Prufer Coding: A Vectorization Method for Undirected Labeled Graph.

Lin Yang, Yongjie Wang

IEEE Access 8: 175360-175369 (2020)

Presenter:

資工三 盧玉隆

資工三 楊雨樓

Outline

INTRODUCTION

INTRODUCTION

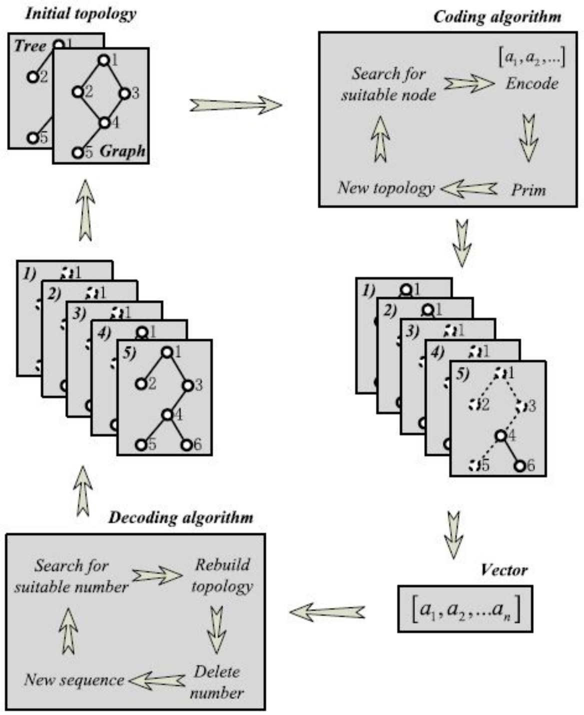
- Graph vectorization is introduced to simplify the graph representation, representing the topology information of a graph as a vector through a graph transformation algorithm
- The traditional prufer algorithm is a method of coding and decoding labeled trees, which can be used to vectorize unrooted trees
- Prufer sequence can be applied on some special occasions, such as
  - be integrated into the design of the Genetic Algorithm
  - solve the Minimum Spanning Tree (MST) problem
  - improve the performance of identifying and discovering complex matches of XML schema matching framework
  - ...

INTRODUCTION

- Although scholars have proposed many improved methods for the traditional prufer algorithm, no one has considered applying it to graph vectorization
- our contributions could be summarized as follows
  - Propose a method for undirected labeled graph vectorization based on the prufer algorithm, including graph encoding and decoding algorithm.
  - Propose a method to check the connectivity of the graph based on the Warshall algorithm and introduce an improved approach, then apply it to increase the accuracy of the prufer algorithm in coding and decoding undirected labeled graph.
- Finally, the application prospect of the graph vectorization in topology calculation will be analyzed

INTRODUCTION

- Process block diagram of topological vectorization.



REVIEW OF THE PRUFER ALGORITHM

PRUFER CODING OF ROOTLESS LABELED TREE

- The coding steps are summarized as follows
  - step_1: Cut the leaf nodes and edges in order from small to large according to vertex labels
  - step_2: Record the node number that connected to the leaf node on the trimmed edge
  - step_3: Repeat step_1 and step_2 until only two nodes and edges between them are left in the tree, the algorithm is end.

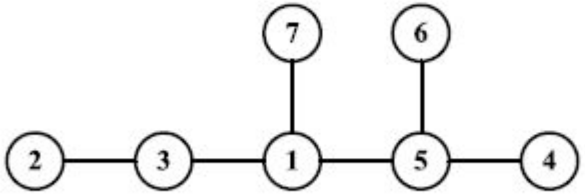
PRUFER CODING OF ROOTLESS LABELED TREE

- example:
  - according to the coding step_1, node 2 and the edge (2, 3) with the smallest sequence number among the leaf nodes are cut out to generate a new tree.
  - According to the coding step_2, record the node number 3 adjacent to node 2, so the current prufer sequence is 3
  - The prufer sequence changes as follows:

    [3]->[3,1]->[3,1,5]->[3,1,5,5]->[3,1,5,5,1]

  - Finally, the prufer sequence is

    [3, 1, 5, 5, 1].



A rootless tree composed of 7 nodes.

PRUFER DECODING OF ROOTLESS LABELED TREE

- Provide two sequences 1,2,...., n and b1,b2,.....,bn-2, which are sequential sequence (SeqtSeq) and prufer sequence (PruferSeq),
- Decoding steps are as follows:
  - step_1: Construct the SeqtSeq according to the node number of the tree. Find the number that is not in PruferSeq and is located on the leftmost side of the SeqtSeq. Connect it to the leftmost number of the SeqtSeq to rebuild this edge.
  - step_2: After completing the step_1, the two node numbers of SeqtSeq and PruferSeq are eliminated to form two new sequence.
  - step_3: Repeat step_1 and step_2 several times until only two numbers left in the SeqtSeq. Then rebuild the edge corresponding to the remaining two numbers, and the algorithm terminates.

