating Maximum Independent Sets in Trees

Here the primary purpose is to design an algorithm to enumerate all maximum sized independent sets in a given tree in a 'Gray code' order. The listing criterion is that every Independent Set in the list should be obtained from the previous one by replacing one vertex by another. So here we want a "Gray code" listing of all maximum independent sets in such a way that the smallest possible change takes place as we go from one independent set to its successor.

2.1 Finding the Maximum Size

For this, we can look at the tree as a rooted tree. Then we apply dynamic programming algorithm as follows :

We denote, for each vertex i ,

S_{inc}^i = size of maximum independent set in a subtree rooted from i , which include the vertex i .

S_{ni}^i = size of maximum independent set in a subtree rooted from i , which does not include the vertex i

S^i = size of maximum independent set in a subtree rooted from i .

We note that $S^i = \max(S_{inc}^i, S_{ni}^i)$.

So for any leaf node i

$$S_{inc}^i = 1$$

and $S_{ni}^i = 0$

And we set up the dynamic programming equations as follows :

$$S_{inc}^i = 1 + \sum_{\text{all children } j \text{ of } i} S_{ni}^j$$
$$S_{ni}^i = \sum_{\text{all children } j \text{ of } i} S^j$$

So if r is the root of the tree then size of maximum independent set in the tree is S^r .

While executing this algorithm we maintain the numbers S_{inc}^i and S_{ni}^i for each vertex i .

2.2 Labeling the Vertices

We note that there can be 3 possible relations between S_{inc}^i and S_{ni}^i for any vertex i . According to this, we label then vertex as either "+", "-", or "+/-". The labeling is as follows:

1. if $S_{ni}^i > S_{inc}^i$ then vertex i is called "-" vertex.
2. if $S_{ni}^i = S_{inc}^i$ then vertex i is called "+/-" vertex.
3. if $S_{ni}^i = S_{inc}^i - 1$ then vertex i is called "+" vertex.

Now we observe a few parent-children relationships with respect to the above labeling.

1. If a parent is "-" vertex, then at least two of its children are "+" vertices. Because here,

$$S_{ni}^i > S_{inc}^i$$

So for all children j of i

$$\sum_j S_{ni}^j + 1 < \sum_j S^j$$

This means for at least two children we have S^j greater than S_{ni}^j . Note that S^j can be at most one more than S_{ni}^j and in this case j is "+" vertex.

2. If a parent is "+/-" vertex then exactly one child of it is a "+" vertex. Because here,

$$S_{ni}^i = S_{inc}^i$$

So for all children j of i

$$\sum_j S_{ni}^j + 1 = \sum_j S^j$$

This means the factor of 1 comes from exactly one children j for which $S_{inc}^j = S_{ni}^j + 1$ which means it is "+" vertex. All others can be either "-" or "+/-".

3. If a parent is "+" node then all its children are either "-" vertices or "+/-" vertices. Because here,

$$S_{ni}^i = S_{inc}^i - 1$$

So for all children j of i

$$\sum_j S_{ni}^j = \sum_j S^j$$

. This means all the children are either "-" vertices or "+/-" vertices.

We notice that "-" vertex can not be included in any maximum independent set.

Theorem 2.1 *All maximum independent sets in a tree can be enumerated using the operation exchange of vertices. Moreover enumeration is such that all maximum independent sets containing root occur before those not containing root.*

Notation : Let

L+ denote the list of all maximum independent sets containing the root.

L- denote the list of all maximum independent sets not containing the root.

L=L+L- is the entire list.

Proof :

By induction on size of the tree.

Base Case : For $|T| = 1$, the only independent set is $\{1\}$. Here we assume that the only vertex has number 1.

Induction Step: Let's assume that the theorem is valid for all trees having size less than n . Now consider the tree with n vertices. Now we have three cases:

Case 1 : If root r of tree is a "-" vertex then it can not be included in any of the maximum independent sets. So only the independent sets not containing the root are

possible. Therefore, L_+ is null. And $L = L_-$ is the list of independent sets obtained by merging the list of independent sets of children of r .

Case 2 : If root r of tree is a "+" vertex then $L = L_+$ and is obtained by merging L_- of its children and adding r to all independent sets in the list. Here we see that some of its children may be "+/-" vertices, in which case the list for them is of the form L_+L_- . We denote

1. I_{++} as the independent set in L_+ at the endpoint away from L_- .
2. I_{+-} as the independent set in L_+ at the endpoint near L_- .
3. I_{-+} as the independent set in L_- at the endpoint near L_+ .
4. I_{--} as the independent set in L_- at the endpoint away from L_+ .

In case of "-" vertices I_{++} and I_{+-} do not exist, where I_{-+} and I_{--} can be chosen arbitrarily from the endpoints of L_- .

So here in enumeration of L_+ for the root, we merge the L_- of the children and get the union of I_{--} 's of children and r as one of the endpoints of L_+ . Let's call this endpoint I'_{++}

Also in this case we can have one more list that is list of maximum possible independent sets if the root is not included. The size of independent sets in this list will be one less than the size of maximum independent set. This can be obtained by merging the lists L of all its children. Note that one endpoint of this merged list is again union of I_{--} of children. Let's call this list as $L-'$. The endpoint of this list can be denoted by I'_{--} . Note that $I'_{++} = r \cup I'_{--}$

So in this case we have two lists: $L=L_+$ which has I'_{++} as its endpoint, and $L-'$ which has I'_{--} as its endpoint.

Case 3 : If r is "+/-" vertex then we have $L=L_+L_-$ with L_+ and L_- being nonempty. Here we have exactly one child of type "+". So L_+ consists of merge of L_- of children and $L-'$ of "+" child and r added to each element of it. Now I_{+-} endpoint for r consists of union of I_{--} of its children and I'_{--} of its "+" child and r . L_- consists of merge of L of all its children with its I_{-+} endpoint being union of I'_{++} and I_{--} of its other children. We see that transition from I_{+-} to I_{-+} takes place by excluding root and including the "+" child.

That completes all the cases and hence the proof.

