



**CALIFORNIA STATE SCIENCE FAIR  
2008 PROJECT SUMMARY**

<b>Name(s)</b> <b>James J. Thomas</b>	<b>Project Number</b> <b>J1321</b>
<b>Project Title</b> <b>An Algorithm to Minimize Memory Usage in Graph-based Applications</b>	
<p style="text-align: center;"><b>Abstract</b></p> <p><b>Objectives/Goals</b> The objective of this project was to invent an efficient algorithm to minimize the memory usage of software applications that use graph data structures. Based on analysis of possible approaches to solving the problem, it was hypothesized that an optimum algorithm could run in approximately <math>O(S^2 \cdot E)</math> time, <math>S</math> and <math>E</math> being the number of sources and number of edges in an input graph, respectively. A heuristic algorithm was considered an acceptable solution if its runtime benefits over optimum algorithms outweighed its lack of consistent optimality.</p> <p><b>Methods/Materials</b> To minimize its memory usage, a graph-based application must dynamically free memory used by graph edges and vertices that are no longer reachable by any sources yet to be traversed. In this context, the weight of a graph edge is assumed to indicate the amount of memory being used by data stored as part of the edge. A graph's average weight is defined as the average of <math>S</math> graph weights, the total weight of the graph being taken after each source traversal. An algorithm that optimally frees memory used by graph edges is defined as one that produces for any input graph a source ordering that enables minimization of the graph's average weight. Use of the average weight system allows early elimination of significant graph weight to be valued highly, an important factor in producing optimum results. A brute force algorithm that checks all possible orderings of a graph's sources was developed to verify algorithm optimality.</p> <p><b>Results</b> The proposed algorithm uses recursion to find the optimum source ordering of an input graph; it divides the graph into smaller and smaller subgraphs until it reaches a base case and then works its way back up, eventually returning to and solving for the original input graph. The algorithm is a heuristic that produced optimum results for most tested graphs; it has sizable runtime benefits over considered optimum algorithms. Analysis shows that the algorithm runs in approximately <math>O(S^4 \cdot V \cdot \log E)</math> time.</p> <p><b>Conclusions/Discussion</b> The major contributions of this work to the field of graph theory are the definition and implementation of the concepts of average graph weight and optimum source ordering for dynamic graph weight minimization. In addition to its applications in improving software efficiency, the proposed algorithm has numerous other practical uses. Most notable among them is its ability to minimize energy usage in factories.</p>	
<b>Summary Statement</b> The purpose of this project was to develop an algorithm that would enable minimization of an input graph's weight by producing an optimum source ordering by which to traverse its sources.	
<b>Help Received</b> Father suggested project idea and reviewed board material	