Algorithms Seminar: March 4, 2005

Lecturer: Steve Skiena

Scribe: Janet Braunstein

Introduction

This week's problem comes from a graph drawing package from *Mathematica*. The notion of the diameter of a graph (the largest distance between any two vertices in a graph, or the largest value in the all-pairs shortest path matrix) is key. Graph drawing utilizes diameter in order to make figures look neat. By fixing the center of the diameter of a graph as the center of the drawn graph, a balanced figure can be achieved. Another motivation for finding the diameter of a graph comes from communications networks. Diameter can be found exactly using the Floyd-Warshall algorithm or Dijkstra's algorithm, but both of these methods run slowly. The *Mathematica* package cuts down the running time by utilizing an algorithm that finds a "pseudo-diameter." The purpose of today's meeting was to attempt to bound both the quality and the running time of this algorithm.

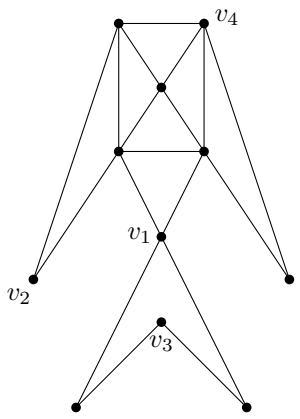
Finding Diameter Exactly: We consider the example of an undirected, unweighted graph $G = (V, E)$, with $|V| = n$ and $|E| = m$. For such a graph, we have two logical methods for finding the diameter exactly. The Floyd-Warshall all-pairs shortest path algorithm finds the exact diameter of G in $O(n^3)$. If G is sparse, n calls to Dijkstra's $O(m \log n)$ algorithm allow us to find the diameter in slightly better time, $O(mn \log n)$. However, both of these methods are quadratic or worse in n. This motivates us to look for an algorithm that sacrifices some accuracy for a decrease in running time.

Finding Pseudo-Diameter: The *Mathematica* package proposes the following algorithm for finding the pseudo-diameter of G:

$D_S \leftarrow -1$
select arbitrary vertex v_i
$v_j \leftarrow v_i$
while $d(v_i, v_j) > D_S$ **do**
  $D_S \leftarrow d(v_i, v_j)$
  $v_i \leftarrow v_j$
  perform BFS with v_i as root to find farthest vertex from v_i (breaking ties arbitrarily)

$v_j \leftarrow$ this farthest vertex
end while
return D_S

Here is an example of the algorithm at work (keeping in mind that this is just one possible sequence of steps for the given graph, since ties are broken arbitrarily):



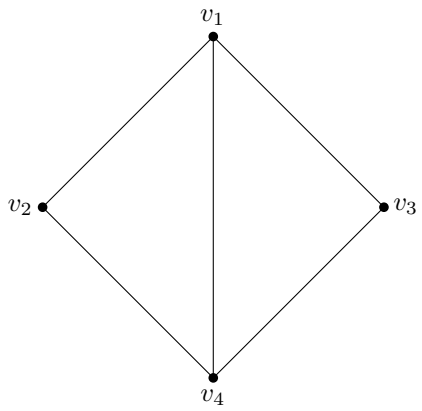
The algorithm starts at v_1. The first iteration takes us to v_2 with $D_S = 2$, the second takes us to v_3 with $D_S = 4$, and the third takes us to v_4 with $D_S = 4$. At this point, the algorithm returns 4, which is the true diameter of this graph. The algorithm does not return the true diameter in all cases for all graphs, however.

Bounding Quality: Clearly, for any graph G, $D_S(G) \leq D(G)$, with strict equality being possible. We will now try to find a tight lower bound.

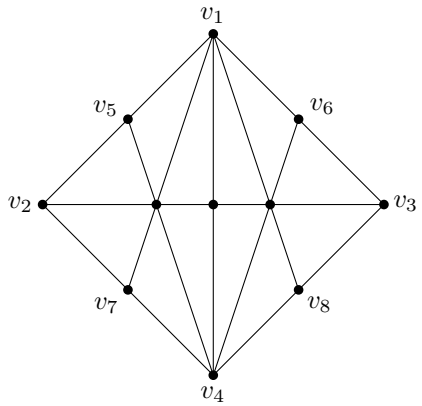
Claim. *$D_S(G)$ can never be less than $\frac{1}{2}D(G)$.*

Proof. Look at two points which serve as the endpoints of the diameter of G, call them v_1 and v_2. Now start algorithm at arbitrary vertex v_3. We claim that either $d(v_3, v_1) \geq \frac{D}{2}$ or $d(v_3, v_2) \geq \frac{D}{2}$. This is clearly true if v_3 is on the diameter. If v_3 is not on the diameter and $d(v_3, v_1) < \frac{D}{2}$ and $d(v_3, v_2) < \frac{D}{2}$, then $d(v_1, v_2) \leq d(v_1, v_3) + d(v_3, v_2) < D$, which gives a contradiction. Therefore, v_3 is at least $\frac{D}{2}$ away from some other vertex, meaning the first iteration of the algorithm will give a pseudo-diameter which is at least half of the true diameter. □

We want to determine if this is a strict lower bound. To that end, we first study the following fan-shaped graph, F_1:



If we start at v_1, all other vertices are distance 1 away. Suppose the algorithm arbitrarily selects v_4 as the farthest away. All vertices are distance 1 away from v_4, as well, so the algorithm will end here and return $D_S = 1$. Clearly, the true diameter is 2. The question, however, is whether the pseudo-diameter is off by a multiplicative 2 or an additive 1. This leads us to examine another graph, F_2:



In this case, if we start at v_1, the farthest vertices are at a distance of 2 away. Suppose the first iteration selects v_4 from among these vertices. The next iteration will find that the farthest vertices from v_4 are also at a distance of 2 away, causing the algorithm to stop and return $D_S = 2$. The actual diameter of F_2 is 4. But now we would like to generalize these findings. We want to find a family of graphs whose actual diameters are D, but whose pseudo-diameters may be found to be $\frac{1}{2}D$ using the *Mathematica* algorithm, for any given $D = 2p$, where $p \in \mathbb{Z}^+$.

