Testing Chordal Graphs with **CUDA®**

Agnieszka Łupińska
PhD student at Jagiellonian University

5 kwietnia 2016

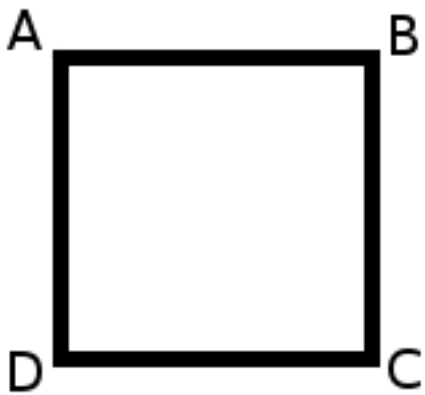
Agenda

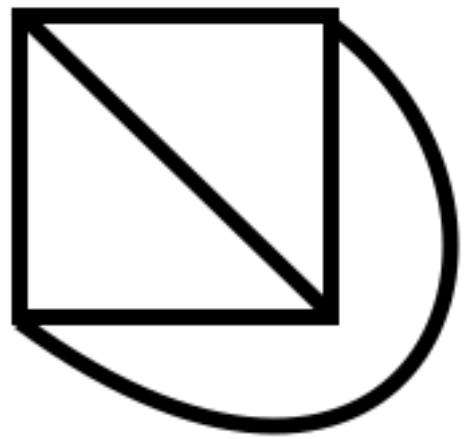
1. Preliminaries
2. Algorithm to testing chordal graphs: the necessary and sufficient condition for a graph to be chordal
3. Motivation
4. The parallel approach
5. Performance test results
6. Summary

Preliminaries

A chord is an edge between 2 non-adjacent vertices on the cycle in a graph.

A graph is chordal if each cycle of size greater than 3 has a chord.



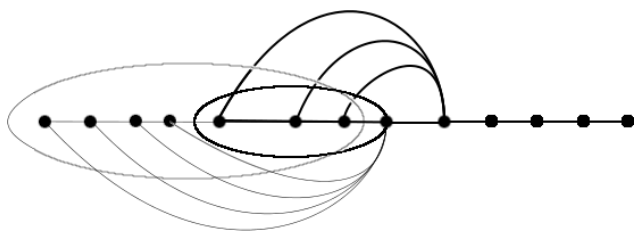


Preliminaries

Chordal graphs are characterized by existance a perfect elimination order (PEO) on vertices.

The order $\pi = v_1, \ldots, v_N$ is a PEO if for each i the neighbors placed on the left from v_i induce a clique.

$v_1, v_2, v_3, \ldots,\ v_i$ - clique.



Preliminaries

A LexBFS order is produced by the LexBFS algorithm.

LexBFS is a restriction of the Breadth-first search (BFS) algorithm, in the following sense: each possible order produced by LexBFS is a BFS order, but not every BFS order is the LexBFS order.

The difference is:
- BFS - **FIFO queue** - priority time
- LexBFS - **priority queue** - lexicographically order on labels of vertices

Preliminaries

LexBFS algorithm proposed by Habib, McConnell, Paul and Viennot in 2000. It uses the partition refinement technique with pivots.

```
LexBFS()
L = (V)
for i = 1 .. N do
    pivot <- remove the first vertex from the first class in L
    PEO(i) <- pivot
    for each C in L in parallel do
        C(pivot) <- C ∩ Adj(pivot)
        C <- C \ C(pivot)
        replace C in L by C(pivot),C
```

Algorithm to testing chordal graphs: the necessary and sufficient condition for a graph to be chordal

The algorithm to test chordality of graphs is based on the following theorem introduced by D. J. Rose, R. E. Tarjan, G. S. Leuker in 1976:

A graph G is chordal if and only if a LexBFS order of G is a perfect elimination order.

```
chordalityTest(G)
    P <- compute a LexBFS order of G
    if P is a perfect elimination order then
        return YES
    else
        return NO
```

Motivation

The LexBFS algorithm is used as a part of many graph algorithms such as:
- recognizing interval graphs
- computing transitive orientation of comparability/co-comparability graphs

Graph is interval if is chordal and co-comparability.