PRUFER DECODING OF ROOTLESS LABELED TREE

- take the above tree T as an example: PruferSeq that we get is [3,1,5,5,1], and construct the SeqtSeq: [1,2,3,4,5,6,7]
- the leftmost sequence number that in SeqtSeq but not in PruferSeq is 2, and the leftmost number of the SeqtSeq is 3, so rebuild edge (2,3)
- According to the decoding step_2, delete the number 2 in SeqtSeq and the leftmost number 3 in PruferSeq
- the above processes will be repeated until SeqtSeq has only two numbers left: [1,7], finally rebuild the edge (1,7)

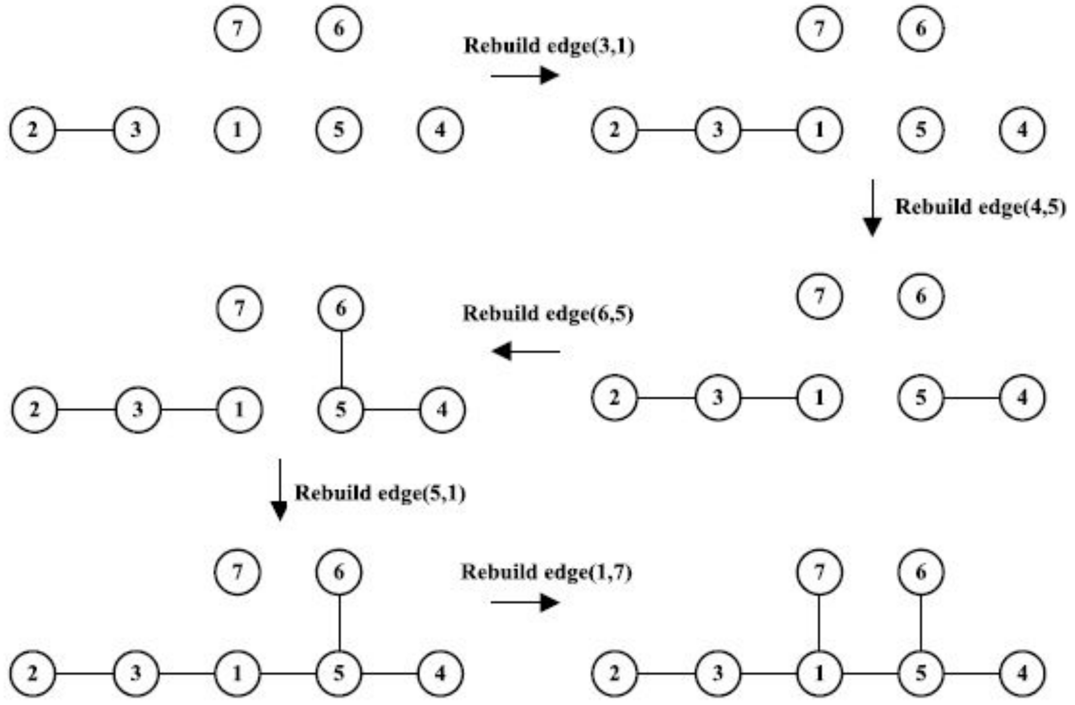
PRUFER DECODING OF ROOTLESS LABELED TREE