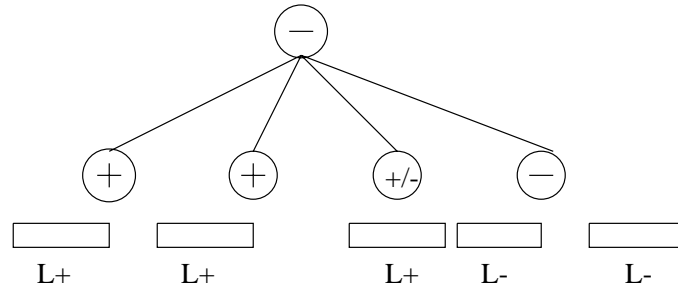


Figure 2.1: $-$ vertex

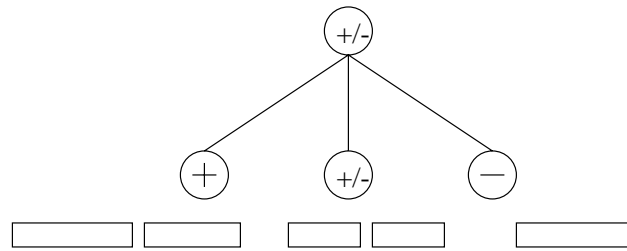


Figure 2.2: $+/-$ vertex

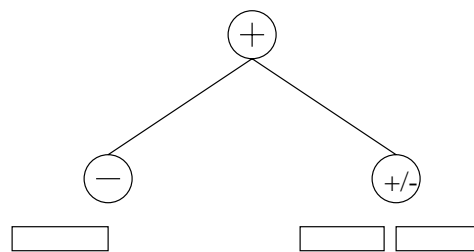


Figure 2.3: $+$ vertex

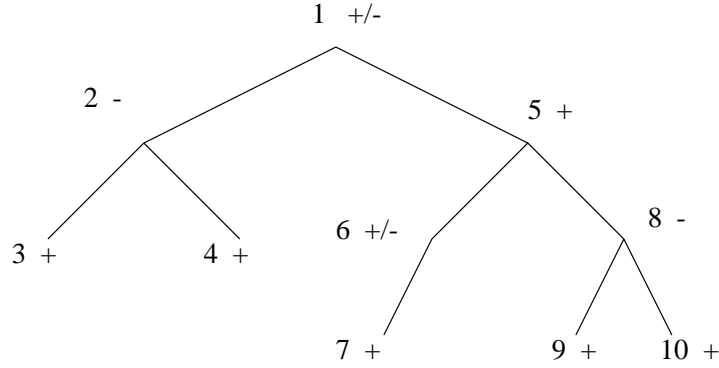


Figure 2.4: Example

2.3 Example

The following shows the lists of independent sets constructed at each vertex of the tree as shown in 2.4.

- Vertex(3) : $L+ = \{3\}$
- Vertex(4) : $L+ = \{4\}$
- Vertex(2) : $L = L- = \{3, 4\}$
- Vertex(7) : $L+ = \{7\}$ $L- = \{\}$
- Vertex(6) : $L+ = \{6\}$ $L- = \{7\}$
- Vertex(8) : $L = L- = \{9, 10\}$
- Vertex(5) : $L+ = \{5, 7, 9, 10\}$ $L- = \{6, 9, 10\}, \{7, 9, 10\}$
- Vertex(1) : $L+ = \{1, 3, 4, 6, 9, 10\}, \{1, 3, 4, 7, 9, 10\}$ $L- = \{3, 4, 5, 7, 9, 10\}$

So the listing is :

$$\{1, 3, 4, 6, 9, 10\}, \{1, 3, 4, 7, 9, 10\}, \{3, 4, 5, 7, 9, 10\}$$

2.4 Explanation of Merge

In the discussion above we have made use of the term ‘merging of lists’. Here, we give an explanation of this procedure. Let’s say we have two lists of independent sets L_1, L_2 . Then merge produces a list which contains a union of each set in L_1 with each set in L_2 . Also, here we have L_1 and L_2 in gray code order ,i.e., in these lists every set differs from its predecessor by only one element. Now as a result of merge we want the newly constructed list to follow the gray code ordering. So, let’s say $\overline{L_2}$ is the reversed list of L_2 . Then, the resulting list is

$$L_3 = l_{11} \cup L_2, l_{12} \cup \overline{L_2}, l_{13} \cup L_2, \dots$$

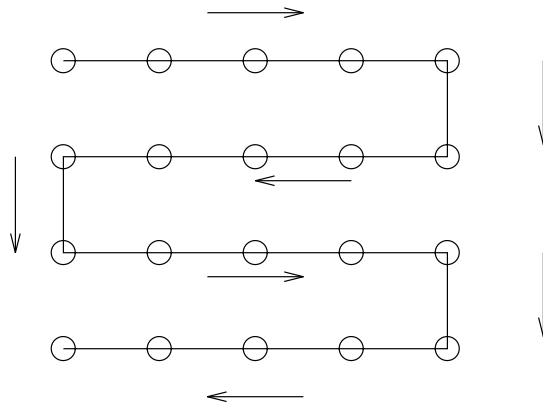


Figure 2.5: Zigzag traversal for merged list

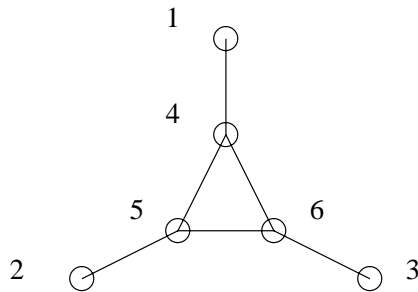


Figure 2.6: A counter example

where l_{ij} denotes j th element of i th list and $l_{ij} \cup L_k$ denotes a list obtained by union of set l_{ij} with each element of list L_k .

More than two lists can be merged recursively i.e.

$$\text{merge}'(L_1, L_2, L_3, \dots, L_n) = \text{merge}'(\text{merge}(L_1, L_2), L_3, \dots, L_n)$$

$$\text{merge}'(L_1, L_2) = \text{merge}(L_1, L_2)$$

Note that one end point of a merged list is the union of first sets of all the lists.

2.5 Enumerating Maximum Independent Sets in Chordal Graph

It is not possible to enumerate all the maximum independent sets in a general chordal graph in gray-code fashion. This can be shown by the following counter example. In the chordal graph shown in 2.6 the size of maximum independent set is 3.

