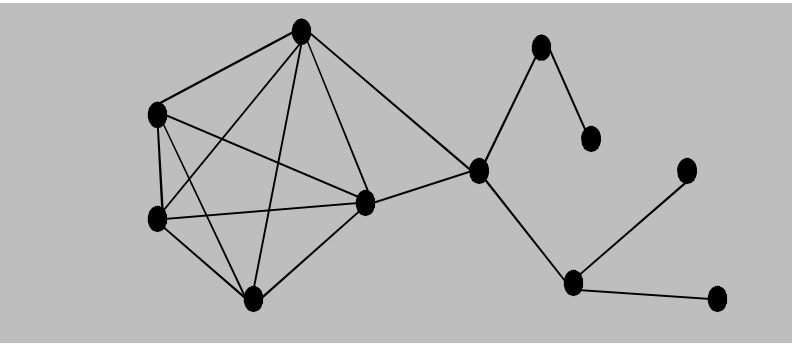
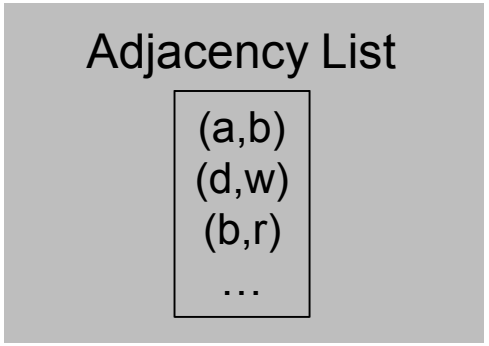
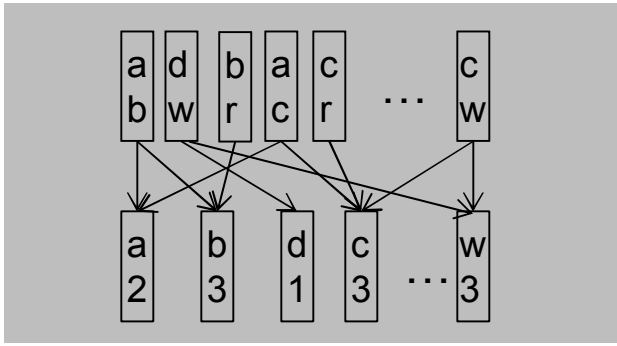
SAND2012-6004P

Exceptional service in the national interest

Sandia National Laboratories







Parallelized Graph Triangle Sampling

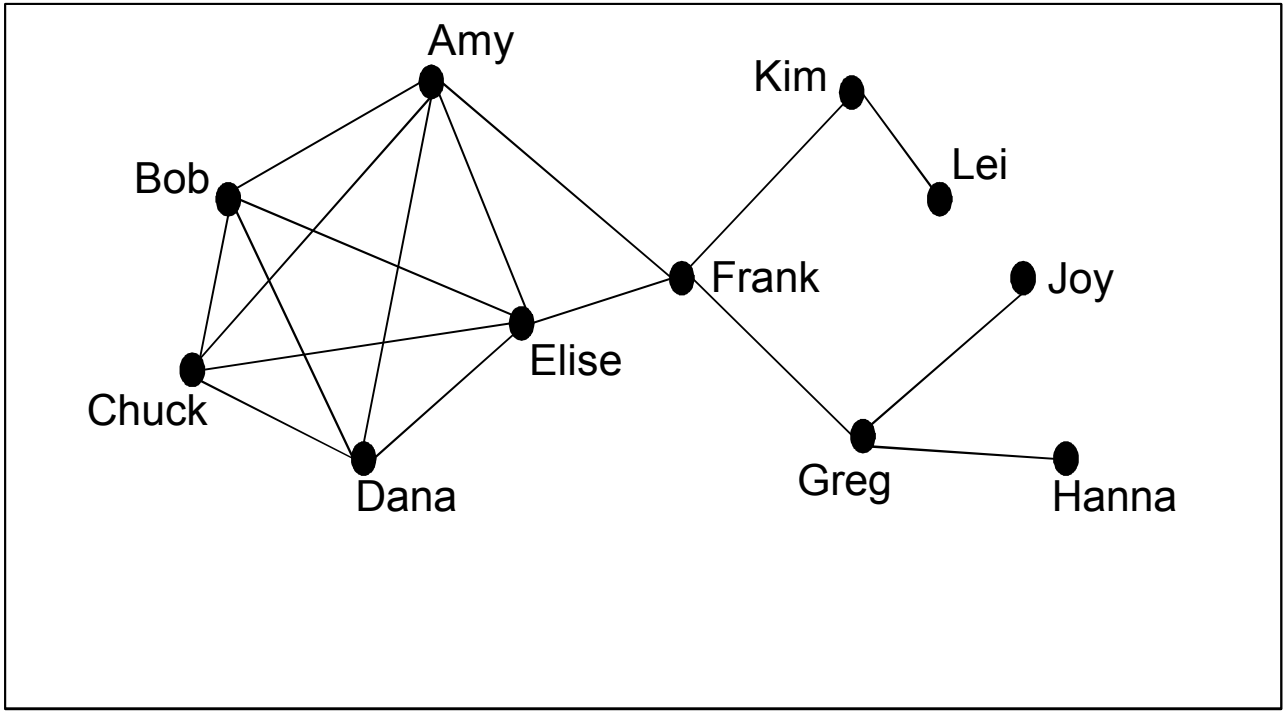
Christine Task, Summer Intern 8966

Mentors: Tammy Kolda, Ali Pinar

U.S. DEPARTMENT OF ENERGY | NNSA

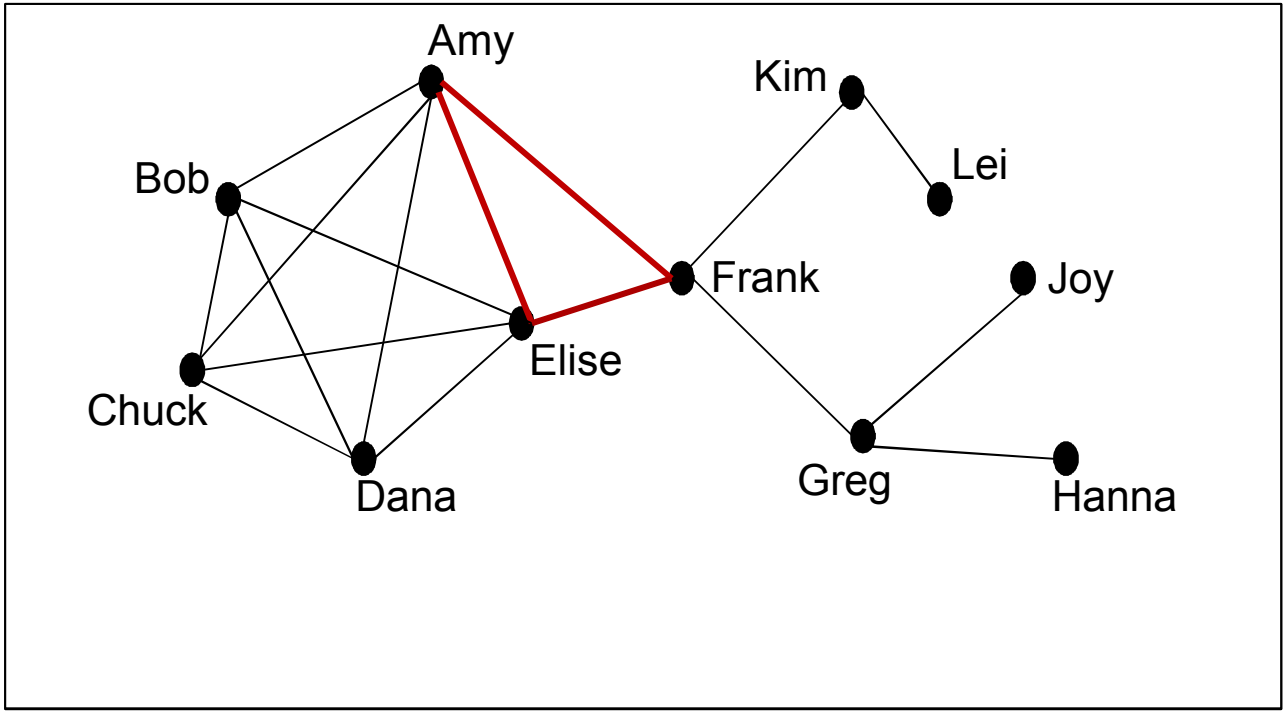
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Why Count Triangles in Social Networks?