- The changes of SeqtSeq and PruferSeq are shown below

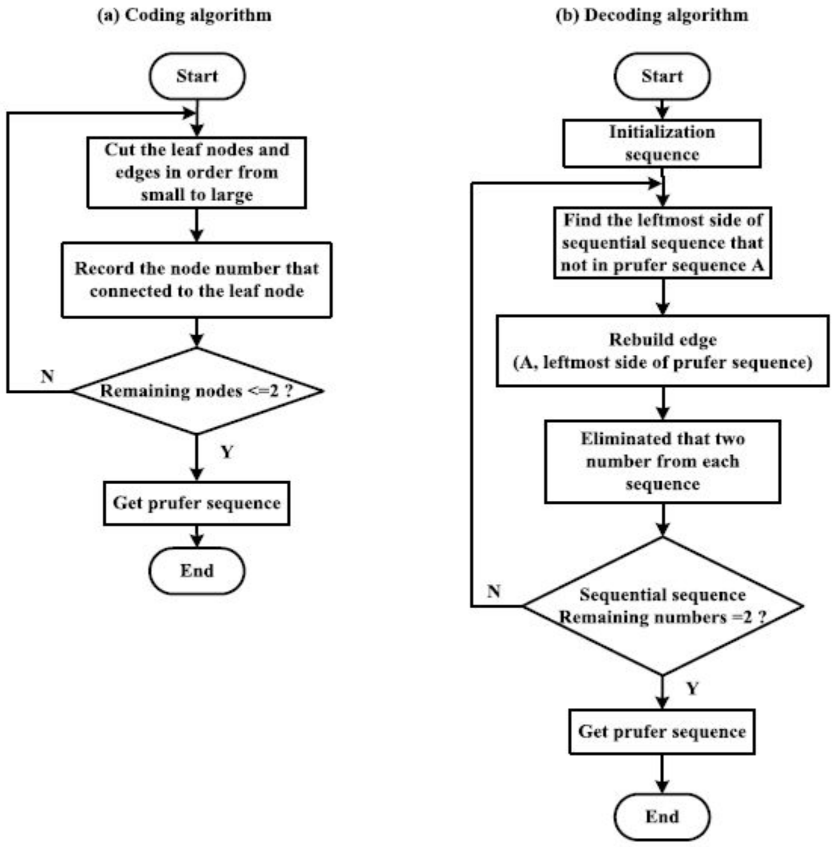
$$
\begin{aligned}
&\begin{cases} [1, \textcircled{3}, 4, 5, 6, 7] \\ [\textcircled{1}, 5, 5, 1] \end{cases}
\rightarrow \begin{cases} [1, \textcircled{4}, 5, 6, 7] \\ [\textcircled{5}, 5, 1] \end{cases}
\rightarrow \begin{cases} [1, 5, \textcircled{6}, 7] \\ [\textcircled{5}, 1] \end{cases} \\
&\rightarrow \begin{cases} [1, \textcircled{5}, 7] \\ [\textcircled{1}] \end{cases}
\rightarrow \begin{cases} [1, 7] \\ \phi \end{cases}
\end{aligned}
$$

PRUFER DECODING OF ROOTLESS LABELED TREE

- Edges (3,1), (4,5), (6,5), (5,1), (1,7) will be decoded in order, as shown below



REVIEW OF THE PRUFER ALGORITHM


(a) Coding algorithm
Start
Cut the leaf nodes and edges in order from small to large
Record the node number that connected to the leaf node
Remaining nodes <=2 ?
N
Y
Get prufer sequence
End
(b) Decoding algorithm
Start
Initialization sequence
Find the leftmost side of sequential sequence that not in prufer sequence A
Rebuild edge (A, leftmost side of prufer sequence)
Eliminated that two number from each sequence
Sequential sequence Remaining numbers =2 ?
N
Y
Get prufer sequence
End

REVIEW OF THE PRUFER ALGORITHM

- Using the integer sorting algorithm obtained by the particularity of the integer values to be sorted, the prufer encoding and decoding problems are simplied to integer sorting problems, which can better improve the efciency of rootless tree prufer coding and decoding.
- uses simple arrays to improve prufer algorithm, which can improve the time complexity of prufer coding to O(n).
- scanned the prufer sequence in reverse order, and it’s proved that the algorithm could run in linear time without the need for additional data structures or sorting processes.

PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

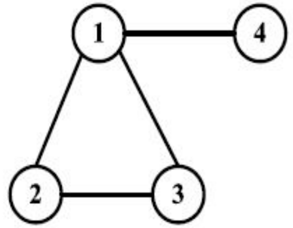
PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

- Traditional prufer algorithms can be used to encode and decode a rootless labeled tree
- However, compared to a rootlesstree, graphs are more widely used to solve network problems, so it is necessary to design a method for coding and decoding labeled graphs
- Compared with a tree, the graph has a unique structure, which is named cycle.
- for example:
  - By coding graph G, we will nd some problems:if use the

    prufer coding method of the rootless tree to coding graph G,

    the node 4 and edge (1,4) in graph G will be trimmed first.

    Then the remaining nodes 1, 2, and 3 form a cycle, where there are no more leaf nodes



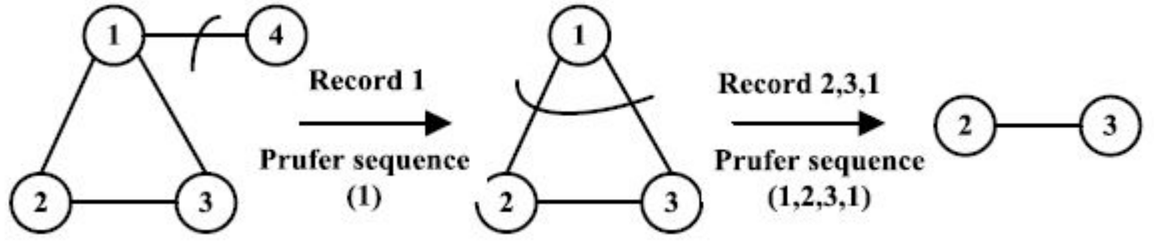
A simple undirected labeled graph G.

