Brook+ SAMPLE

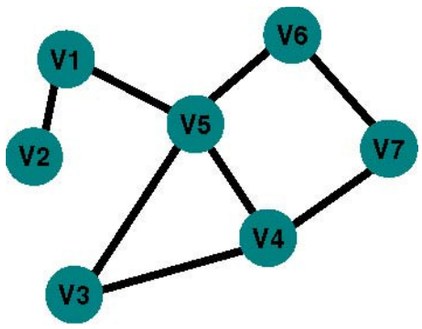
Floyd-Warshall Algorithm

1 Introduction

A graph is an essential tool to model and understand both natural and man-made phenomena. Graphs are frequently encountered when solving problems such as Internet routing, supply-chain networks, oil pipelining, electrical grids, VLSI fabrication, and social networks. The unifying motivation for all such problems is the minimization of energy spent in moving material or information from source to destination through an intervening network of entities. A commonly occurring problem in graph theory is to find the *shortest path* between two entities. Before elaborating the concept of a shortest path, we recall the formal definition of a graph as understood in mathematics and computer science:

> A *graph* is a collection of vertices and edges that maybe denoted as ***G(V,E)***, where ***V* = {V1, V2, …, Vn}** is a set of vertices or nodes, and ***E* = {e1, e2, …, em}** is a set of edges, where each edge is an unordered pair (ordered pair for directed graphs) of vertices. An edge e_i connecting the vertices V_p and V_q is denoted (V_p, V_q).

Figure 1 shows a diagram of a graph with seven vertices and eight edges.



G(V,E)

V = {V1,V2,V3,V4,V5,V6,V7}

E = { (V1,V2), (V1,V5), (V5,V3), (V5,V4), (V5,V6), (V6,V7), (V7,V4), (V4,V3) }

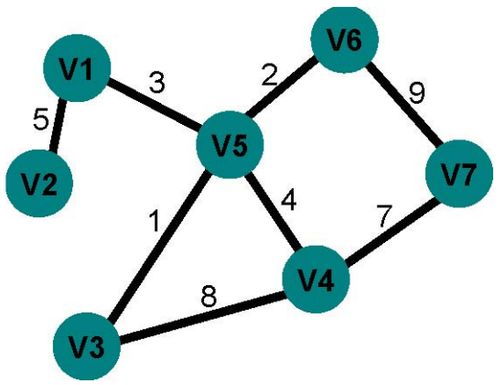
Figure 1 Graph G(V,E): V has Seven Vertices, and E has Eight Edges

2 Shortest Path

A *weighted graph* is a graph where every edge has a weight attached to it. The weight generally signifies some “cost” or “distance” associated with that edge. The *shortest path problem* tries to find the path between a given pair of vertices that minimizes the sum of the weights of edges encountered along that path. Shortest path problems can be formulated for a single source or a single destination. A single-source shortest path problem finds the shortest path from a given source vertex to all other vertices; a single-destination problem finds shortest paths from all vertices to a specified one. Alternatively, we can find shortest paths from every vertex in a graph to every other vertex. This is termed as the *all pair shortest path* problem and is the premise of the Floyd-Warshall algorithm.

3 Adjacency Matrix

The *adjacency matrix* of a graph is a square matrix of dimensions ***n x n***, where ***n*** is the number of nodes in the graph. If **P(nxn)** is the adjacency matrix for graph **G(V,E)**, then **P(i,j)** indicates the weight of the edge from $\mathbf{V_i}$ to $\mathbf{V_j}$. **Figure 2** shows the graph in **Figure 1** with each edge labeled by a weight and the corresponding adjacency matrix.



	V1	V2	V3	V4	V5	V6	V7
V1	0	5	∞	∞	3	∞	∞
V2	5	0	∞	∞	∞	∞	∞
V3	∞	∞	0	8	1	∞	∞
V4	∞	∞	8	0	4	∞	7
V5	3	∞	1	4	0	2	∞
V6	∞	∞	∞	∞	2	0	9
V7	∞	∞	∞	7	∞	9	0

Figure 2 **Graph G(V,E) with Weights Attached to Each Edge (Left) and the Corresponding Adjacency Matrix (Right)**

Note that the weight between a pair of nodes is infinity (∞) if there is no edge between them. Also note that zeroes appear along the diagonal. The weight of an edge from a node to itself is zero because there is no cost in the traversal.

