Group Decision Diagrams (GDDs): A Data Structure for Mathematical Groups

Akriti Dhasmana
Prof. Matt Anderson, Computer Science

ABSTRACT

Mathematical groups are used in physics, chemistry, statistics and cryptology. In this project, we create a concise data structure for groups that improves the speed and ease in which binary operations, like taking union and intersection, can be carried out. For the purpose of the project we will only consider certain finite abelian groups. We also conducted time analysis for each set operation.

BACKGROUND

Permutation Decision Diagrams (πDDs) is a data structure that stores a set of permutations [4]. Every permutation can be decomposed into a sequence of transposition. Every node in the data structure represents a transposition.

Identity Permutation
1, 2, 3, 4, 5
Current Permutation
4, 3, 1, 5, 2
Index mapping: 1→3,
2→5,
3→2,
4→1,
5→4
Decomposition:
(3, 5, 2, 1, 4) =
(2, 1)(3, 2)(4, 3)(5, 4)

Figure 1. Permutation in transpositions

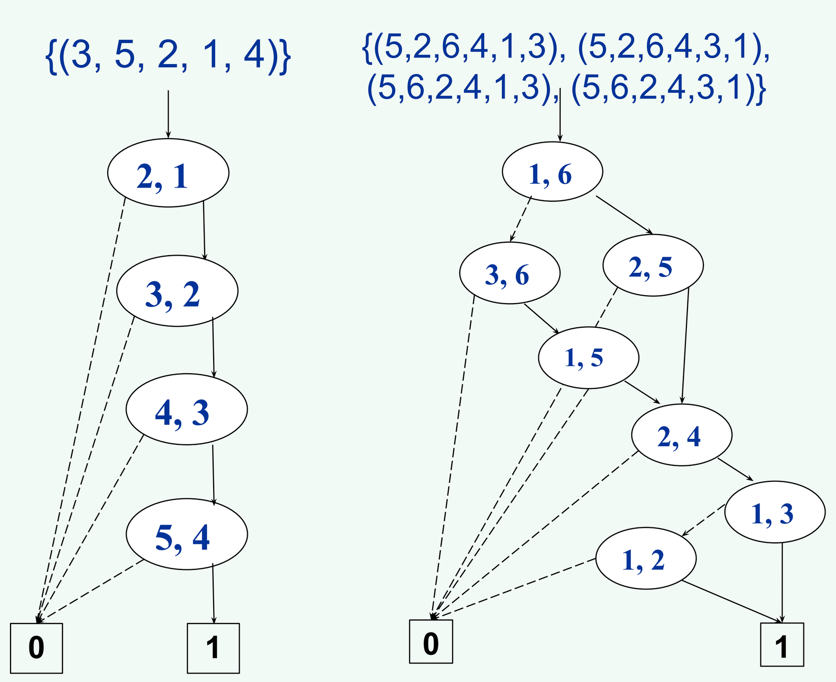

Figure 2. πDD of a single permutation

Figure 3. Basic structure of a πDD.

The Fundamental Theorem of Finitely Generated Abelian Groups (FTFGAG) states that any finite abelian group can be expressed as a direct product of cyclic groups [3]. Just like permutations can be decomposed into a series of transpositions, a finite abelian group can be represented using the single generators for each cyclic group in its decomposition.

x_5	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

$2^0 = (1)\text{mod}5 = 1$
$2^1 = (2)\text{mod}5 = 2$
$2^2 = (4)\text{mod}5 = 4$
$2^3 = (8)\text{mod}5 = 3$

Figure 4. Cayley Table for (axb)mod5 with generator <2>.

GOAL

Modify πDDs into a data structure storing the elements of a finitely abelian group in a concise way using the FTFGAG and the generators for cyclic groups. We call this data structure Group Decision Diagrams (GDD).

DESIGN

We use FTFGAG to decompose a finite abelian group into a unique series of cyclic group. The group G can be written as:

$$G = g^0 G_0 + g^1 G_1 + g^2 G_2 + g^3 G_3 + \ldots + g^n G_n$$

In the GDD, every node represents a generator. The GDD for the cyclic group from figure 4 is given in figure 5. Each branch from the generator node corresponds to increasing exponents from left to right. This means that i^{th} child of the node with generator g in figure 6 has g^i applied to it. In figure 5, this means that each child node is $<2^i>$.I.