The independent sets are:

1. {1,2,3}
2. {4,2,3}

3. $\{1,5,3\}$

4. $\{1,2,6\}$

We see that adjacency graph ¹ for these independent sets does not have Hamiltonian path.

¹the graph in which vertices represent independent sets and edges indicate that two independent sets differ in only one element

Chapter 3

Enumerating all k -sized independent sets in path

In this chapter, we describe a recursive algorithm to list all the independent sets of given size k in a path having n vertices. Here again we use induction to prove that this can be done using exchange of vertex operation. So, each independent set differs from its predecessor in only one element.

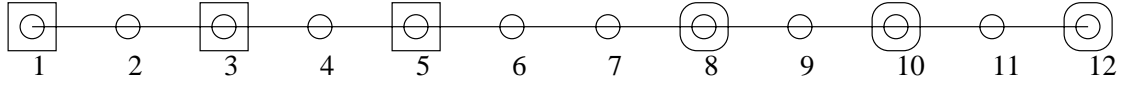
Here we look for one more property of listing that is first element of listing should be the leftmost independent set (i.e. $\{1,3,5,\dots,2k-1\}$) and the last should be the rightmost independent set (i.e. $\{n-2k+2,\dots,n-2,n\}$). Now, by induction, we assume that we have such a listing from the path between vertex 2 and vertex n . So what we want is the listing of all independent sets containing 1 such that it starts with the leftmost independent set and ends at the independent set which is adjacent (i.e. obtained by exchanging 1 vertex) to the first independent set in the listing of independent sets in the path starting from 2.

Theorem 3.1 *All independent sets of size k in a path of n vertices can be enumerated using exchange of vertex operation. Also, the first and the last independent sets in the enumeration are the leftmost and the rightmost independent sets respectively.*

Proof :

We use induction on number of vertices in the path.

Base Case : For $n = 1$ and $k = 1$ the only independent set is $\{1\}$. For $n = 2$ and $k = 1$, listing is $\{1\}, \{2\}$. For $n = 3$ and $k = 1$, the listing is $\{1\}, \{2\}, \{3\}$ and for $n = 3$ and $k = 2$ the listing is $\{1, 3\}$.



$n = 12$ leftmost $\{1,3,5\}$
 $k = 3$ rightmost $\{8,10,12\}$

Figure 3.1: An example of path with the first and the last independent set

Induction Step : We assume that the above theorem holds $\forall n < m$. Now, we consider a path with m vertices. Now, as discussed earlier, we first list all the independent sets containing vertex 1. Then by induction we assume that we have a listing of independent sets which do not contain 1. To get the required end points we split the first part into two subparts, one containing 3 and one not containing 3. So, our listing consists of three parts:

1. Independent sets including both 1 and 3.
2. Independent sets including 1 but not including 3.
3. Independent sets not including 1.

So in the first part we keep 1 and 3 in all Independent sets and the remaining parts of the sets are formed by listing of independent sets of size $k - 2$ in the path starting from 5. So the first in this list is $\{1, 3, 5, \dots, 2k - 1\}$ and the last in the list is $\{1, 3, n - 2k + 6, \dots, n - 2, n\}$. The second part is formed by keeping 1 constant and the remaining parts of the independent sets are formed by the listing of independent sets of size $k - 1$ in the path starting from 4. So the first in this list becomes $\{1, 4, 6, \dots, 2k\}$ and last is $\{1, n - 2k + 4, \dots, n - 2, n\}$. We note that last independent sets in the above two lists are adjacent. Also the first in the second list is adjacent to the first in the third list which is $\{2, 4, 6, \dots, 2k\}$. So we traverse the first list, then traverse the second list in reverse order and then traverse the third list.

So the listing is

$$\begin{aligned}
 & \{1, 3, 5, \dots, 2k - 1\} \dots \{1, 3, n - 2k + 6, \dots, n - 2, n\} \\
 & \{1, n - 2k + 4, \dots, n - 2, n\} \dots \{1, 4, 6, \dots, 2k\} \\
 & \{2, 4, 6, \dots, 2k\} \dots \{n - 2k + 2, \dots, n - 2, n\}
 \end{aligned}$$

This shows the listing and hence completes the proof.

□

3.1 Example

The listing of above algorithm for $n = 10$ and $k = 3$ is

{1, 3, 5 }	{1, 4, 10}	{3, 5, 9 }
{1, 3, 6 }	{1, 4, 6 }	{3, 5, 10}
{1, 3, 7 }	{2, 4, 6 }	{3, 8, 10}
{1, 3, 8 }	{2, 4, 7 }	{3, 7, 10}
{1, 3, 9 }	{2, 4, 8 }	{3, 7, 9 }
{1, 3, 10}	{2, 4, 9 }	{3, 6, 9 }
{1, 8, 10}	{2, 4, 10}	{3, 6, 10}
{1, 7, 10}	{2, 8, 10}	{3, 6, 8 }
{1, 7, 9 }	{2, 7, 10}	{4, 6, 8 }
{1, 6, 9 }	{2, 7, 9 }	{4, 6, 9 }
{1, 6, 10}	{2, 6, 9 }	{4, 6, 10}
{1, 6, 8 }	{2, 6, 10}	{4, 8, 10}
{1, 5, 8 }	{2, 6, 8 }	{4, 7, 10}
{1, 5, 9 }	{2, 5, 8 }	{4, 7, 9 }
{1, 5, 10}	{2, 5, 9 }	{5, 7, 9 }
{1, 5, 7 }	{2, 5, 10}	{5, 7, 10}
{1, 4, 7 }	{2, 5, 7 }	{5, 8, 10}
{1, 4, 8 }	{3, 5, 7 }	{6, 8, 10}
{1, 4, 9 }	{3, 5, 8 }	

3.2 Successor Algorithm

The above proof describes a recursive algorithm for listing the independent sets. Here we describe a successor algorithm which describes the way to obtain next independent set given a previous independent set. We also describe a predecessor algorithm which is just a counterpart of successor algorithm, and which is required to construct the successor algorithm. The following is a pseudo-code for successor (succ) and predecessor (pred) algorithms.

Successor Algorithm:

Check the first element of set call it as i .