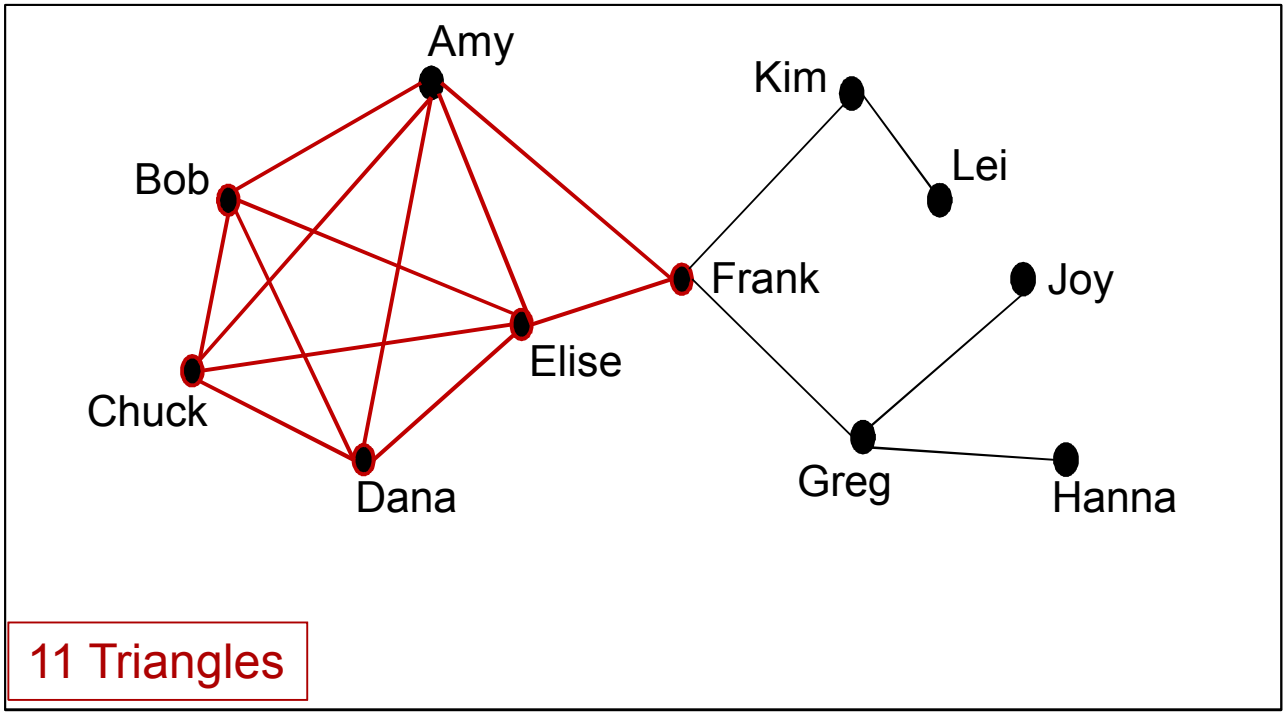
Social Cohesion: Are my friends friends with each other?

Why Count Triangles in Social Networks?

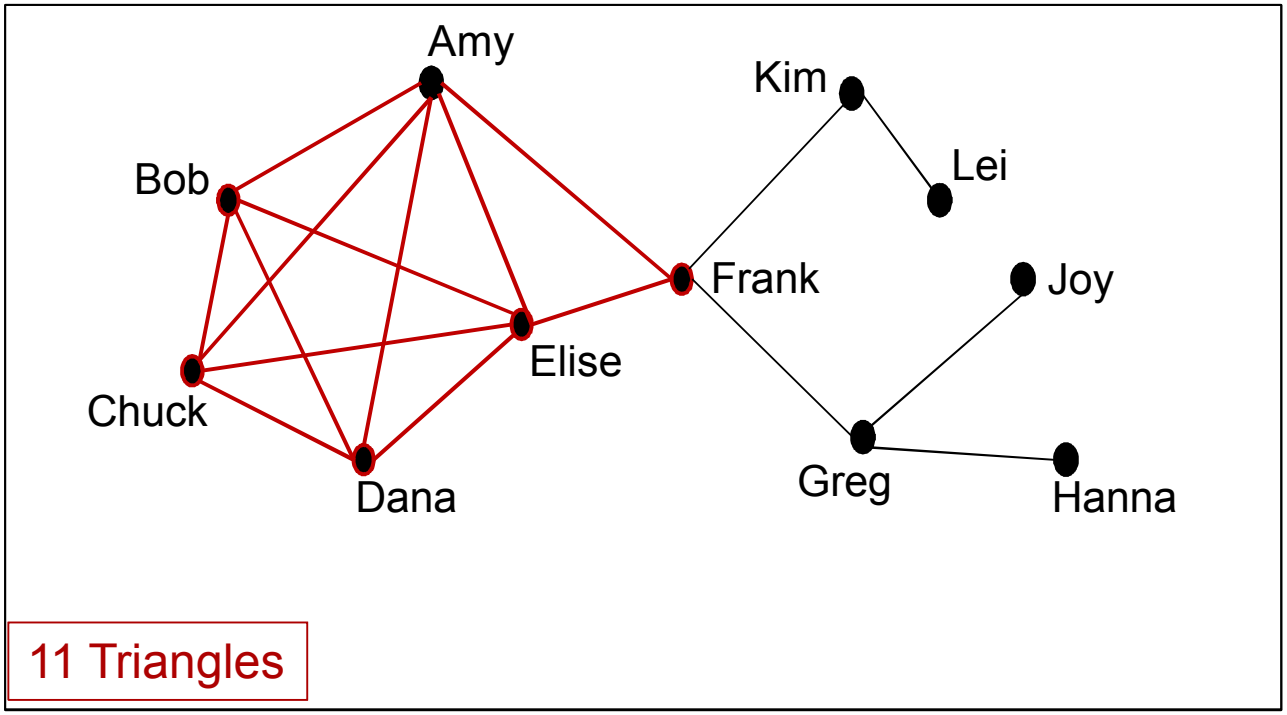


Social Cohesion: Are my friends friends with each other?

Why Count Triangles in Social Networks?



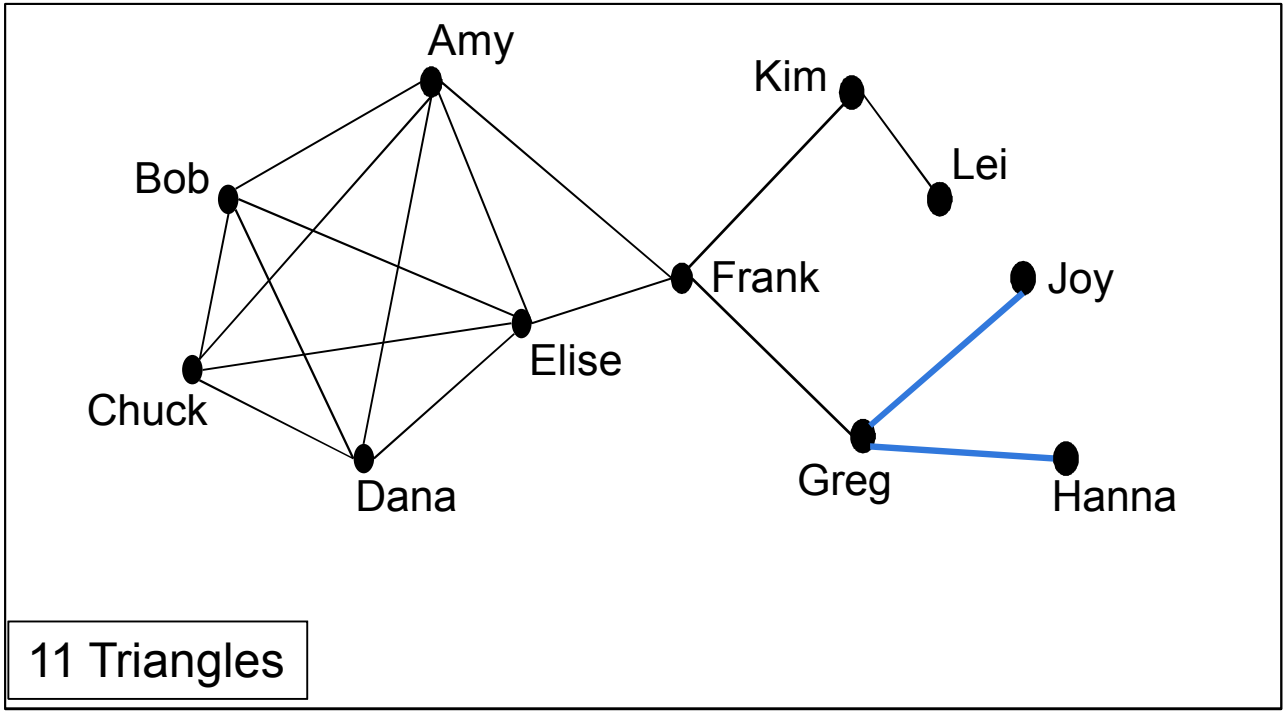
Social Cohesion: Are my friends friends with each other?



Social Cohesion: Are my friends friends with each other?