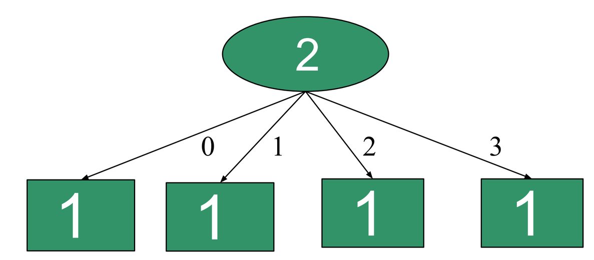

Figure 5. GDD for cyclic group

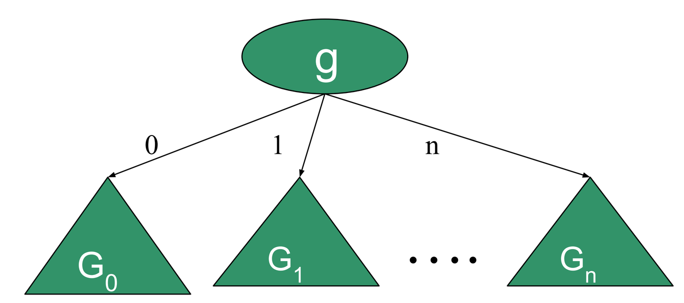

Figure 6. GDD for abelian group

INVARIANT OF DATA STRUCTURE

1. Each generator has a unique id and associated order that is used to arrange the nodes relatively in the structure.
2. Children of a node must have an id less than the node itself.
3. Each node in the tree must be unique, that is, two nodes cannot have the same order, id and children.