I am going to find the CUDA implementation of the algorithm to recognize the Interval graphs.

The parallel approach: data structures

We use N threads assigned to N vertices in a graph.

At the beginning all vertices are stored in the class C, and the class C is stored in the linked list L.

The pivot is a global variable shared by all threads, and it stores the vertex number with the lexicographically largest label

The vertex is active if it is in the list L. At the beginnig all vertices are active.

The parallel approach: the parallel LexBFS

The first step is computing the LexBFS order. The main loop runs on the Host and for each pivot we run 4 kernels

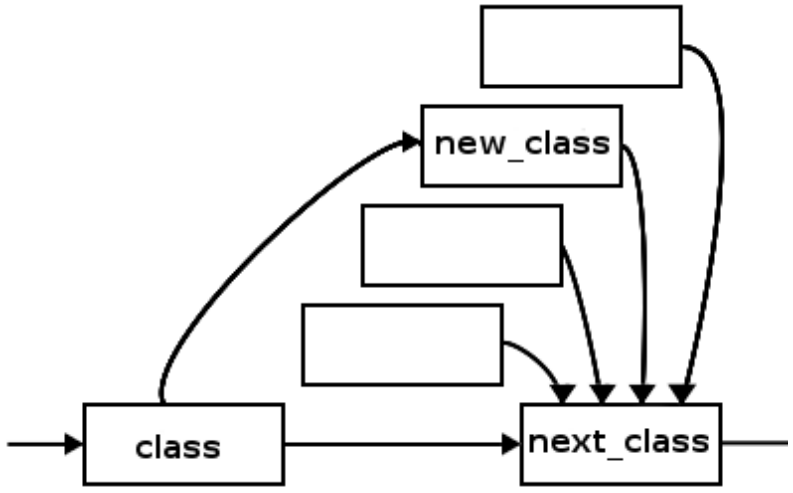
```
function LexBFS(N - vertex number):
    pivot <- vertex o number 1
    for time <- 1 to N do:
        setup_kernel
        partition_kernel
        removal_kernel
        get_next_pivot_kernel
    end for
```

The parallel approach: the parallel LexBFS

Setup:

```
kernel setup:
    x <- the vertex number
    if x is active then
        write to global memory some pointers of x
        if x is pivot then
            peo_order[time] = x
            mark x as inactive
        end if
    end if
```

Partition:



The parallel approach: the parallel LexBFS

Removal:

```
kernel removal:
    x <- the vertex number
    if x is active then
        isEmpty[class(x)] <- false
        tmp <- class(x)->next
    end if
```

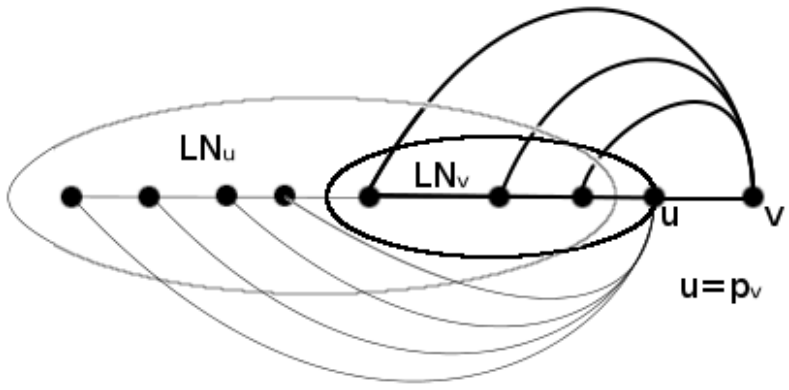
Get next pivot:

```
kernel nextPivot:
    x <- the vertex number
    if x is active then
        if isEmpty[tmp] then
            class(x)->next <- tmp->next
        end if
        if class(x)->next is null then
            pivot <- x
        end if
    end if
```