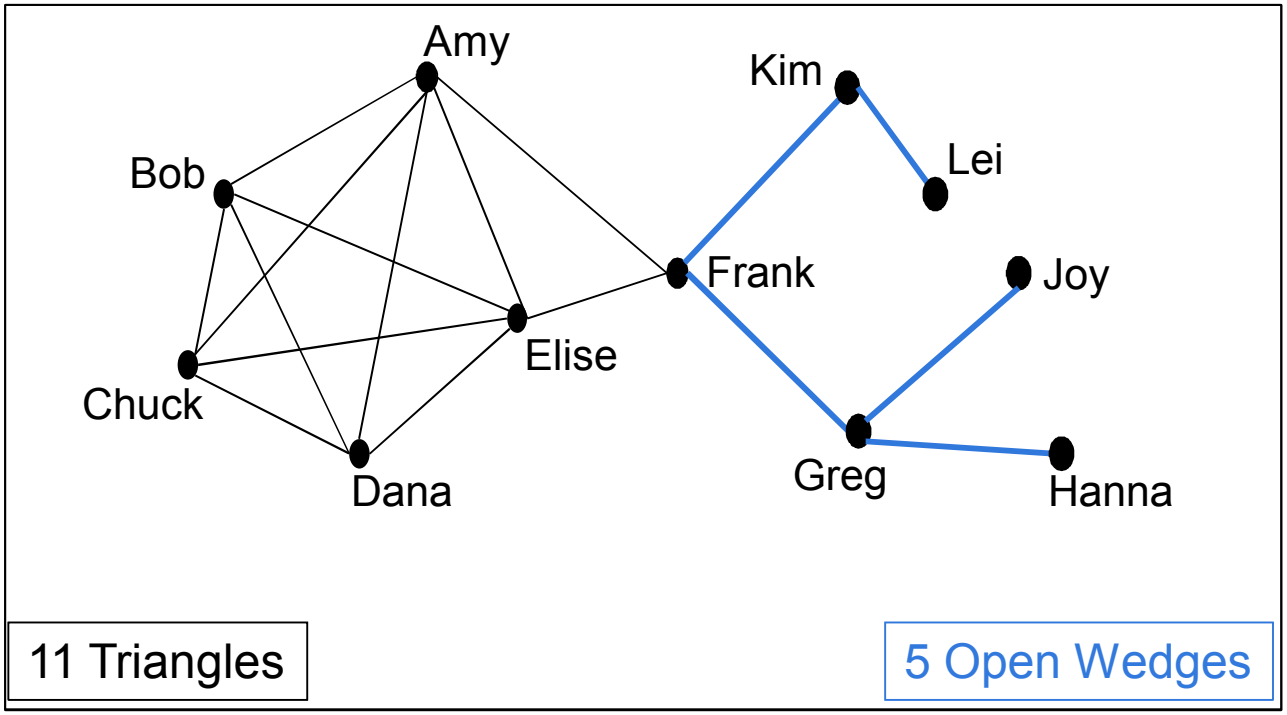
- Graph measure which indicates meaningful properties of social community represented.
- Used when comparing and modeling social networks.

Why Count Triangles in Social Networks?



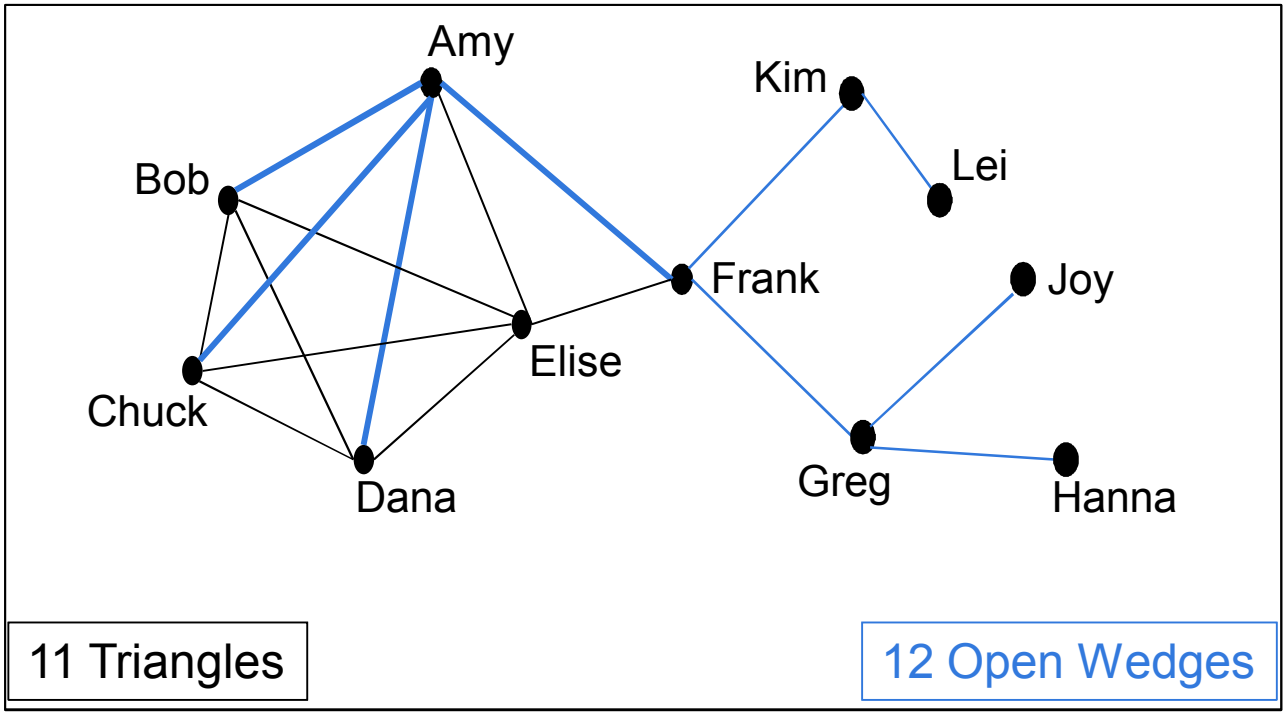
Social Cohesion: Are my friends friends with each other?

Why Count Triangles in Social Networks?



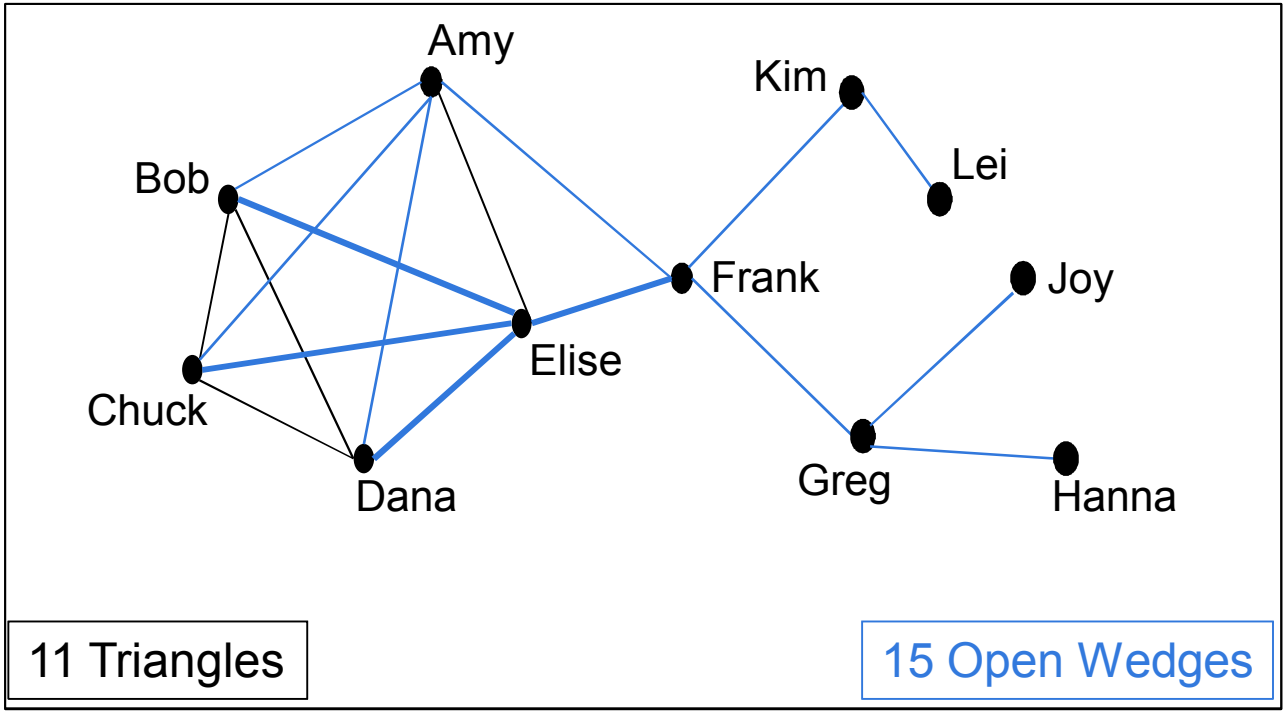
Social Cohesion: Are my friends friends with each other?

Why Count Triangles in Social Networks?



Social Cohesion: Are my friends friends with each other?

Why Count Triangles in Social Networks?

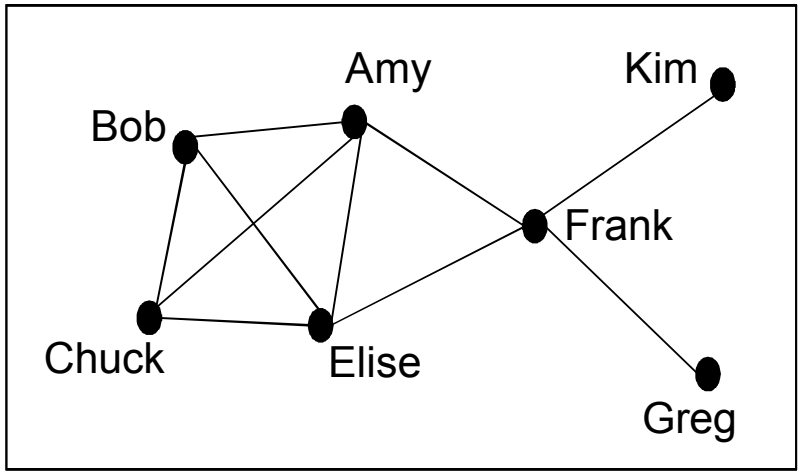


Social Cohesion: Are my friends friends with each other?