PRUFER CODING OF UNDIRECTED LABELED GRAPH

- Suppose that the n vertex of the undirected labeled graph G is denoted as a1,a2,...,an. The coding steps are designed as follows:
    - step_1: If the current undirected labeled graph has leaf nodes ai,...., aj, cut out the smallest node a_min among the leaf nodes, as well as the edge (ai, bi) formed with the adjacent node bi. If there is no leaf node left, the one with the smallest sequence number among the remaining nodes will be trimmed.
    - step_2: If the clipped node is a leaf node, only its adjacent node bi should be recorded; if the clipped node is not a leaf node and its degree is j(j >= 2), All nodes b1,b2,...,bj that connected to ai through edges (ai,b1); (ai,b2),..., (ai,bj) should be recorded, assuming that b1 < b2 < ... < bj follow the order from small to large, record all of them and add the clipped node ai at the end to generate a sequence [b1, b2,..., bj,ai].
    - step_3: When each trim is complete, a new undirected labeled graph will be generated. Continue to repeat step_1 and step_2 until the undirected labeled graph has only two nodes left, and the algorithm terminates.

PRUFER CODING OF UNDIRECTED LABELED GRAPH

- Taking graph G as an example graph, its coding process is shown below

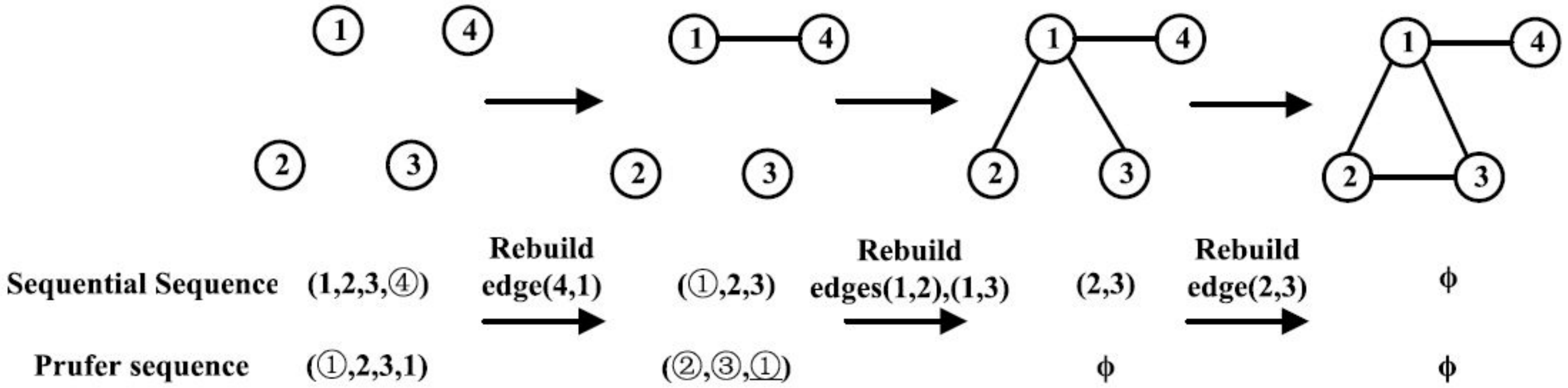


PRUFER DECODING OF UNDIRECTED LABELED GRAPH

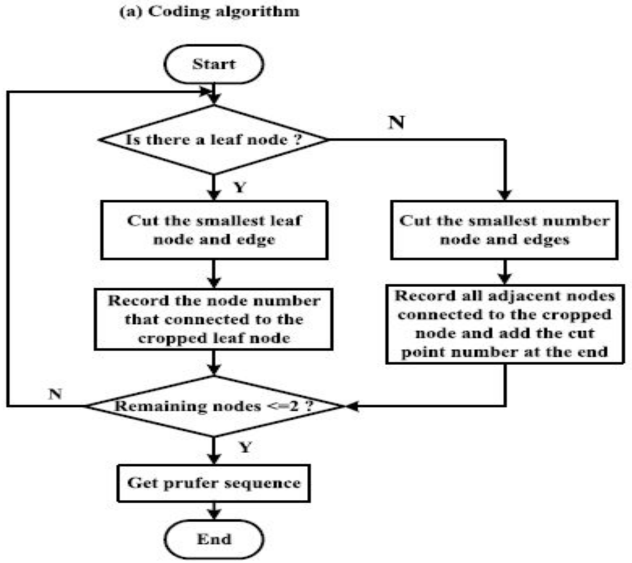
- Decoding steps could be designed as follows:
  - step_1. Find the number a that located in the leftmost side of SeqtSeq but does not exist in the PruferSeq.
  - step_2. Connect the node a with the node b that located in the leftmost side of PruferSeq, rebuild edge (a; b).
  - step_3. Delete a in SeqtSeq and b in PruferSeq.
  - step_4. If the above number a does not exist, nd the position of the leftmost side number of SeqtSeq in PruferSeq, mark that number as bj.
  - step_5. Connect the node bj with each node (b1,...,bj-1) that in front of bj, in order to rebuild edges (bj,b1),...,(bj,bj-1).
  - step_6. Delete number bj in SeqtSeq and all numbers b1,...,bj-1,bj in PruferSeq;
  - step_7. Repeat the above process until there are only two numbers left in SeqtSeq. Connect the remaining two numbers, rebuild the nal edge, the algorithm is over.