Union

$$G \cup H = \bigcup_{0}^{n} g^i (G_i \cup H_i)$$

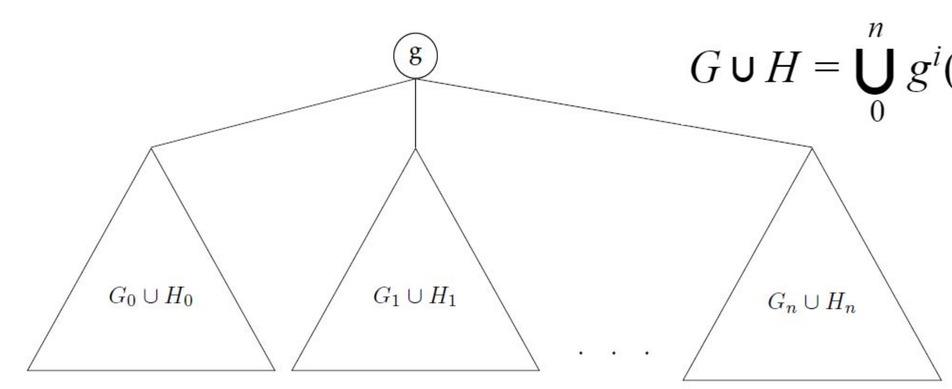

Figure 7. G∪H when g=h,

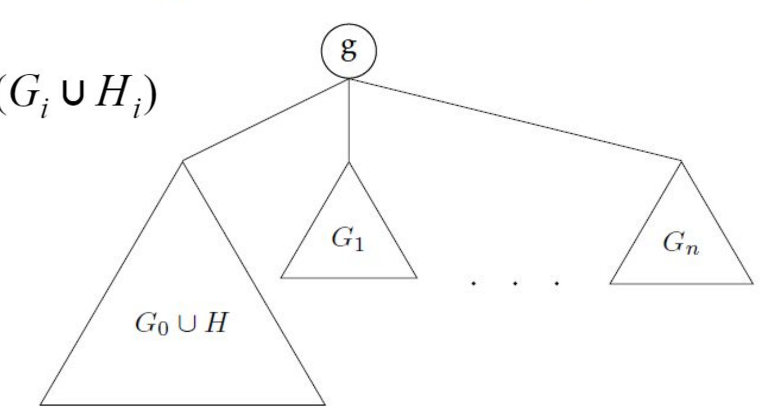

Figure 8. G∪H when g<h

Intersection

$$G \cap H = \bigcup_{0}^{n} g^i (G_i \cap H_i)$$

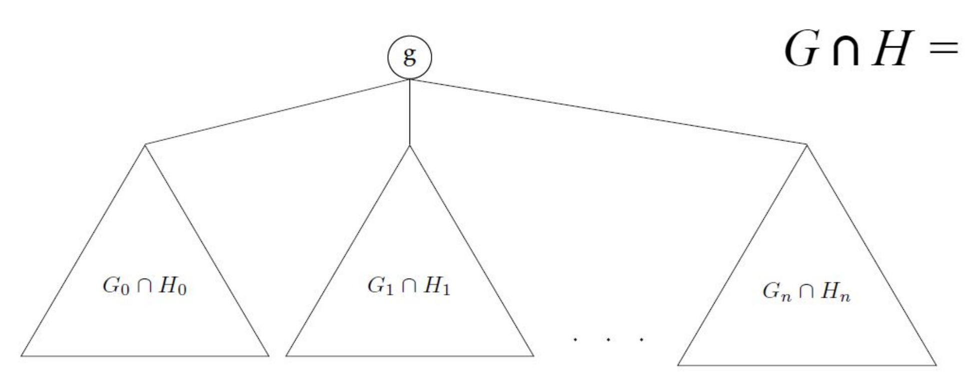

Figure 9. G∩H when g=h

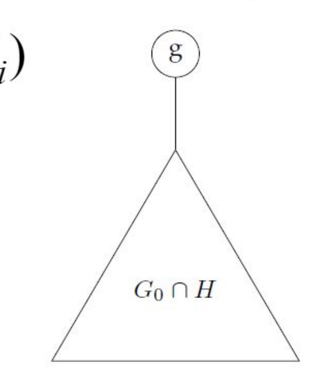

Figure 10. G∩H when g<h

Difference

$$G/H = \bigcup_{0}^{n} g^i (G_i / H_i)$$

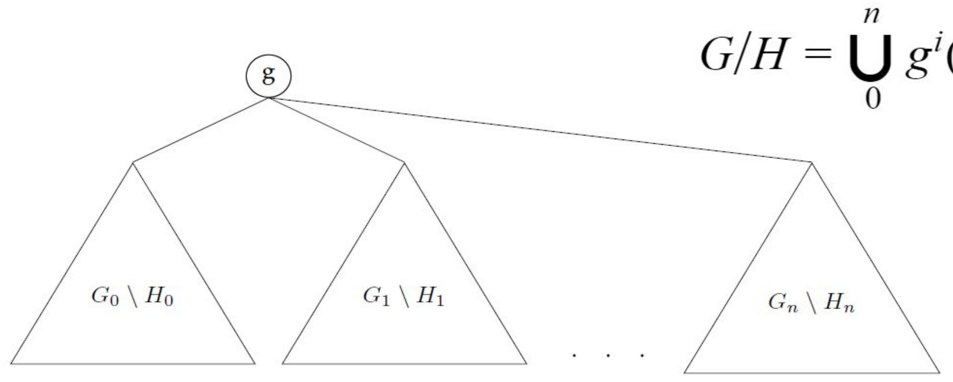

Figure 11. G\H when g=h

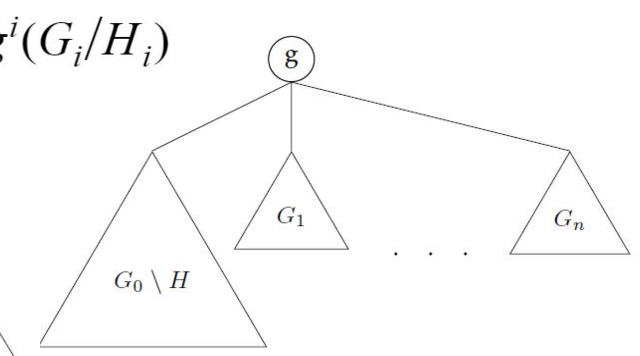

Figure 12. G\H when g>h

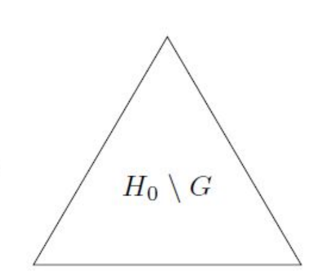

Figure 13. G\H when g>h

TIME ANALYSIS

Thus, we were successfully able to implement the data structure in Python. We also implemented additional complex operations like multiply and apply for the groups. The apply operation is analogous to the dot product while multiplying is vector product. Based on our time analysis experiment shown in the table below, we can see that the data structure is efficient in carrying out union and intersection operations. Comparatively, multiply operation is slower and can be improved with future revisions.

GDD Depth	Union	Intersection	Multiply	Apply
2	$4.19\text{x}10^4$	$2.81\text{x}10^4$	$7.72\text{x}10^5$	$1.35\text{x}10^5$
3	$2.61\text{x}10^5$	$5.88\text{x}10^4$	$5.21\text{x}10^7$	$1.61\text{x}10^6$
4	$1.78\text{x}10^6$	$9.02\text{x}10^4$	$3.96\text{x}10^9$	$5.33\text{x}10^6$
5	$1.93\text{x}10^6$	$8.15\text{x}10^4$	$3.71\text{x}10^9$	$2.50\text{x}10^6$

Figure 9. Average time taken (ns) over 100 runs for worst case for operations on GDD with different depths.

FUTURE WORK

During our initial research on group theory we had come across a paper by Cannon on the Todd-Coxeter and Schreier-Sims algorithms [1, 2]. More in-depth analysis of these algorithms may provide insight into obtaining decompositions for the groups that our data structure is currently able to take as input.

Acknowledgements

I would like to thank Union College Davenport Research Fellowship and Scholars Program for supporting me.

References

[1] John Cannon. "Implementation and Analysis of the Todd-Coxeter Algorithm". In: *Mathematics of Computation* 27.123 (1973), pp. 463–463. DOI: doi:10.1090/s0025-5718-1973-0335610-5.
[2] Schreier. "Schreier-Sims Method". In: *Algorithms for Permutation Groups Lecture Notes in Computer Science*
[3] Allan Clark. *Elements of Abstract Algebra*. Dover Publications, Oct. 1984. ISBN: 0486647250.
[4] Shin-Ichi Minato. "πDD: A New Decision Diagram for Efficient Problem Solving in Permutation Space". In: *Theory and Applications of Satisfiability Testing - SAT 2011 Lecture Notes in Computer Science (2011)*, pp. 90–104. ISSN: 0303-2647. DOI: 10.1007/978-3-642-21581-09.