How a Computer Sees a Graph:

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)



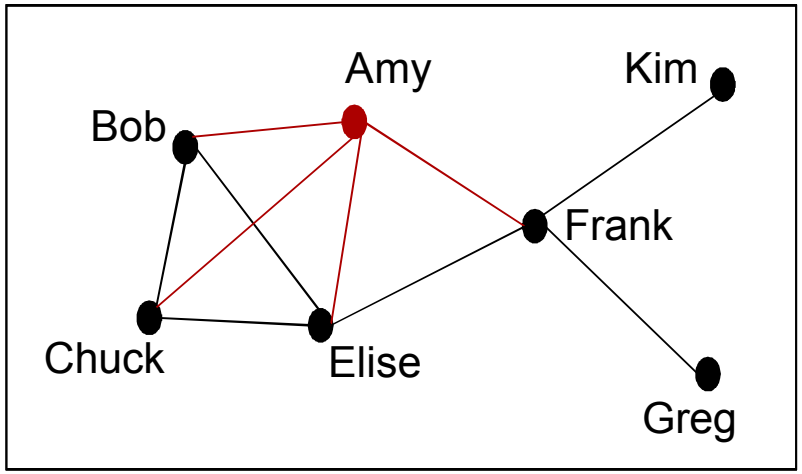
How a Computer Sees a Graph:

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)

Node Degrees

(A,4)
(B,3)
(C,3)
(E,4)
(F,4)
(K,1)
(G,1)



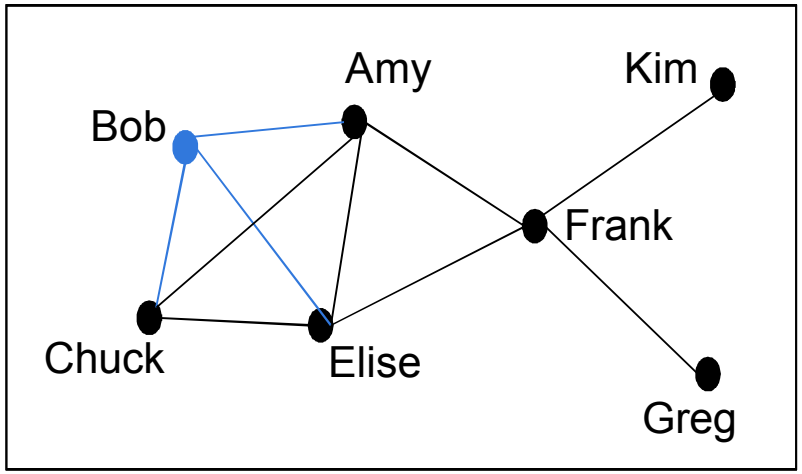
How a Computer Sees a Graph:

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)

Node Degrees

(A,4)
(B,3)
(C,3)
(E,4)
(F,4)
(K,1)
(G,1)



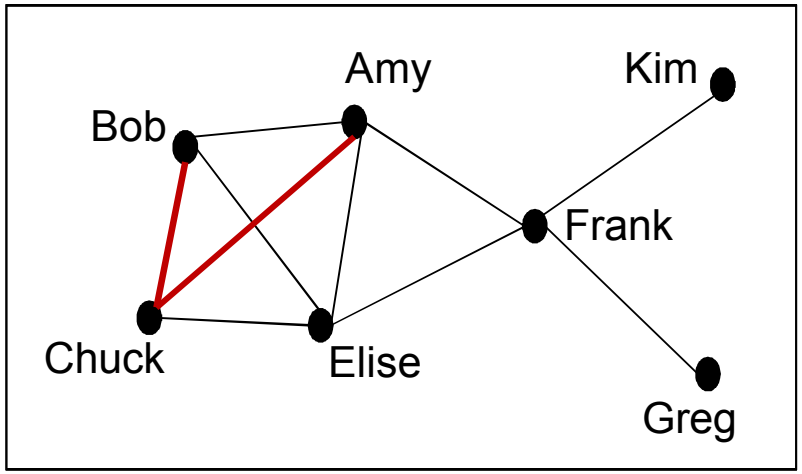
How a Computer Sees a Graph:

Sandia National Laboratories

Adjacency List
(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)

Node Degrees
(A,4)
(B,3)
(C,3)
(E,4)
(F,4)
(K,1)
(G,1)

Node Wedges
(A,6)
(B,3)
(C,3)
(E,6)
(F,6)
(K,0)
(G,0)



Total number of wedges at a node with *d* friends is *(d choose 2) = d(d-1)/2*

How a Computer Sees a Graph:

Adjacency List

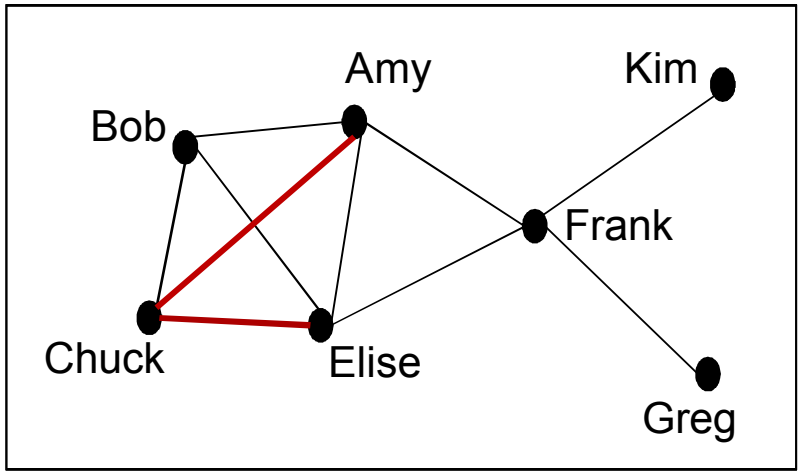
(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)

Node Degrees

(A,4)
(B,3)
(C,3)
(E,4)
(F,4)
(K,1)
(G,1)

Node Wedges

(A,6)
(B,3)
(C,3)
(E,6)
(F,6)
(K,0)
(G,0)



Total number of wedges at a node with *d* friends is *(d choose 2) = d(d-1)/2*

How a Computer Sees a Graph:

Adjacency List

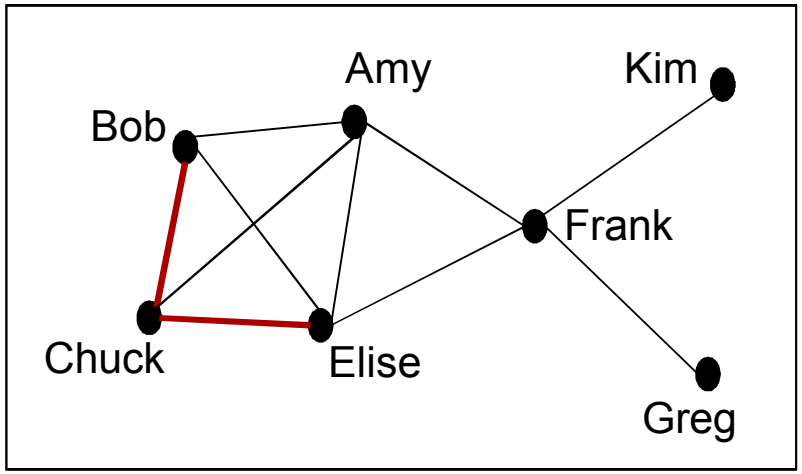
(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
(E,F)
(F,G)
(F,K)

Node Degrees

(A,4)
(B,3)
(C,3)
(E,4)
(F,4)
(K,1)
(G,1)

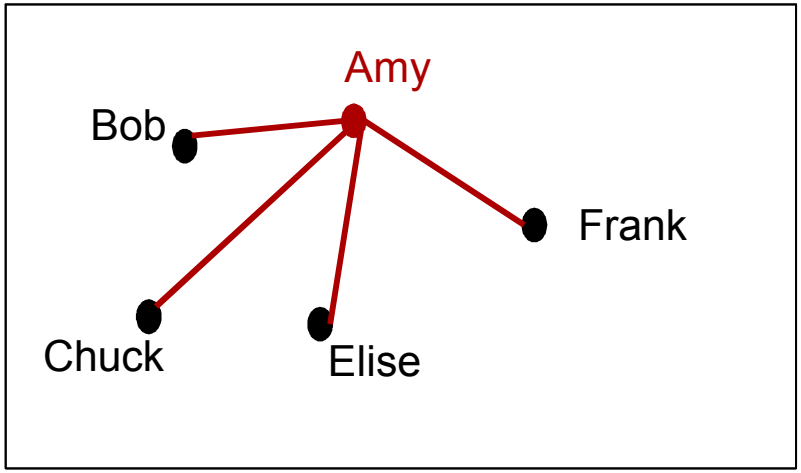
Node Wedges

(A,6)
(B,3)
(C,3)
(E,6)
(F,6)
(K,0)
(G,0)



Total number of wedges at a node with *d* friends is *(d choose 2) = d(d-1)/2*

How a Computer Sees a Graph:



Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
….