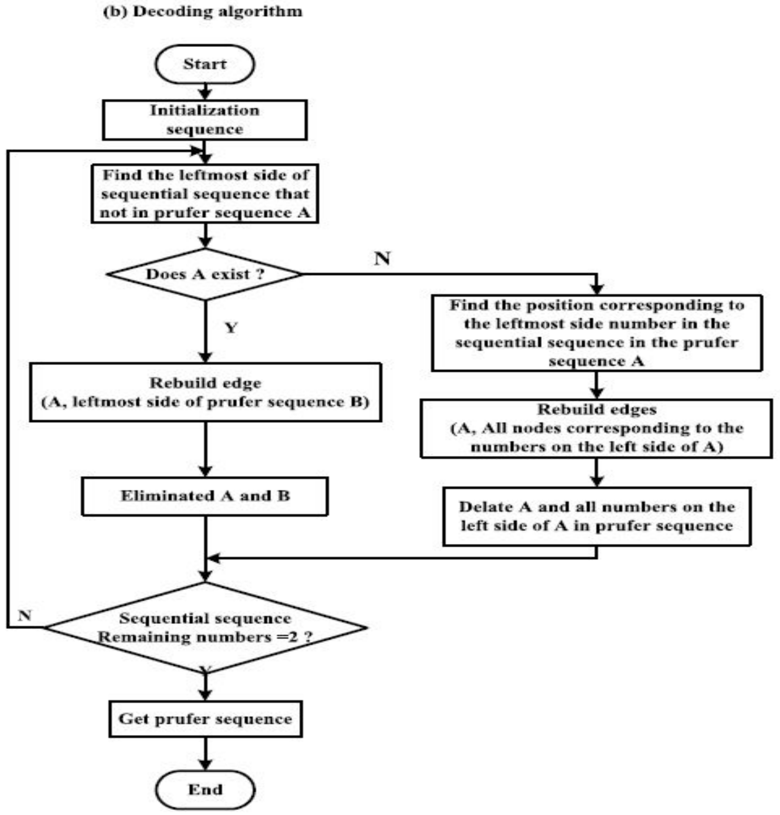
PRUFER DECODING OF UNDIRECTED LABELED GRAPH

- Taking the undirected labeled graph G as an example



Prufer coding and decoding flowcharts for undirected labeled graph.


(a) Coding algorithm
Start
Is there a leaf node ?
N
Y
Cut the smallest leaf node and edge
Record the node number that connected to the cropped leaf node
Cut the smallest number node and edges
Record all adjacent nodes connected to the cropped node and add the cut point number at the end
N
Remaining nodes <=2 ?
Y
Get prufer sequence
End

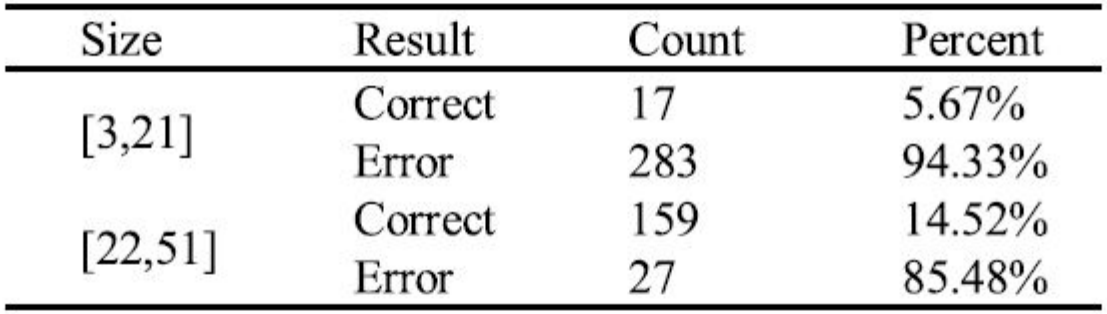

(b) Decoding algorithm
Start
Initialization sequence
Find the leftmost side of sequential sequence that not in prufer sequence A
Does A exist ?
N
Y
Find the position corresponding to the leftmost side number in the sequential sequence in the prufer sequence A
Rebuild edge (A, leftmost side of prufer sequence B)
Rebuild edges (A, All nodes corresponding to the numbers on the left side of A)
Eliminated A and B
Delate A and all numbers on the left side of A in prufer sequence
N
Sequential sequence Remaining numbers =2 ?
Y
Get prufer sequence
End

time complexity

- The basic operation of the coding algorithm is to determine whether a node is a leaf node; its time complexity is O(n). It needs to determine the remaining nodes number, so its time complexity is also O(n), therefore the time complexity of the coding algorithm is O(n^2)
- The basic operation of the decoding algorithm is to find the node number that in the leftmost of SeqtSeq but not in PruferSeq, its time complexity is O(n^2), so the time complexity of decoding algorithm is O(n^3).
- The optimal time complexity of basic operation can reach O(n log n). Meanwhile, the time complexity of the outer while loop can be reduced to O(log n) by selecting the appropriate data storage structure, so the optimal coding and decoding algorithm time complexity are O(n log n) and O(n log^2 n) respectively.