Check the next

if it is $i + 2$ call succ(remaining part)

here if successor returns failure i.e. it is at last element of list

then update $i + 2$ to last most possible vertex i.e. $n - 2k + 6$.

if even this can't be done then algorithm returns saying last set.

if it is not $i + 2$ call pred(remaining part including the second element)

here if predecessor returns failure then update i to $i + 1$.

Predecessor Algorithm :

Check the first element of of set call it as i

Check the next

if it $i + 2$ call pred(remaining part)

here if predecessor returns failure then update i to $i - 1$.

If this is not possible then return failure. Note that predecessor before changing i to $i - 1$ checks if $i - 2$ is present in the set,if yes it returns failure.

if it is not $i + 2$ call succ(remaining part including the second element)

here if successor returns failure then change the above second element to $i + 2$.

3.3 Similarity between Path and Intervals

An independent set in a path can be seen equivalent to independent set of intervals in which every interval intersects with two others except for one which starts first and that which ends last. The set of intervals is as shown in figure 3.2.

Independent set of intervals is set of intervals such that no two intervals overlap.

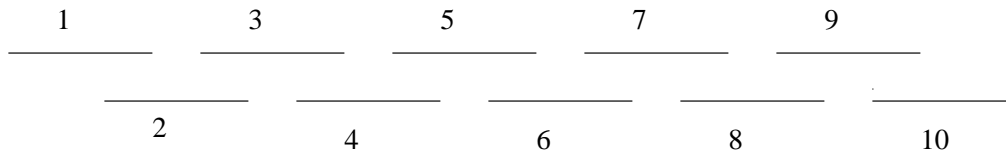


Figure 3.2: Set of intervals corresponding to path

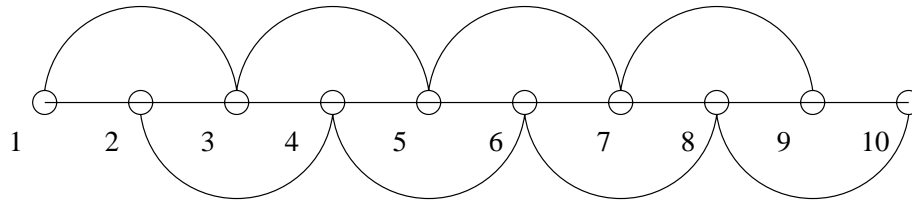


Figure 3.3: Extension of path

3.4 Extension

The solution above can be extended to the case, where every interval overlaps with 4 others except for the first and last. The corresponding graph instead of path here is as shown in 3.3.

The distinction here is that minimum difference between the numbers chosen for Independent Set is 3 instead of being 2 as in the previous case. Also, 3 can be further generalized to any number j .

Chapter 4

Enumerating k-sized Independent Sets in Interval Graphs

Here we discuss an algorithm to enumerate all k-sized independent sets in a particular class of interval graphs. Let's first begin with some definitions.

Definition 4.1 Interval Graph : *Given a set of intervals, we can construct a graph in which each vertex represents an interval and there is an edge between two vertices if the intervals corresponding to the vertices overlap.*

Definition 4.2 Proper Interval Graph : *A proper interval graph is an interval graph corresponding to the set of intervals in which no interval contains the other.*

4.1 Representation of Set of Intervals

Any set of intervals can be represented by a sequence of numbers which represent the rank of the first interval to the right of that interval with which it does not overlap. Here we assume that intervals are labeled (ranked) in ascending order of their starting points. We call this sequence as next. The figure 4.1 illustrates the definition of next.

Theorem 4.1 1. *All k-sized independent sets in a proper interval graph can be enumerated with operation exchange of interval, starting with leftmost independent set and ending at the rightmost independent set.*

2. *All k-sized independent sets in a proper interval graph with $V = \{1, \dots, n\}$ which include 1 can be enumerated with exchange operation, starting with leftmost*

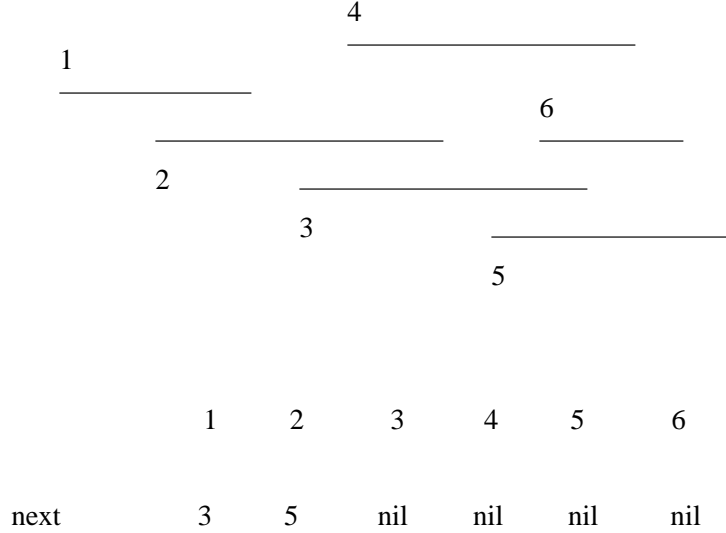


Figure 4.1: Representation of set of intervals

Independent set in $\{1, \dots, n\}$ and ending at the independent set which is adjacent to leftmost independent set in $\{2, \dots, n\}$, if $next(2) > next(1)$.

Proof :

We give an inductive proof for both the above parts of theorem.

Base Case : For $n = 1$ and $k = 1$ the only independent set is $\{1\}$. Also it can be verified for $n = 2, 3$.

Induction Step : We assume that both the above theorems are true for $n < m$.

Now we consider the interval graph with m vertices.

The proof of the first part of the theorem includes the proof of second part. For proving the first part we just have to prove the second part and rest of the listing can be obtained by induction. Here we consider two cases:

case1 : $next(2) > next(1)$. In this case we just need to prove the second statement.

case2 : let j be a smallest number such that $next(j) > next(1)$. So here all the vertices(intervals) numbered $1, 2, \dots, j - 1$ overlap with each other. All of these vertices(intervals) are interchangeable for each other in any independent set. So they can be considered as one vertex and then each independent set which has this vertex can be expanded into $j - 1$ independent sets. Also, adjacency can be maintained by expanding the first independent set in increasing order of labeling , expanding the second independent set in decreasing order of labeling , again increasing for the third and so on. So this case becomes similar to the first one.