Node Degrees

(A,4)
…

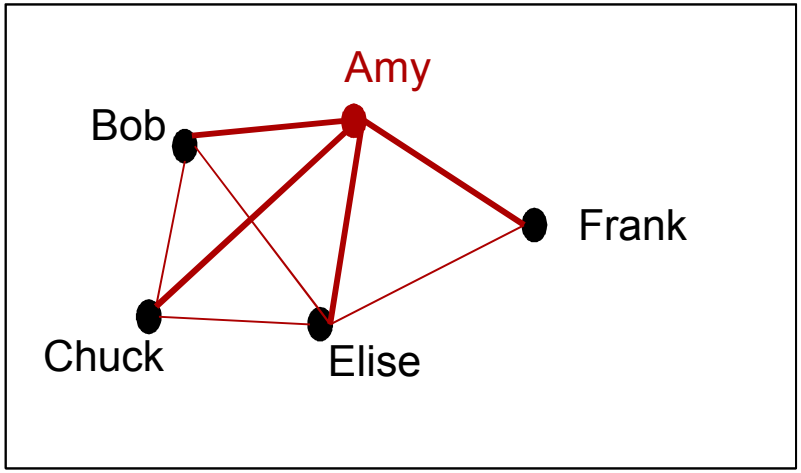
Node Wedges

(A,6)
…

Check for Triangle Closure:

Wedge (A,B)(A,C): Is (B,C) in Adjacency List?
Wedge (A,B)(A,E): Is (B,E) in Adjacency List?
Wedge (A,B)(A,F): Is (B,F) in Adjacency List?
Wedge (A,C)(A,E): Is (C,E) in Adjacency List?
Wedge (A,C)(A,F): Is (C,F) in Adjacency List?
Wedge (A,E)(A,F): Is (E,F) in Adjacency List?

How a Computer Sees a Graph:



Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
….

Node Degrees

(A,4)
…

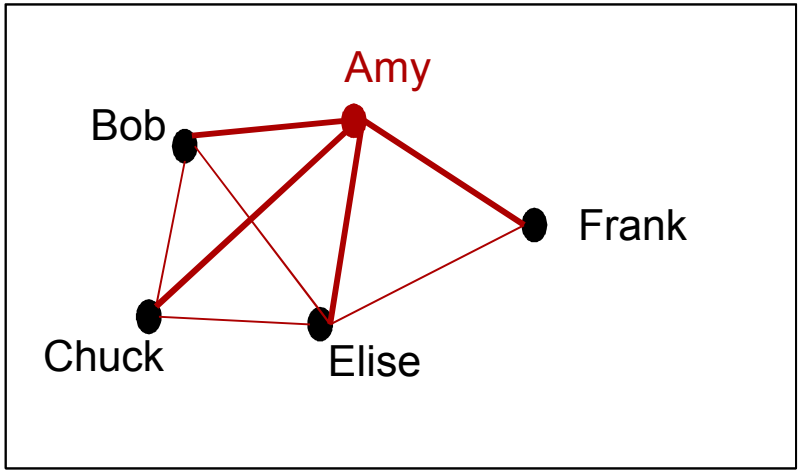
Node Wedges

(A,6)
…

Check for Triangle Closure:

Wedge (A,B)(A,C): Is (B,C) in Adjacency List? Y
Wedge (A,B)(A,E): Is (B,E) in Adjacency List? Y
Wedge (A,B)(A,F): Is (B,F) in Adjacency List? N
Wedge (A,C)(A,E): Is (C,E) in Adjacency List? Y
Wedge (A,C)(A,F): Is (C,F) in Adjacency List? N
Wedge (A,E)(A,F): Is (E,F) in Adjacency List? Y

How a Computer Sees a Graph:



Node A
4 Triangles
2 Open Wedges

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
….

Node Degrees

(A,4)
…

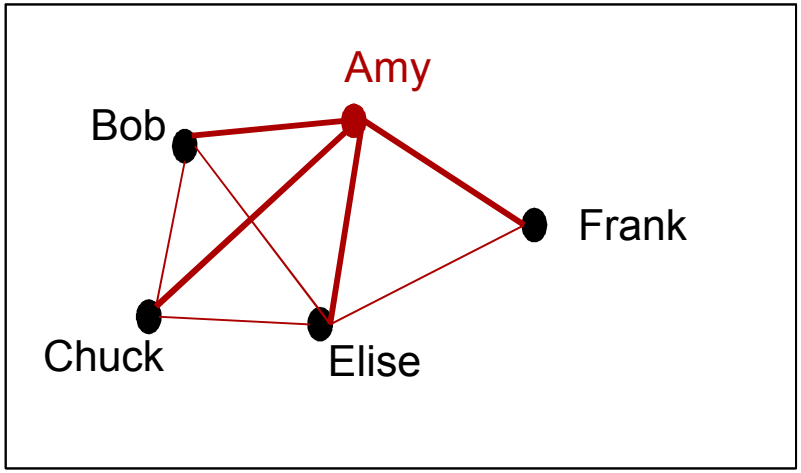
Node Wedges

(A,6)
…

Check for Triangle Closure:

Wedge (A,B)(A,C): Is (B,C) in Adjacency List? Y
Wedge (A,B)(A,E): Is (B,E) in Adjacency List? Y
Wedge (A,B)(A,F): Is (B,F) in Adjacency List? N
Wedge (A,C)(A,E): Is (C,E) in Adjacency List? Y
Wedge (A,C)(A,F): Is (C,F) in Adjacency List? N
Wedge (A,E)(A,F): Is (E,F) in Adjacency List? Y

How a Computer Sees a Graph:



Node A
4 Triangles
2 Open Wedges

Node B?
Node C?
Node D?
….

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
….

Node Degrees

(A,4)
…

Node Wedges

(A,6)
…

Check for Triangle Closure:

Wedge (A,B)(A,C): Is (B,C) in Adjacency List? Y
Wedge (A,B)(A,E): Is (B,E) in Adjacency List? Y
Wedge (A,B)(A,F): Is (B,F) in Adjacency List? N
Wedge (A,C)(A,E): Is (C,E) in Adjacency List? Y
Wedge (A,C)(A,F): Is (C,F) in Adjacency List? N
Wedge (A,E)(A,F): Is (E,F) in Adjacency List? Y

The Trouble with Triangles:

Assume the Graph has *n* nodes and *m* edges.
If a step takes *O(f(x))* time, it takes about* *f(x)* many steps to compute.

Adjacency List

(A,B)
(A,C)
(A,E)
(A,F)
(B,E)
(B,C)
(C,E)
….

Gather Node Edges

(A: B, C, E, F, ..)
(B: A, E, C, ..)
(C: A, B, E, ...)
(E: A, B, C, …)
(F: A, …)
…..

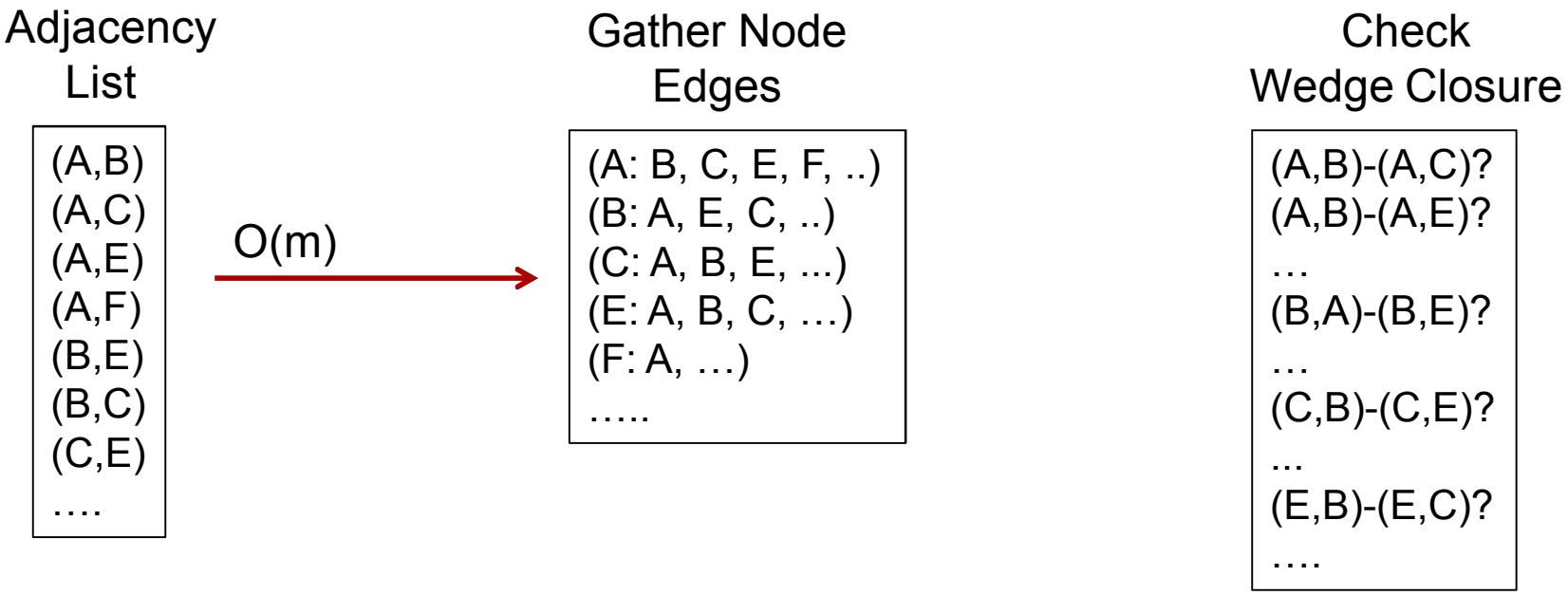
Check Wedge Closure

(A,B)-(A,C)?
(A,B)-(A,E)?
…
(B,A)-(B,E)?
…
(C,B)-(C,E)?
...
(E,B)-(E,C)?
….