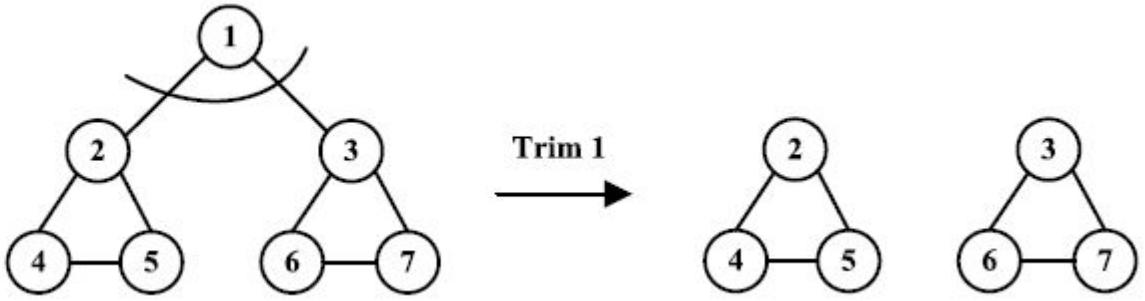
PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

- For undirected labeled graphs with different node size scales, a large of experiments have been carried outs.
- The accuracy rate of the codec still has not reached 100%



Size	Result	Count	Percent
[3,21]	Correct	17	5.67%
	Error	283	94.33%
[22,51]	Correct	159	14.52%
	Error	27	85.48%

PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

- the original graph will be divided into two or more graphs in some particular cases. In this situation, the algorithm execution result will be wrong,
- 

PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

- In order to solve this problem, the shearing condition needs to be added
- TheWarshall algorithm uses the idea of dynamic programming to find transitive closures, which can be used to judge the connectivity of the graph
- a vector can be introduced to record the reachability of a single node, so that the n power of the adjacency matrix represents the number of paths that each node can reach through n hops to another node (including itself)
- the connectivity detection algorithm can be designed as follows
- When considering this particular case, it means that there is no leaf node at present, so a cycle will appear, it will accelerate the check
- Only in the worst case, the outer loop needs to be performed n-2 times
- the time complexity of this check algorithm is O(n log n).

Algorithm 3 Connectivity Check

Input:

undirected labeled Graph G

Output:

connectivity judgment result

```
1: algorithm ConnectionCheck (G)
2: CheckLine ← G (1)
/*Use the first row of the adjacency matrix for inspection*/
3: for PowerCount ← 2 to G.nodenum
4:     CheckLine ← CheckLine and G (1)
/*CheckLine performs AND operation with the first row of
the current matrix*/
5:     if AllOnes (CheckLine)
/*If CheckLine is all 1 then return true*/
6:         return TRUE
7:     end if
8:     G ← G * G_Init; // Continue multiplication
9: end for
10: return FALSE
11: end algorithm
```

PRUFER ALGORITHM FOR UNDIRECTED LABELED GRAPH

- Therefore, the algorithm needs to make the following improvements
- If it is found that trimming the current non-leaf node will divide the original graph into multiple graphs, then mark and skip this node until a node that does not decompose the original graph is found, exchange it with the smallest marked node, and record this exchange in order to recover when decoding