The parallel approach: testing a PEO order

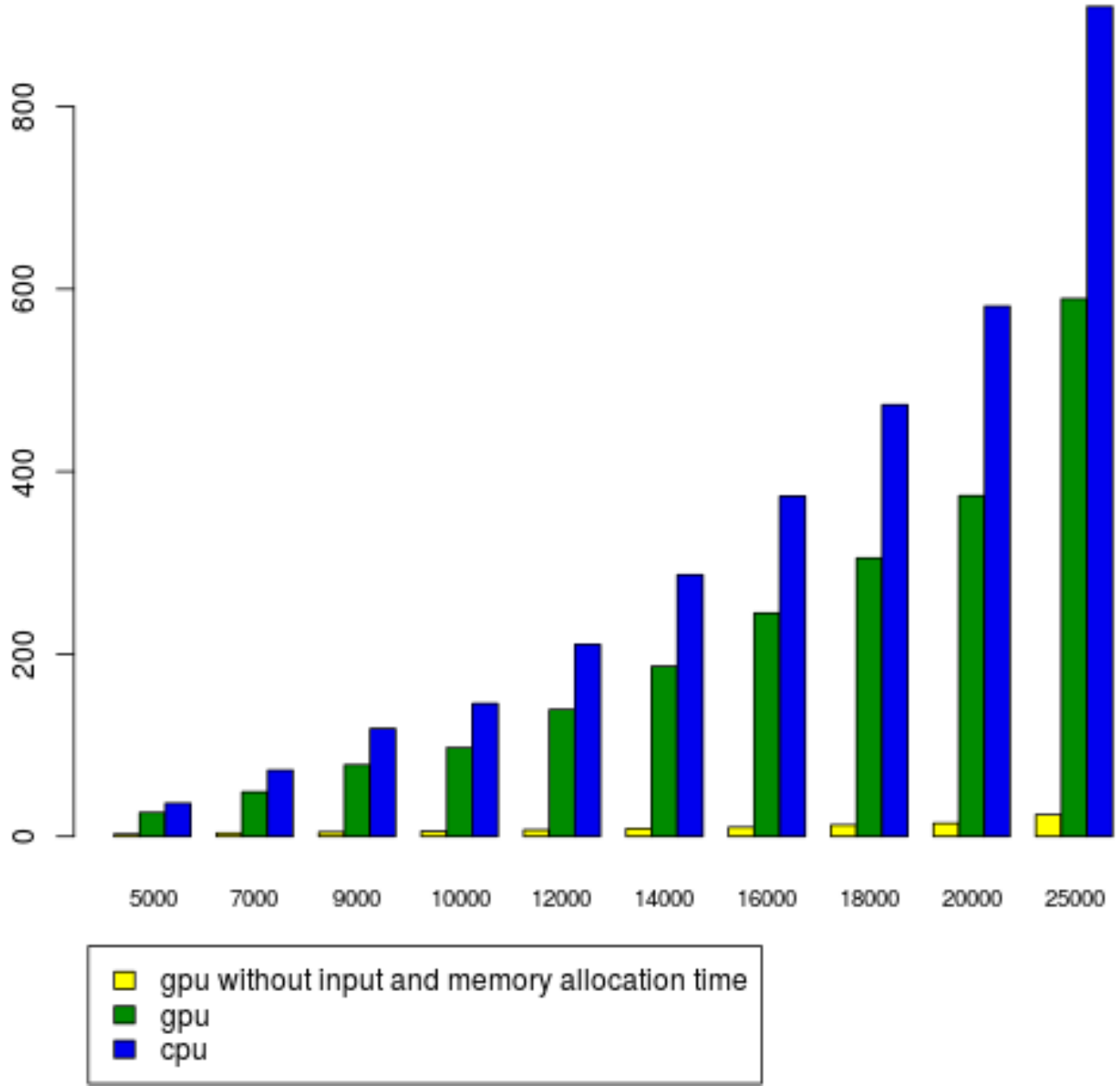
Let π - an order of G, N_v - a neighborhood of v in G, $LN_v \subset N_v$ - left neighbors of v and $p_v \in LN_v$ be the most right vertex in LN_v.