*Formally: If a step takes *O(f(x))* time, then given input of size *x*, the step requires at most *(c*f(x) + m)* computations for some constants *c, m*.

The Trouble with Triangles:

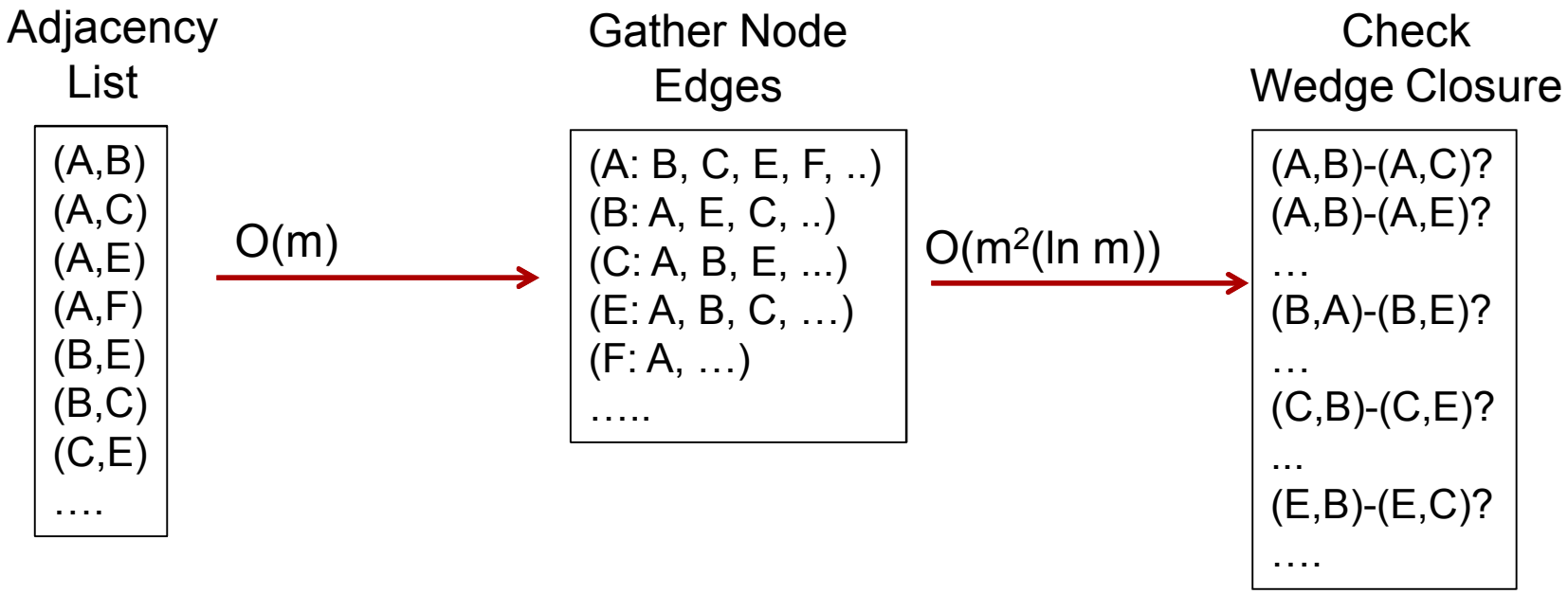
Assume the Graph has *n* nodes and *m* edges.
If a step takes *O(f(x))* time, it takes about *f(x)* many steps to compute.



Step 1: Read through adjacency list and compile list of edges at each node

The Trouble with Triangles:

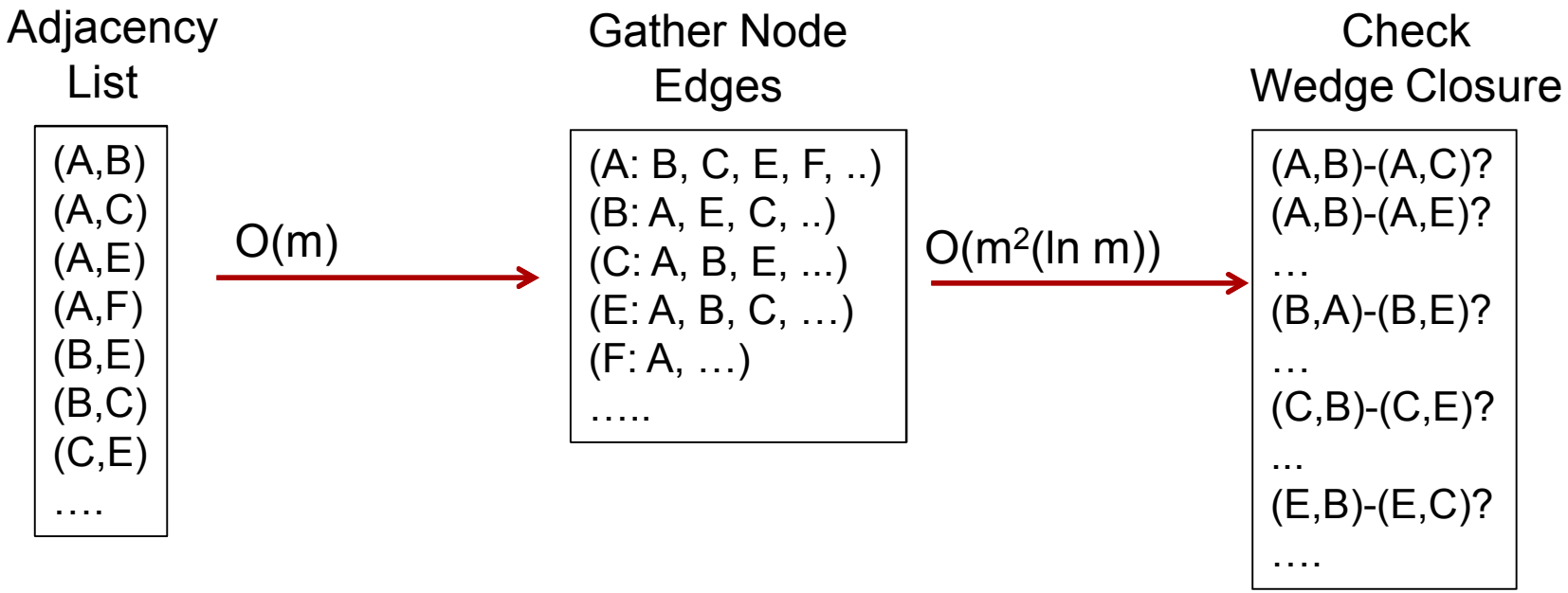
Assume the Graph has *n* nodes and *m* edges.
If a step takes *O(f(x))* time, it takes about *f(x)* many steps to compute.



Step 2: For each wedge, check to see if it's closed to make a triangle. There are maximum m^2 possible wedges, and each check takes O(ln m) look-up time.

The Trouble with Triangles:

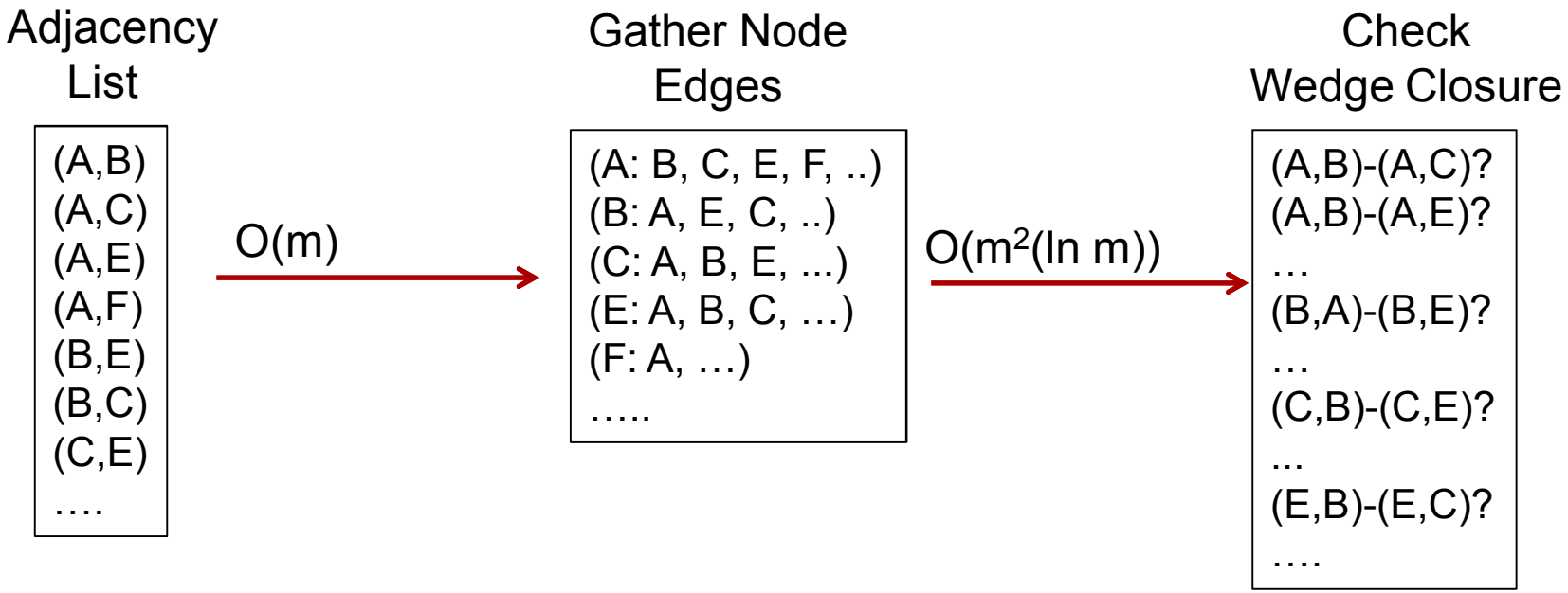
Assume the Graph has *n* nodes and *m* edges.
If a step takes *O(f(x))* time, it takes about *f(x)* many steps to compute.