ALGORITHM APPLICATION

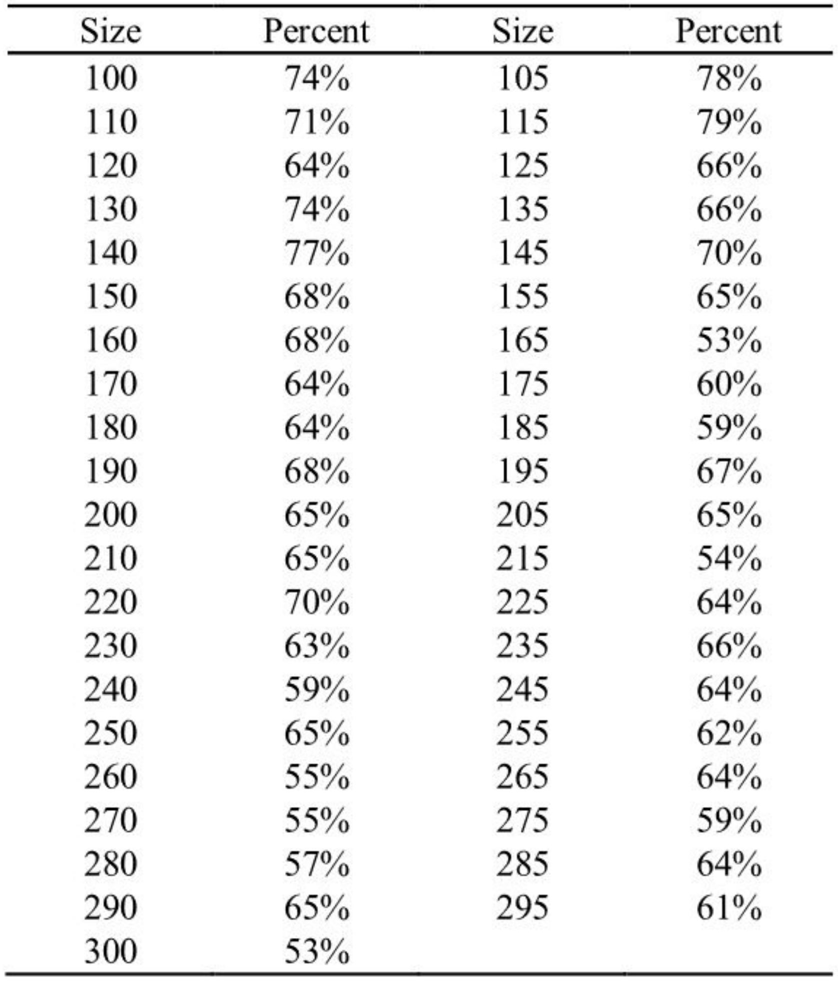
Algorithm Application

- Prufer Encoding such as the method proposed in this paper can sometimes greatly simply graph operations and improve the efficiency in solving graph-related problems.
    - This is due to how prufer coding records the “useful” information regarding the connectivity of graphs, as opposed to recording all information through methods such as the adjacency matrix.
    - This difference is more significant the more sparse the graphs are.
- Another application would be the generation of of random graphs that meet specific criteria using these sequences.

EXPERIMENT

Algorithm Accuracy

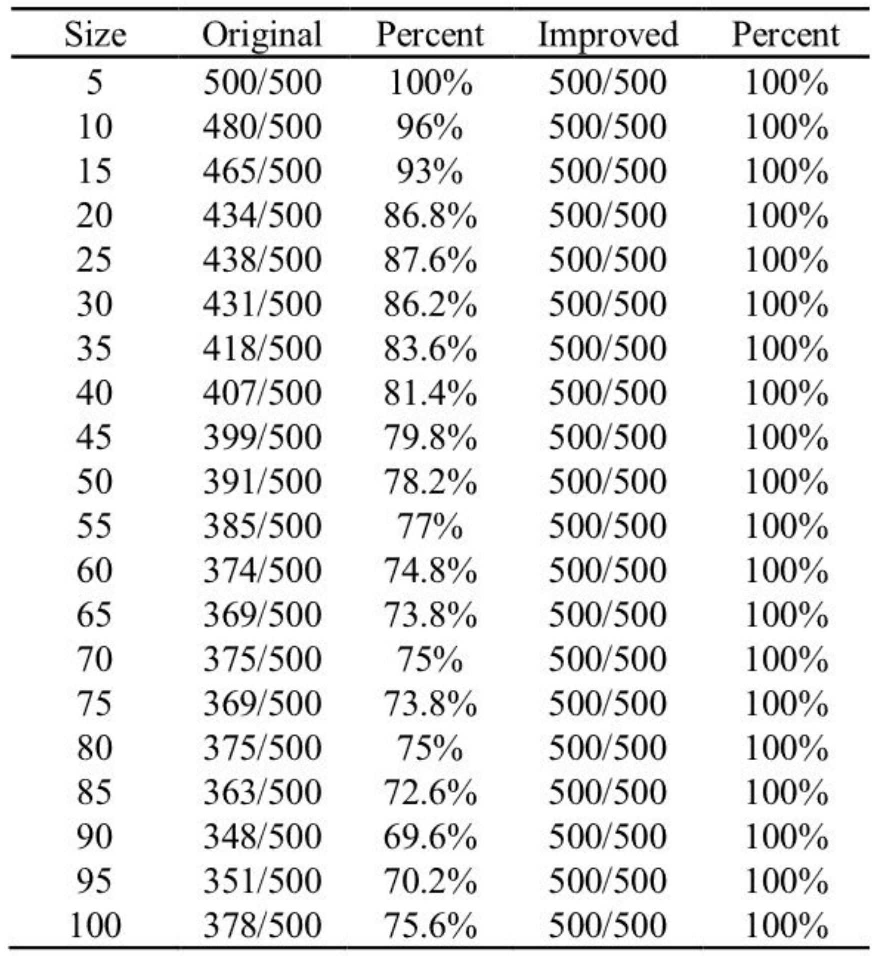
- Experiments show that the accuracy of the naïve method decreases as the size of the graphs increase, due to the increased chance of aforementioned errors.