4 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm computes the shortest path between each pair of nodes in a graph. It is a dynamic programming approach that iteratively refines the adjacency matrix of the graph in question until each entry in the matrix reflects the shortest path between the corresponding

nodes. The main idea of the algorithm is as follows: Given the shortest path between node V_i and V_j using $\mathbf{V_1 \ldots V_k}$ as intermediate nodes, find out the shortest path between $\mathbf{V_i}$ and $\mathbf{V_j}$ using $\mathbf{V_1 \ldots V_{k+1}}$ as intermediate nodes. This idea can be recursively formulated as:

ShortestPath(i, j, k) = min(ShortestPath(i, j, k-1) , ShortestPath(i, k, k-1) + ShortestPath(k, j, k-1)) ShortestPath(i, j, 0) = EdgeCost(i, j) [1]

5 Floyd-Warshall on Brook+

For the Brook+ implementation of the Floyd-Warshall algorithm, we start with a randomly generated adjacency matrix. It is a full matrix in the sense that there is an edge between every pair of nodes, which makes the graph a clique. The weights for self-loops are set to zero, so zeroes appear along the diagonal of the matrix.

The inputs to the Brook+ kernel are the adjacency matrix **P** and the step number ***s***. The Brook+ kernel is invoked *n* times, where *n* is the number of nodes in the graph. At the $\boldsymbol{s}^{th}$ step, the kernel computes two numbers for every pair of nodes in the graph. The "direct distance" between them is determined by looking up the adjacency matrix **P** and the "indirect distance," if one were to use node ***s*** as an intermediate node. During the initial pass, ***s***=1, the direct distance is simply the weight of the edge between $\mathbf{V_i}$ and $\mathbf{V_j}$. The indirect distance using V_1 as an intermediate node is the sum of the entries $\mathbf{P(V_i, V_1)}$ and $\mathbf{P(V_1, V_j)}$. The smaller of the two distances is written back to the output. At the end of pass 1, the adjacency matrix reflects the distances between each pair of nodes in the graph using $\mathbf{V_1}$ as an intermediate node. This matrix is used as a basis for the next pass, where we perform the same computation with respect to $\mathbf{V_2}$. The final matrix reflects the lengths of the shortest paths between each pair of nodes in the graph.

6 Performance

The performance of Brook+ Floyd-Warshall was tested on a system with the AMD Athlon™ 64 X2 Dual Core Processor CPU and an ATI Radeon HD 4800 Series GPU. Figure 3 shows a comparison of the time taken to compute shortests paths for all pairs of nodes on the GPU versus the CPU for varying graph sizes.

1. Wikipedia contributors, "Floyd–Warshall algorithm," *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm (accessed August 04, 2008).