The proof of second part : Let's say $i=next(1)$ and $j=next(2)$. We know that $i < j$. We include 1 in all the independent sets. Now, using induction on the second part we can have listing of all independent sets of size $k - 1$ in subgraph formed by

vertices $\{i, i+1, \dots, m\}$ which include i , start from leftmost independent set and end at independent set which is adjacent to leftmost independent set in the subgraph formed by vertices $\{i+1, \dots, m\}$. Here we assume that $\text{next}(i+1) > \text{next}(i)$. If not so, again they can be considered as a single vertex and the independent sets can be expanded. Repeating this for $i+1, i+2, \dots, j-2$ we reach to an independent set which is adjacent to the leftmost independent set of the subgraph formed by vertices $\{j-1, j, \dots, m\}$. So now for the listing of the rest of the independent sets we subdivide the list into two parts

1. including 1 and $j-1$.
2. including 1 but not $j-1$.

The remaining parts of the independent sets of the first part comes from the listing of $k-2$ sized independent sets in the subgraph formed by vertices $\{\text{next}(j-1), \dots, m\}$. The remaining parts of the second part of the listing comes from the listing of $k-1$ sized independent sets in the subgraph formed by vertices $\{j, \dots, m\}$. Appending this second part in reverse order maintains adjacency and gives the required end point. This completes the proof.

4.2 Example

The listing of the above algorithm for the set of interval shown in figure 4.1 with $k=2$ is

$$\{1, 3\}, \{1, 4\}, \{1, 6\}, \{1, 5\}, \{2, 5\}, \{2, 6\}$$

Chapter 5

Enumerating k-sized Independent Sets in Trees

Here, we have proved the existence of Gray code for a particular class of trees. The proof again is constructive and it directly gives an algorithm to enumerate the independent sets. The main idea used here is to use "shadows" of maximum independent sets. By shadows we mean all the k-sized subsets of a given maximum independent set. So, by this method we enumerate all the k-sized independent sets which are subsets of some maximum independent set in the tree. This forms an entire gray code listing for the trees in which all the k-sized independent sets are subsets of some maximum independent set.

5.1 Enumerating k-sized Independent Sets in Well Covered Trees

Definition 5.1 Well Covered Trees : *The trees in which all maximal independent sets have same size are called Well Covered Trees.*

Here we note that any maximal independent set is a maximum independent set. Hence any independent set can be extended to form a maximum independent set. i.e. all k-sized independent sets are contained in some maximum independent set.

Such trees are in general characterized as trees in which each non-leaf node is adjacent to exactly one leaf node. This means there is bijection between the set of leaf nodes and the set of non-leaf nodes. The tree of such kind has even number of vertices. Here the definition also includes a tree with just two vertices and an edge between them, in which case we consider one of them as leaf and the other as

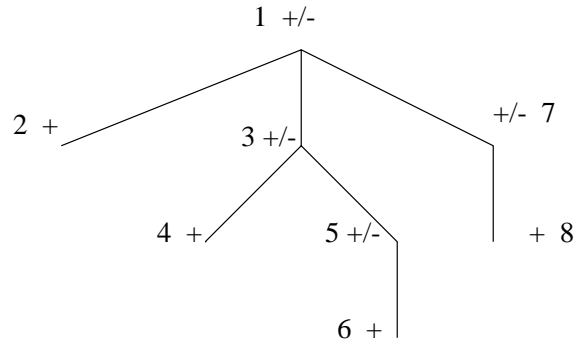


Figure 5.1: Example of a well covered tree

non-leaf.

We recall the labeling of the vertices as in chapter 1. Take a root to be an internal node. Here each leaf node would be a "+" node. Also each non-leaf node is "+/-" node.

Claim : All non-leaf nodes in a well covered tree are "+/-" nodes.

Proof by induction:

Base Case : $|V| = 2$ and the tree is a path of length one. Here in this particular case we consider one node to be leaf-node and other to be root (non-leaf nodes). We see that above claim is obviously true.

Induction Step : We assume that above claim is true for all such trees with $|V| < k$. Now consider a well covered tree with k (k even) nodes. We see that root has exactly one trivial subtree i.e. "+" node as children. All other children of the root are internal nodes. These nodes are roots of the subtrees which are also well covered. And hence by induction they are "+/-" nodes. So the root has exactly one "+" node as children. So the root is "+/-" node. Since any internal node is a root of smaller subtree, all the internal nodes of this tree are "+/-" nodes.

□

To prove the existence of gray code for shadows we have to consider some subtle issues like maintaining continuity between the shadows of two different maximum independent sets (which occur consecutively in the gray code listing for maximum independent sets as in chapter 2) and avoiding repetition because some k -sized independent set may be the subset of two different maximum independent sets.

Theorem 5.1 *There exists a Gray code for all k -sized shadows of the maximum independent sets and hence there exists a Gray code listing for all k -sized independent sets in well covered trees.*

Proof :

Consider the gray code listing of all maximum independent sets of any such tree as in chapter 2. Now for each maximum sized independent set list all the k -sized subsets of it which include the all "+/-" nodes present in this maximum independent set in standard gray code order. We call these "+/-" nodes as fixed nodes. The set consisting of these nodes as is called fixed set. So for each maximum independent set there is a fixed set which is subset of it. There is a list of these fixed sets corresponding to the list of maximum independent sets. We call this list as fixed-list.

Now we can show that in this enumeration, there is no repetition. Here each k -sized subset would be part of only that maximum independent set which contains all and exactly those "+/-" nodes which are in that k -sized subset.

While enumerating we take the gray code listing of maximum independent set and replace each maximum independent set in it by a gray code list of its k -sized subsets which contain the fixed nodes. So if l is the size of fixed list then for this listing we keep the fixed nodes in all the subsets and list the $k - l$ subsets of $k - l$ remaining nodes in maximum independent set in gray code order as in [2]. Note that ordering of elements of set is not important so the enumeration of these $k - l$ subsets can start with any subset. Here we choose the starting subset which would be adjacent to the last subset of the previous maximum independent set.