Size	Percent	Size	Percent
100	74%	105	78%
110	71%	115	79%
120	64%	125	66%
130	74%	135	66%
140	77%	145	70%
150	68%	155	65%
160	68%	165	53%
170	64%	175	60%
180	64%	185	59%
190	68%	195	67%
200	65%	205	65%
210	65%	215	54%
220	70%	225	64%
230	63%	235	66%
240	59%	245	64%
250	65%	255	62%
260	55%	265	64%
270	55%	275	59%
280	57%	285	64%
290	65%	295	61%
300	53%		

Algorithm Accuracy

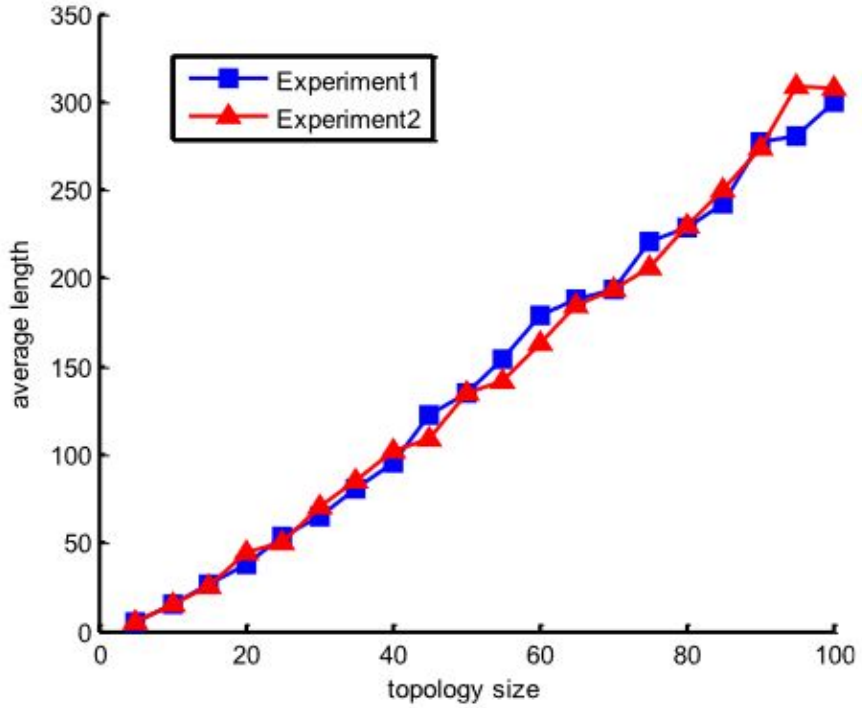
- Even though the improved algorithm is significantly more time-consuming, the advantage it has in accuracy vastly outweighs the cost.
- This advantage becomes more important as the size of graphs increase.



Size	Original	Percent	Improved	Percent
5	500/500	100%	500/500	100%
10	480/500	96%	500/500	100%
15	465/500	93%	500/500	100%
20	434/500	86.8%	500/500	100%
25	438/500	87.6%	500/500	100%
30	431/500	86.2%	500/500	100%
35	418/500	83.6%	500/500	100%
40	407/500	81.4%	500/500	100%
45	399/500	79.8%	500/500	100%
50	391/500	78.2%	500/500	100%
55	385/500	77%	500/500	100%
60	374/500	74.8%	500/500	100%
65	369/500	73.8%	500/500	100%
70	375/500	75%	500/500	100%
75	369/500	73.8%	500/500	100%
80	375/500	75%	500/500	100%
85	363/500	72.6%	500/500	100%
90	348/500	69.6%	500/500	100%
95	351/500	70.2%	500/500	100%
100	378/500	75.6%	500/500	100%

Prufer Sequence Length

- Main disadvantage of the proposed method is the varying length of the generated Prufer code depending on the topology of the graphs.
- This makes it harder to align the codes together to do matrix operations when solving graphs problems.
- Even so, experiments show that the code generated by this method has an average length of $l = n^{1.237}$, significantly better than using adjacency matrices. (n is the number of nodes in the graph.)



CONCLUSION

Conclusion

- This paper proposed a method of expanding the Prufer algorithm to undirected graphs, presented their experimental results on the algorithm, and discussed its application.
- The time complexity of this algorithm is shown to be within manageable range.
- Their experiments demonstrated 100% accuracy for their improved algorithm, but they cannot to provide theoretical proof for it to work in all situations.
- Both the benefits and drawbacks of this method are discussed in this paper regarding its applications.

Reference

L. Yang and Y. Wang, "Prufer Coding: A Vectorization Method for Undirected Labeled Graph" in IEEE Access, vol. 8, pp. 175360-175369, 2020