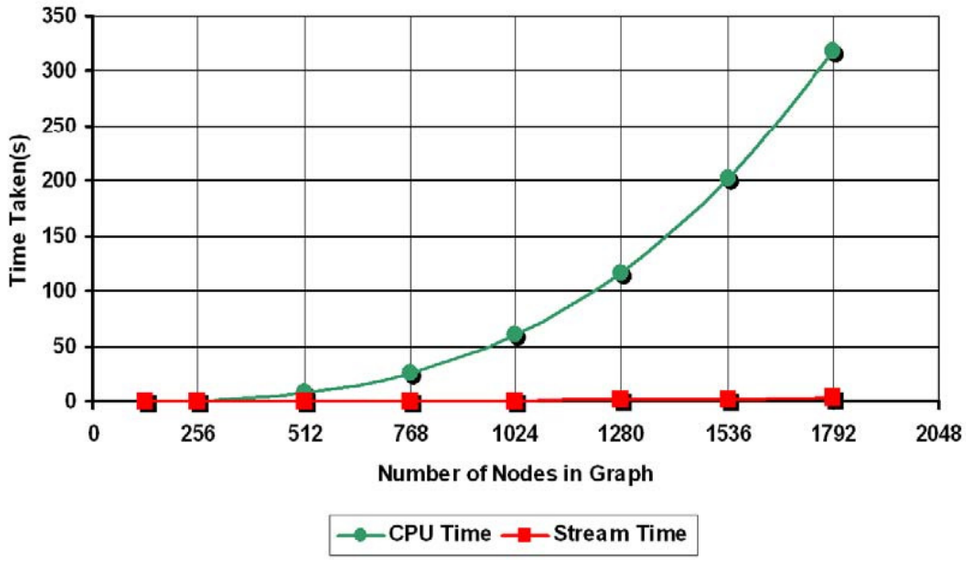


Figure 3 Performance of Floyd-Warshall: ATI Stream Processor vs CPU

Figure 4 shows the speedup achieved over the reference CPU implementation. The Radeon 4800 Series can achieve speedups of over 100 for this sample implementation.

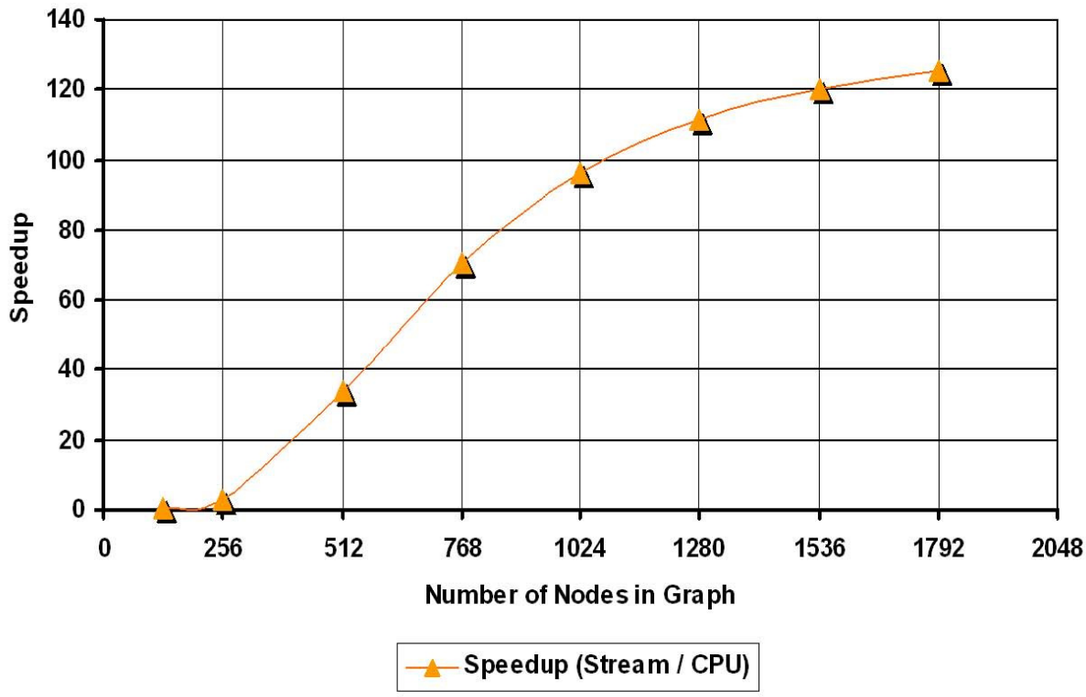


Figure 4 Floyd-Warshall Speedup Comparison: ATI Stream Processor vs CPU

7 Conclusion

Graph algorithms often can be formulated as matrices; therefore, they fit well into the realm of accelerated computing. We presented an example of one such algorithm to find the lengths of the shortest paths between each pair of nodes in a graph. The Brook+ SIMD implementation takes approximately four seconds to converge for a graph with 2048 nodes. This outperforms the reference CPU implementation by over a factor of 100. Note that the reference CPU implementation may not be the best or most optimized one, but the performance gain by orders of magnitude establishes the incentive to parallelize.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For Stream Computing:
URL: www.amd.com/stream
Questions: streamcomputing@amd.com
Developing: streamdeveloper@amd.com
Forum: www.amd.com/streamdevforum