Claim. *For any given $D = 2p$, we can construct a graph with actual diameter D for which the algorithm may return a pseudo-diameter of $\frac{D}{2}$.*

Proof. Given p, construct G_p from the graph F_1 as follows. Place $p-1$ vertices along the edge from v_1 to v_2. Place $p-1$ vertices along each of the 4 outer edges of the fan. Draw an edge from each of the $2p-2$ new vertices along the top outer half of the fan to v_4. Draw an edge from each of the $2p-2$ new vertices along the bottom outer half of the fan to v_1. Place a vertex at each of the $2(p-1)^2$ intersection points of the new edges. Now run the algorithm. Suppose we start at v_1. The maximum distance to any other vertex is p. v_4 is at a distance of p away from v_1, so suppose the algorithm selects it next. By symmetry, the algorithm may next select v_1 and terminate, returning a pseudo-diameter of p. The actual diameter is $D = 2p$ because $d(v_2, v_3) = D$. Therefore, the quality of the algorithm can be as bad as $\frac{1}{2}OPT$. □

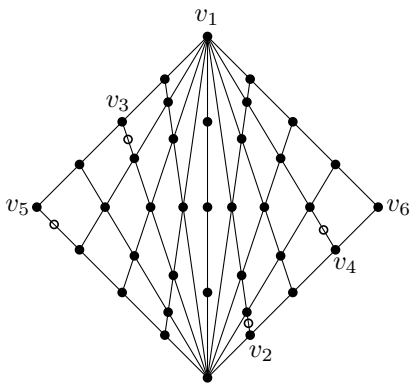
Bounding Time: We would now like to determine some bounds on the running time of the algorithm, in terms of the actual diameter of a graph.

Claim. *There can be no more than $\frac{D}{2} + 2$ iterations.*

Proof. From above, the first iteration of the algorithm always finds $D_S \geq \frac{D}{2}$. Since D_S cannot increase by a fractional amount, and since equality causes the algorithm to terminate, an increase of 1 at each iteration will yield the longest possible running time, $\frac{D}{2}+2$ iterations. □

Conjecture. *The running time of the algorithm can be linear in D.*

We would like to find a family of graphs for which the algorithm may complete $O(D)$ iterations. We find that this is the case with the single graph F_2. Suppose the algorithm starts at v_1. Among all choices at distance 2 away, suppose it selects v_7. Suppose it then selects v_3 at distance 3 away. It then must select v_2 at distance 4 away. Finally, it must select v_3 at distance 4 away and terminate. This leads us to look for a new fan-shaped graph with larger diameter for which the algorithm finds an initial pseudo-diameter of approximately half the true diameter and then adds 1 at each iteration. This leads us to the following graph, F_3:



It is just the graph G_4 with additional strategically placed vertices (shown as open circles) that make it possible for the pseudo-diameter to increase by 1 at each iteration. If the algorithm starts with v_1, a possible sequence of steps is to go through the vertices in numerical order. In this case, the distances found are 5, then 6, then 7, then 8, then 8, giving an apparent linear running time in the actual diameter (which is 8). We can again generalize this for any given p. We use the same family of graphs as we did above, based on F_1 and modify them by adding p strategically placed vertices. Look at the p paths from v_1 to the bottom right outer quarter of the fan (including the top right outer quarter of the fan as a path, but not including path from v_1 to v_4). Start at the leftmost of these paths and move right, placing a vertex on the bottom edge of every other path. Now look at the p paths from the top left outer quarter of the fan to v_4 (including the bottom left outer quarter of the fan as a path, but not including the path from v_1 to v_4). Start at the second-rightmost of these paths and move left, placing a vertex on the top edge of every other path. Label these new vertices $u_1 \ldots u_p$, in increasing order with respect to horizontal distance from the (vertical) path from v_1 to v_4. Now run the algorithm. Suppose we start at v_1. The maximum distance to any other vertex is $p+1$. A possible sequence of events is for the algorithm to select vertices in the following pattern: v_1, the vertex below u_1, the vertex above u_2 (via v_4), the vertex below u_3 (via v_1), the vertex above u_4 (via v_4) ... until either v_2 or v_3 is reached. The algorithm will then select v_3 or v_2, respectively, and terminate, returning the actual diameter $D = 2p$. The total number of iterations is $p + 1$, or $\frac{D}{2} + 1$. Therefore, the running time of the algorithm can be as bad as linear in D, verifying our conjecture.

In conclusion, we have bounded both the quality and running time of the *Mathematica* algorithm. We have found that, for general graphs, it may return a pseudo-diameter which is as bad as half the actual diameter, and it may take linear time (in D) to find the pseudo-

diameter.

Note: The Mathematica algorithm does work optimally for trees in all cases. Furthermore, it finishes in 2 or 3 iterations (2 if the start vertex is an endpoint of the diameter, 3 if not). If the start vertex is an endpoint of the diameter, the algorithm will find the other endpoint on the first iteration, then find the first endpoint on the second iteration and finish. If the start vertex is not an endpoint of the diameter, the algorithm will select one of the endpoints on the first iteration, the second endpoint on the second iteration, and the first endpoint on the third iteration and finish.