This adjacency can be obtained if fixed sets corresponding to those independent sets differ only by one element. That is the next fixed set has either one additional element or one element deleted or one element replaced. In this case ,while listing maximum independent sets each new maximum independent set is obtained by replacing a "+" node by "+/-" node or vice-versa. So in the fixed list each new fixed set is obtained by adding or deleting an element (" +/-" node) to/from the previous fixed set. So adjacency is maintained.

So only case which remains is when maximum independent set has $l > k$. In this case we do not enumerate any k -sized subsets at all. So we skip all the maximum independent sets with $l > k$. Now we need to ensure that the continuity is still preserved in the fixed list even after deleting the fixed sets of size $> k$. This is true because of the following lemma 5.1 . We note that all the fixed sets with size $> k$ appear in contiguous blocks between two k sized fixed sets. So we just have to make sure that these two k sized fixed sets are adjacent i.e. differ in only one element.

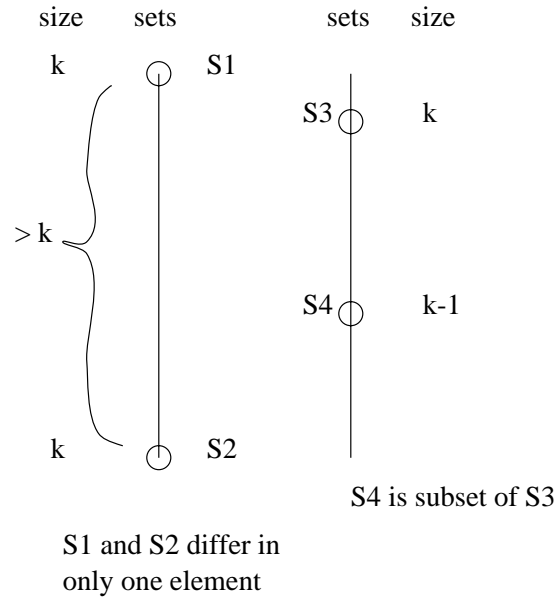


Figure 5.2: Lemma 5.1 and Lemma 5.2

Since this is true by the following lemma 5.1, we get the required gray code listing of k -sized independent sets in well covered trees.

□

Lemma 5.1 *For any given k , any two fixed sets of size k in the fixed-list which have all elements between them of size greater than k differ in exactly one element.*

Lemma 5.2 *In the listing of fixed sets i.e. fixed-list, in any of forward or reverse orders, the first k -sized fixed set and the first $(k-1)$ -sized fixed set, if occurring after it, differs in one element.*

We give a simultaneous inductive proof of both the above lemmas.

Proof :

First we prove that merge of chapter 2 section 2.4 preserves the above two properties. That is if all smaller list have the above properties than the list made by merging them also has the above properties. Since the elements of fixed list are just an images of some maximum independent sets and there is a bijection between gray code listing of maximum independent set and fixed-list, merge is also fundamental for the construction of fixed-list. In fact the fixed list is nothing but a merge of fixed lists of subtrees.

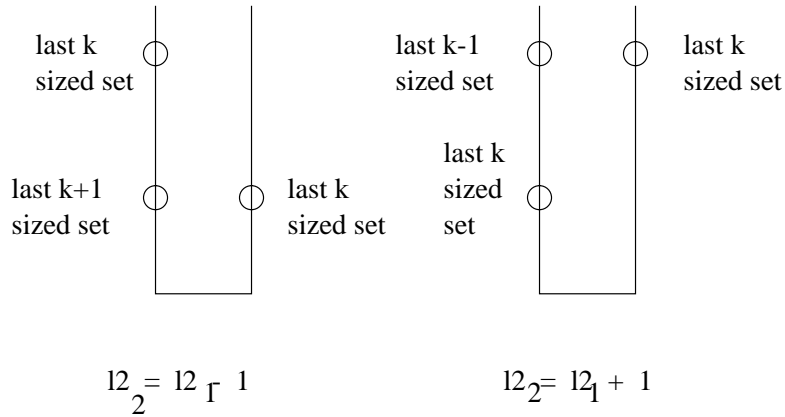


Figure 5.4: Subcases 1.1 and 1.2 of Case 1 for Lemma 5.1

If not so, i.e. all the fixed sets in the first branch after the k -sized fixed set are of greater size then we check out for the k -sized fixed set in the second branch. Now there are following subcases :

subcase 1.1 : $l_2 = l_1 - 1$ i.e. the next fixed set in fixed list 2 of merge deletes an element from the previous fixed set. Then look at the last (i.e. first from bottom) k -sized fixed set in branch 2. Then look at its $(k+1)$ -sized neighbour along the row towards the left. That is if the last k -sized fixed set in branch 2 is S_{j_2} look at S_{j_1} of size $k + 1$ in the branch 1. Now this set S_{j_1} is one element more than S_{j_2} . Now by lemma 5.2 assumption on branch 1 (essentially fixed list of subtree 1) S_{i_1} of size k and S_{j_1} of size $(k+1)$ are adjacent i.e. the latter has one more element. So S_{i_1} and S_{j_2} are adjacent (by replacement). Note that S_{i_1} can't be last element of branch one in this case since the next fixed set has to be of size $(k+1)$.

subcase 1.2 : $l_2 = l_1 + 1$ i.e. the next fixed set in fixed list 2 of merge adds an element to the previous fixed set. Then look at the last (i.e. first from bottom) k -sized fixed set in branch 2. Then look at its $(k-1)$ -sized neighbour along the row towards the left. That is if the last k -sized fixed set in branch 2 is S_{j_2} look at S_{j_1} of size $k - 1$ in the branch 1. Now this set S_{j_1} is one element less than S_{j_2} . Now by lemma 5.2 assumption on branch 1 (essentially fixed list of subtree 1) S_{i_1} of size k and S_{j_1} of size $(k-1)$ are adjacent i.e. the latter has one less element. So S_{i_1} and S_{j_2} are adjacent (by replacement). But if in this case , if S_{i_1} of size k is minimum sized fixed set in branch 1 then we can not find such j . But note that minimum sized fixed set implies l_1 minimum in the fixed list 1. So the next k -sized fixed set would appear in the same row when l_2 becomes equal to l_1 and all l_2 's between them of greater size. So by induction (lemma 5.1) assumption on fixed list 2 we get the property that S_{i_i} and S_{j_i} differ in exactly one position.