Time Complexity (worst case, naïve algorithm): $O(m^2(\ln m))$

The Trouble with Triangles:

Assume the Graph has *n* nodes and *m* edges.
If a step takes *O(f(x))* time, it takes about *f(x)* many steps to compute.



Time Complexity (worst case, naïve algorithm): $O(m^2(\ln m))$
But real social networks can have ***hundreds of millions*** of edges!

Parallel Triangle Sampling

Sandia National Laboratories

We can count triangles quickly by making two improvements:

- Don't Count All Triangles: Uniformly randomly sample some wedges from the graph, check them for closure, and use this to create an estimate of the total triangle count.

- Parallelize the Algorithm: Use the Hadoop Map-Reduce framework to work on the adjacency list in parallel.

Parallel Triangle Sampling: Sample

We can get a close estimate of the triangle count on large graphs by uniformly randomly sampling wedges:

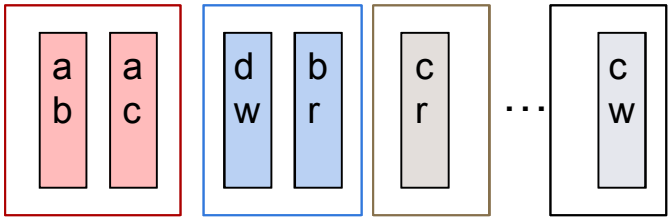
- Each wedge is a random variable which equals 1 if the wedge is closed, and 0 if open. The variable's expected value is equal to the graph's triangle/wedge ratio. This random variable is identical for all wedges.
- Checking if a wedge is closed is like sampling the random variable. The average of many samples converges on the true triangle/wedge ratio.
- Just 38,000 wedge samples gives 0.01 accuracy with 99.9% confidence, much better than actually checking *all* $O(m^2)$ wedges, for $m \approx 10^8$
- We can also compute the true total wedge count easily, from node degrees.
- With an estimate of the triangle/wedge ratio, and true total wedge count, we can accurately estimate the total number of triangles.

C. Seshadhri, A. Pinar and T. G. Kolda. ***Fast Triangle Counting through Wedge Sampling***, arXiv:1202.5230, February 2012.

Parallel Triangle Sampling: Parallel

Rather than read through the (possibly enormous) adjacency list sequentially to find the node degrees, wedges, and check for wedge closure, we can use the Hadoop Map-Reduce framework to operate in parallel.

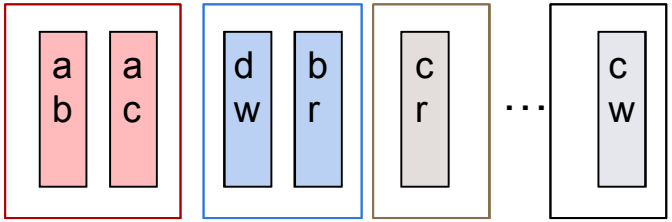
Adjacency List
Stored Across Multiple Processors



Parallel Triangle Sampling: Parallel

Pass 1: This is a detailed look at the first pass of the parallel algorithm

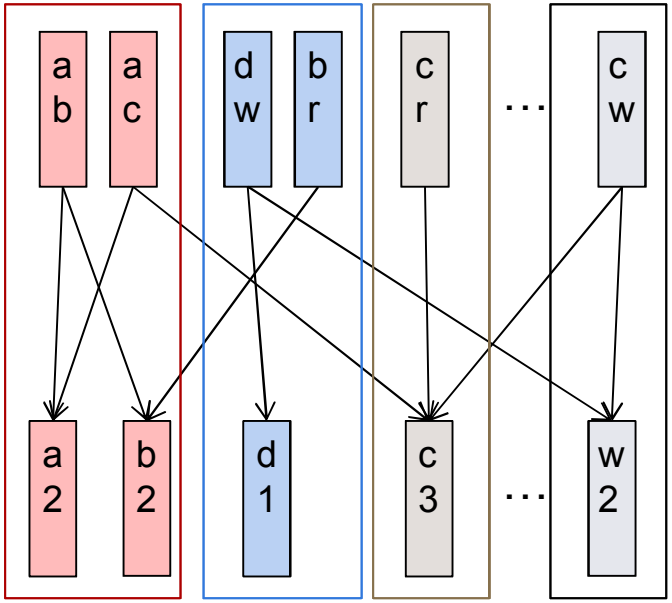
Adjacency List
Stored Across Multiple Processors



Parallel Triangle Sampling: Parallel

Pass 1: This is a detailed look at the first pass of the parallel algorithm

Adjacency List
Stored Across Multiple Processors



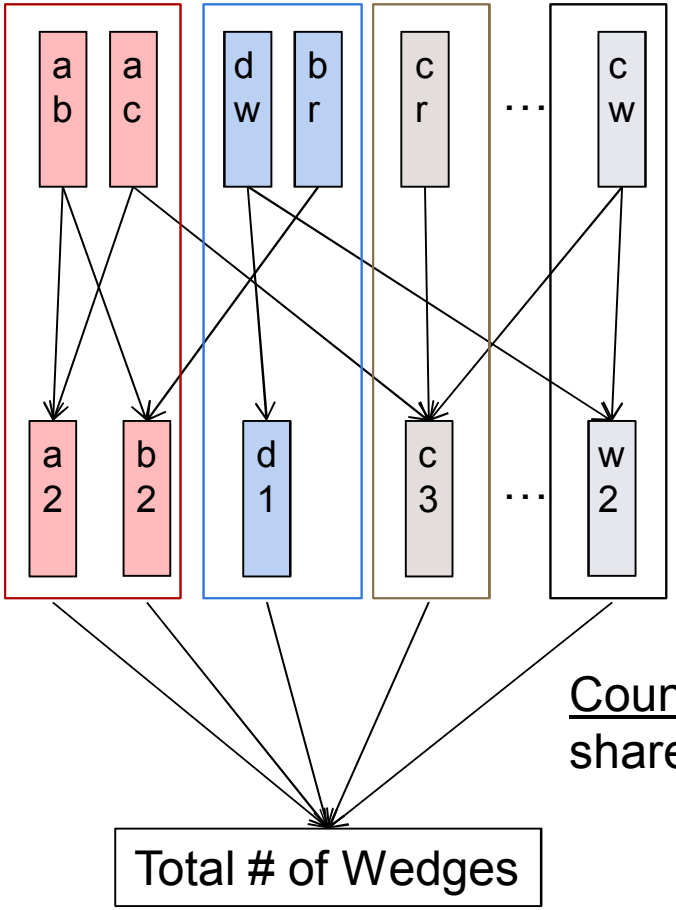
Reduce/ Global Sorting:
Gathers data by node ID. All information about node x goes to same processor

Node Degrees List
Stored across Multiple Processors

Parallel Triangle Sampling: Parallel

Pass 1: This is a detailed look at the first pass of the parallel algorithm

Adjacency List
Stored Across Multiple Processors



Reduce/ Global Sorting:
Gathers data by node ID. All information about node x goes to same processor

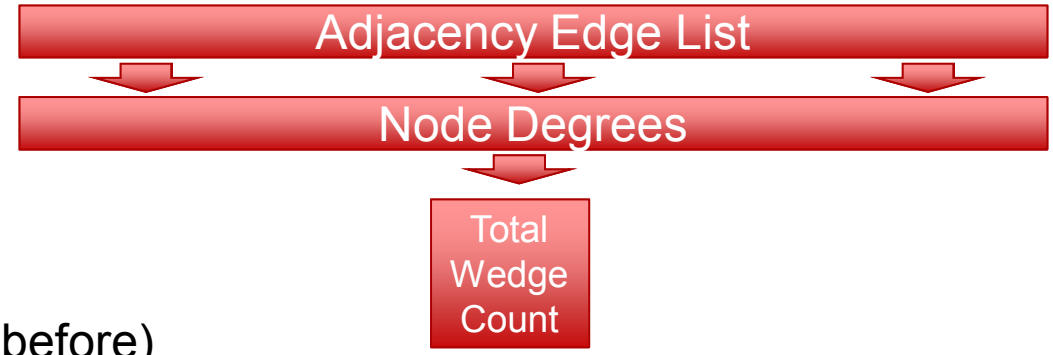
Node Degrees List
Stored across Multiple Processors

Counter: Processors update shared counter in Parallel

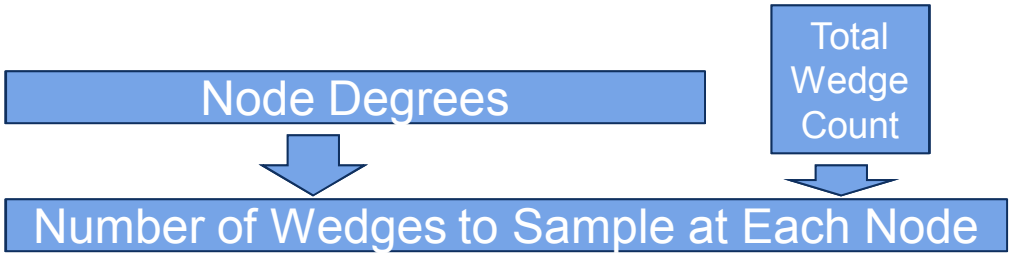
Totals Wedge Counter:
Stored in Shared Variable