Case 1 for lemma 5.2: Both the k -sized fixed set and $(k-1)$ -sized fixed set are in the same part of the fixed list i.e. either in including (root) part or excluding part. In this case it suffices to prove that merge preserves the property in lemma 5.2. Now without loss of generality assume that first k -sized fixed set occurs in first branch of merge. Call it as S_{i1} . Now if the first $(k-1)$ -sized fixed set, if occurring after it, is in the same branch then by induction (lemma 5.2) assumption on first subtree we are done. Otherwise S_{i1} is the fixed set of minimum size in branch 1. Now we traverse the matrix S along the row i because next $(k-1)$ sized fixed set can only be in this row. This element would appear when we get first j such that $l2_j = l2_1 - 1$. Again by induction (lemma 5.2) assumption on fixed list 2 we S_{i1} of size k and S_{ij} of size $(k+1)$ are adjacent.

So, this proves that merge preserves the properties in lemma 5.1 and lemma 5.2. Hence lemma 5.1 and lemma 5.2 are proved for the case when both the concerned fixed sets are in the same part of the listing i.e. either both in including (root) part or both in excluding part. So the case which remains is when they fall in different parts of the listing. First we prove lemma 5.2 in this case.

Case 2 for lemma 5.2 : Here we note that minimum size of the fixed set in non-including part of the fixed list is 0. This happens when the corresponding maximum independent set consists of all leaf nodes. So if we see the fixed list in reverse order (starting from bottom) then if first $k-1$ sized fixed set appears after first k sized fixed set then both of them have to occur before this minimum sized fixed set of size 0. So $k=0$. Hence there is a contradiction and this subcase can't arise. Similarly minimum size of fixed set in the including part of fixed list is 1. So when we see the list in forward order (starting from top) both the concerned fixed sets should appear before this minimum sized fixed set, if $k > 1$. So $k = 1$. In this subcase adjacency is trivial.

Now, we come to our final case.

Case 2 for lemma 5.1 : We observe that including part of fixed list is a merge of non-including parts of fixed lists of subtrees with an additional element (root node) added to each fixed set in this part of the list. Non including part of this listing is merge of fixed lists of subtrees taken in reverse order. Let S denote the matrix of order $p \times q$ of the fixed sets in non-including part. Let S' be the matrix of order $p_1 \times q_1$ of the fixed sets in including part. The entire fixed list is then $\dots, S'_{12}, S'_{22}, \dots, S'_{p_1 2}, S'_{p_1 1}, \dots, S'_{11}, S_{11}, \dots, S_{p1}, S_{p2}, \dots, S_{12}, \dots$. Note that p is number of fixed sets in the fixed list of subtree 1 and p_1 is number of fixed sets in the non including part of the fixed list of subtree 1. q is the number of fixed sets in the fixed list obtained by merge of remaining subtrees (2,3, ...) and q_1 is the number fixed sets

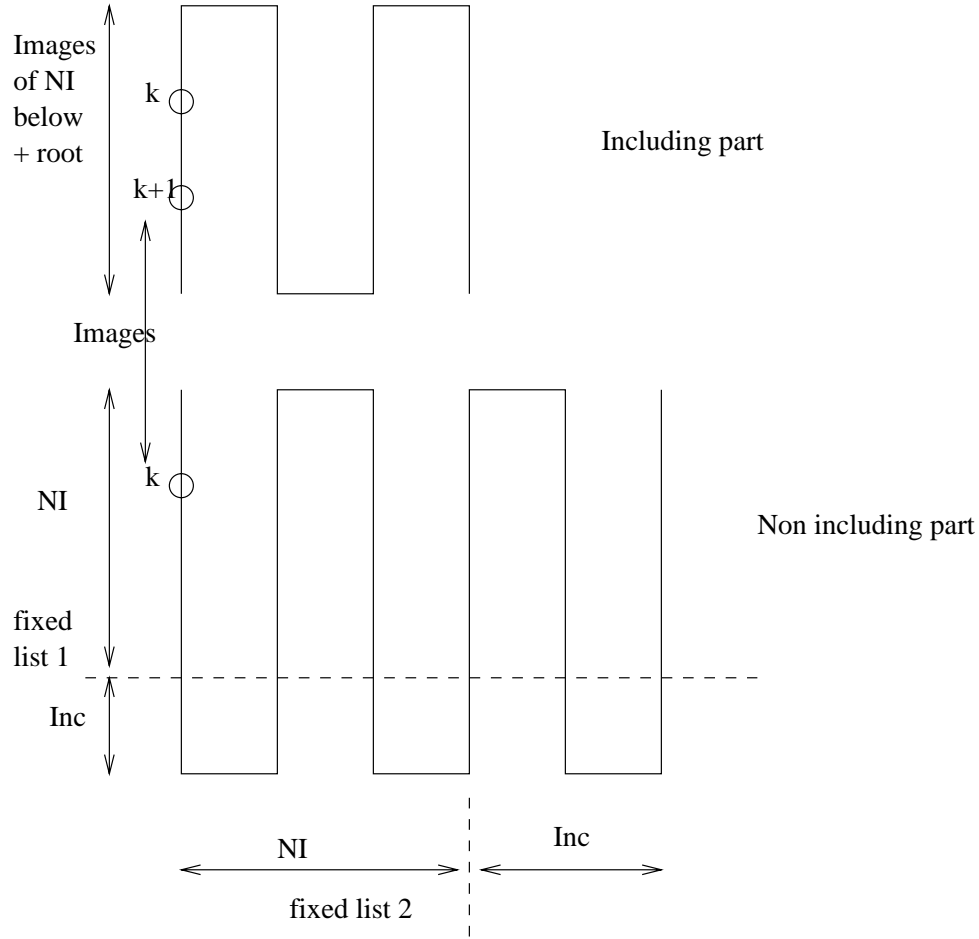


Figure 5.5: Case 2 for lemma 5.1

in the non including part of the same.