Parallel Triangle Sampling: Parallel

Pass 1:
Get Node Degrees and Total Wedge Count (as described before)

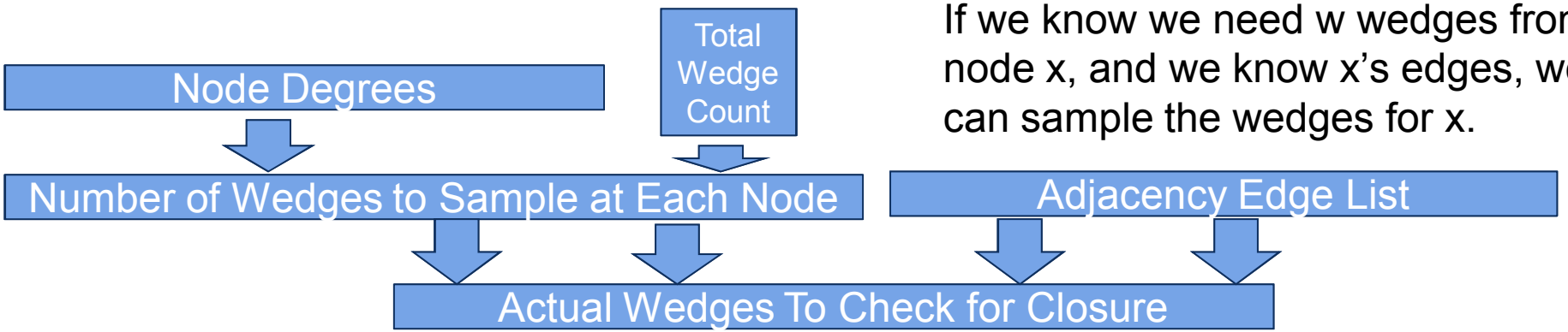


Parallel Triangle Sampling: Parallel



Pass 2:
If we know the degree of a node, and the total number of wedges, then we can compute how many wedges we should sample from that node. This can be done completely in parallel (map step).

Parallel Triangle Sampling: Parallel

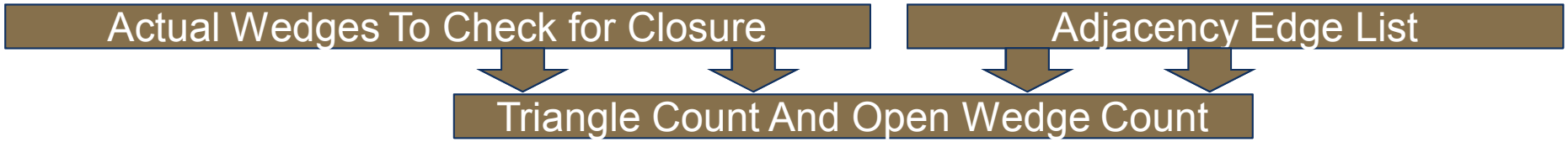


Pass 2:
If we know we need w wedges from node x, and we know x's edges, we can sample the wedges for x.

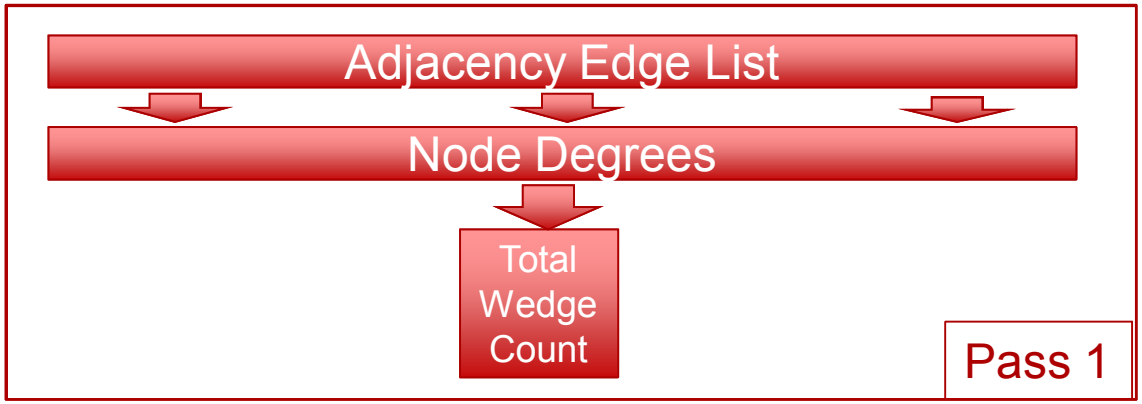
Parallel Triangle Sampling: Parallel

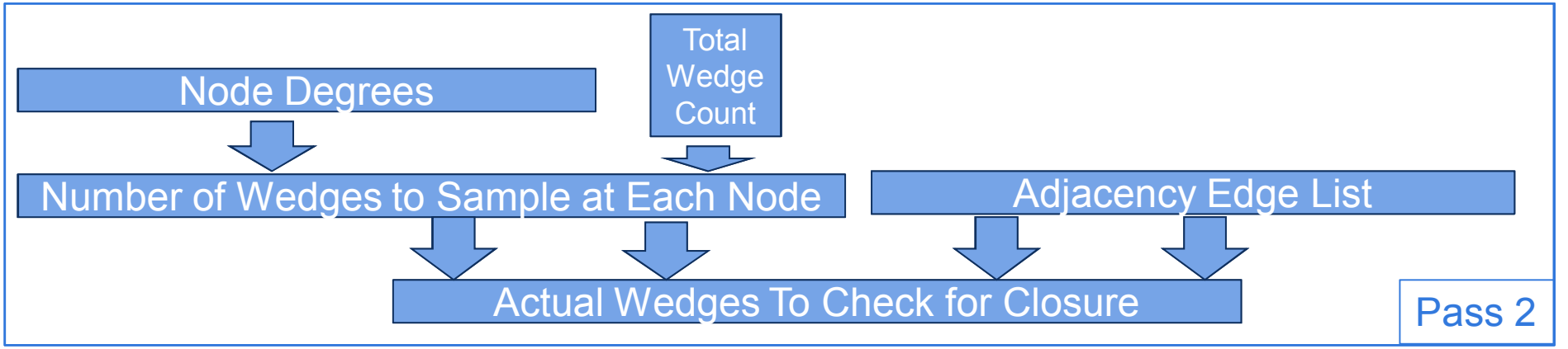
Sandia National Laboratories

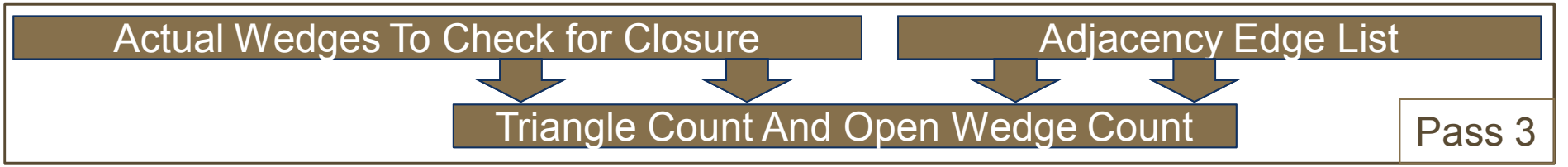
Pass 3:
We can check in parallel whether wedge closure edges exist in the graph, by grouping together (real) edges from the adjacency list with (hypothetical) wedge-closure edges.



Parallel Triangle Sampling: Parallel







Parallel Triangle Sampling: Contributions

Sandia National Laboratories

- Just Three Passes! Instead of sequentially reading the adjacency list many times, we can accomplish our goals by just acting on the adjacency list in parallel three times, with a highly optimized distributed sort step between each pass.

- Generalizable! Remember at the beginning, nodes with lots of friends were also likely to have lots of open wedges? We have also generalized this technique to separate wedges by the degree of their central node, to help us understand that effect better. We can even work on directed graphs (where friendships may not be mutual).

- Additional Optimizations: Some optimizations have been omitted here for simplicity. In the full implementation, secondary sort is used to randomly permute edges in Pass 2, enabling the uniform random sample at the reducer to read minimal data into memory.

Parallel Triangle Sampling: Results

Sandia National Laboratories

Note to Review and Approval Official:
Time allowing, this slide will contain a chart comparing the maximum graph sizes which can be handled by the parallel algorithm described, as compared to the maximum limits of the previously existing non-parallel and non-sampled versions. The graphs will be either publically available online or synthetically generated.

Conclusions

Sandia National Laboratories

- Triangle counts offer a meaningful characterization of the cohesiveness of social networks, but are too time-consuming on large, real social networks.
- Sampling wedges greatly reduces computations while preserving accuracy, but still involves searching the adjacency list repeatedly to find node degrees and check for wedge closures. A large graph => a long list => slow searches.
- Wedge sampling can be reframed in Map Reduce to be highly parallel, allowing us to make all our adjacency list accesses simultaneously in each step.
- *Instead of the computation time depending on the size of the graph, it now depends primarily on the number of processors available.*
- Fast triangle counts on real, large social networks will enable real-time characterization of these networks. This will help us study and understand properties that were previously impossible to see.