Now for all i from 1 to p_1 and for all j from 1 to q_1 , $S'_{ij} = S_{ij} \cup \{root\}$. Refer to figure 5.5.

So we look for the last k -sized fixed set in the listing of including part. Let it be S'_{lm} . Now look at the last $(k+1)$ -sized fixed set in the listing of including part. Let it be S'_{ij} . Note that these two sets are adjacent (differ in one position) by lemma 5.2. Now look at S_{ij} . This is k -sized fixed set in non including part which is one element (root) less than S'_{ij} . So S_{ij} is adjacent to S'_{lm} . Now we just have to show that S_{ij} is the first k -sized element in the non including list.

This would be obviously true if rows (extra rows) $p_1 + 1, \dots, p$ of S were ignored. Because here we have S'_{ij} as last $k+1$ sized fixed set in including part so its image S_{ij} in non including part will be the first k sized fixed set. So if $j = 1$ i.e. the last $k+1$ sized fixed set occurs in the column one (i.e. last branch of merge for including part) then we need not bother about the extra rows.

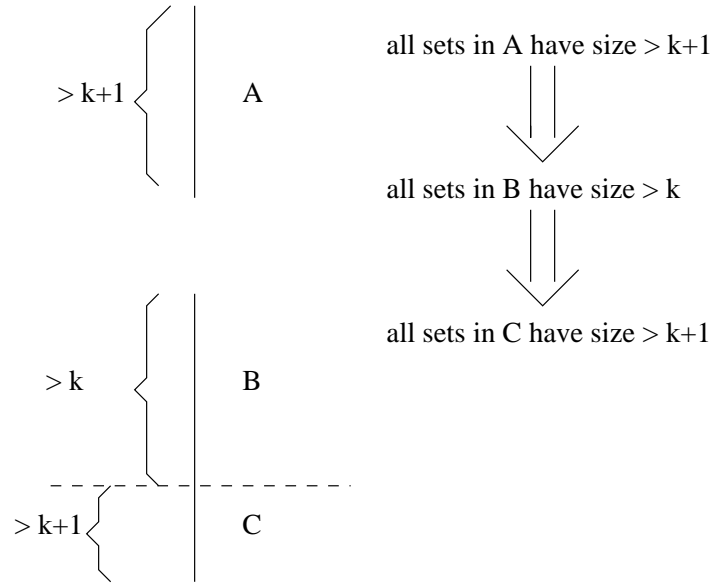


Figure 5.6: Sizes of various sets in the branches

If last $k+1$ sized fixed set in including part is not in last branch (i.e. if $j > 1$) then all columns 1 to $j - 1$ in S' have sets of size $> k+1$. Then any column r from 1 to $j - 1$ in S will have all sets of size $> k$. This is easy to see for rows 1 to p_1 of that column since they are images of fixed sets of column r of S' . In column r , the sets in rows $p_1 + 1$ to p correspond to including part of fixed list of subtree 1. So each of these has some image (of size one less) in rows 1 to p_1 . So all these sets are of size $> k+1$.

So all the columns 1 to $j - 1$ can be ignored. And j th column can be considered equivalent to the first column so this fall into the above case.

□

5.2 Enumerating k -sized shadows of maximum independent sets in trees

Here we give an algorithm to enumerate all the k -sized independent sets which are subsets of some maximum independent sets in trees. That is we prove here that there exists a gray code for k -sized shadows of maximum independent sets in tree. We use exactly same algorithm (proof) as in section 5.1 with some modifications. In this section we just figure out those modification the rest of the proof is exactly the same.

Here we fix only those "+/-" nodes which are free in the particular independent set.

That is for a particular maximum independent set the corresponding fixed set does not include all the "+/-" nodes but only those "+/-" nodes which can be substituted by some neighbouring vertex to obtain some other maximum independent set. Because if a non-free "+/-" vertex is fixed then there is no other maximum independent set which would include all the k-sized subsets of this maximum independent set which do not include that particular "+/-" node. Note that each entry in the fixed list is still unique. And any k-sized subset would be enumerated only as a part of that maximum independent set which has exactly those "+/-" nodes free.

Here each next fixed set in the fixed list can be obtained from the previous one by adding, deleting or replacing the vertex. This gives one more subcase in case 1 for lemma 5.1. That is when $l2_1 = l2_2$. In this case we directly look sideways that is if S_{j1} is the first k-sized set then S_{j2} will be the next k-sized set which can be obtained from the previous by induction on second fixed list of merge.

All the other cases are exactly the same or the changes required are trivial.

5.3 Example

Here we work out the example of above algorithm with the tree in figure 5.1 and $k = 2$. Note that maximum size of independent set in this tree is 4.

Max. Ind. set	Fixed set	2-sized subsets
{1,4,6,8}	{1}	{1,4},{1,6},{1,8}
{1,4,5,8}	{1,5}	{1,5}
{2,4,5,8}	{5}	{2,5},{4,5},{5,8}
{2,4,6,8}	{ }	{6,8},{4,8},{2,8},{2,6},{4,6},{2,4}
{2,3,6,8}	{3}	{2,3},{3,6},{3,8}
{2,3,6,7}	{3,7}	{3,7}
{2,4,6,7}	{7}	{6,7},{4,7},{2,7}
{2,4,5,7}	{5,7}	{5,7}

Here column 3 gives the required list.

Chapter 6

Conclusions

The main emphasis in this project has been on proving the existence of Gray code for independent sets. All the proofs given use induction and hence corresponding algorithms were recursive. We haven't really bothered about efficiency of algorithms or the size of data structures involved.

Some more general problems like enumerating k -sized independent sets in general trees and general interval graphs are yet unsolved. We have only proved the existence of gray codes for some subclasses of the above.

References

- [1] P.M. Deshpande. Enumeration of combinatorial objects. In *B.Tech Project*, 1994.
- [2] H. Wilf. *Combinatorial Algorithms : an update*. Philadelphia : SIAM ,1989, 1st edition, 1989.

Acknowledgements

5th April, 1997

I would like to thank my guide **Dr. A. A. Diwan** for several reasons. First, for offering me such a wonderful project. Then for some excellent ideas he gave without which this project would not have been possible. And lastly for his constant guidance because of which this project took the right direction.

Rahul T. Shah