To test if π is a perfect elimination order we only need to check if for each v is $LN_v - \{p_v\} \subset LN_{p_v}$

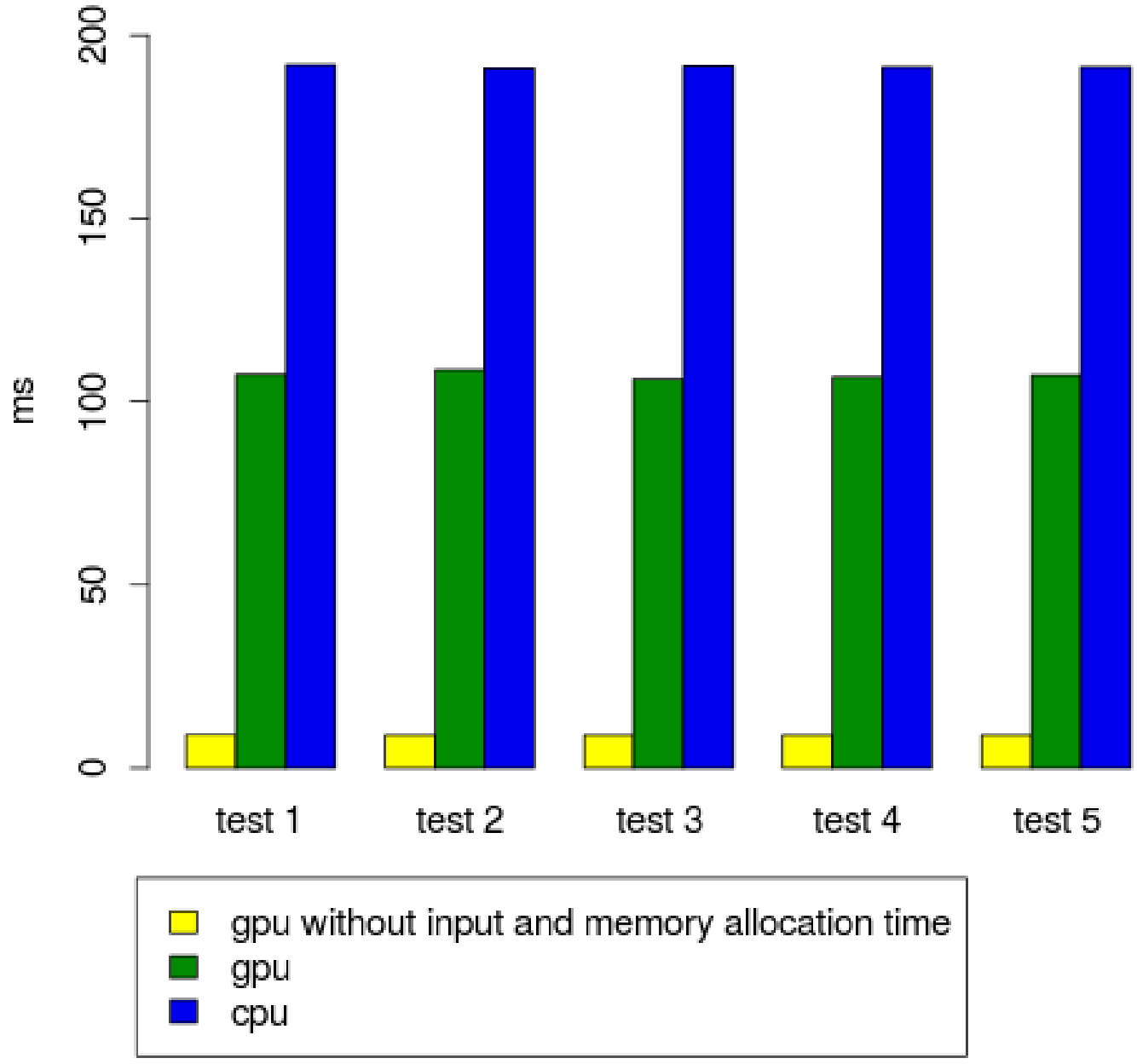



```
kernel PEO_testing:
    v <- the vertex number
    for each x adjacent to v do:
        if x is not adjacent to p(v) then
            isChordal <- false
        end if
    end for
```

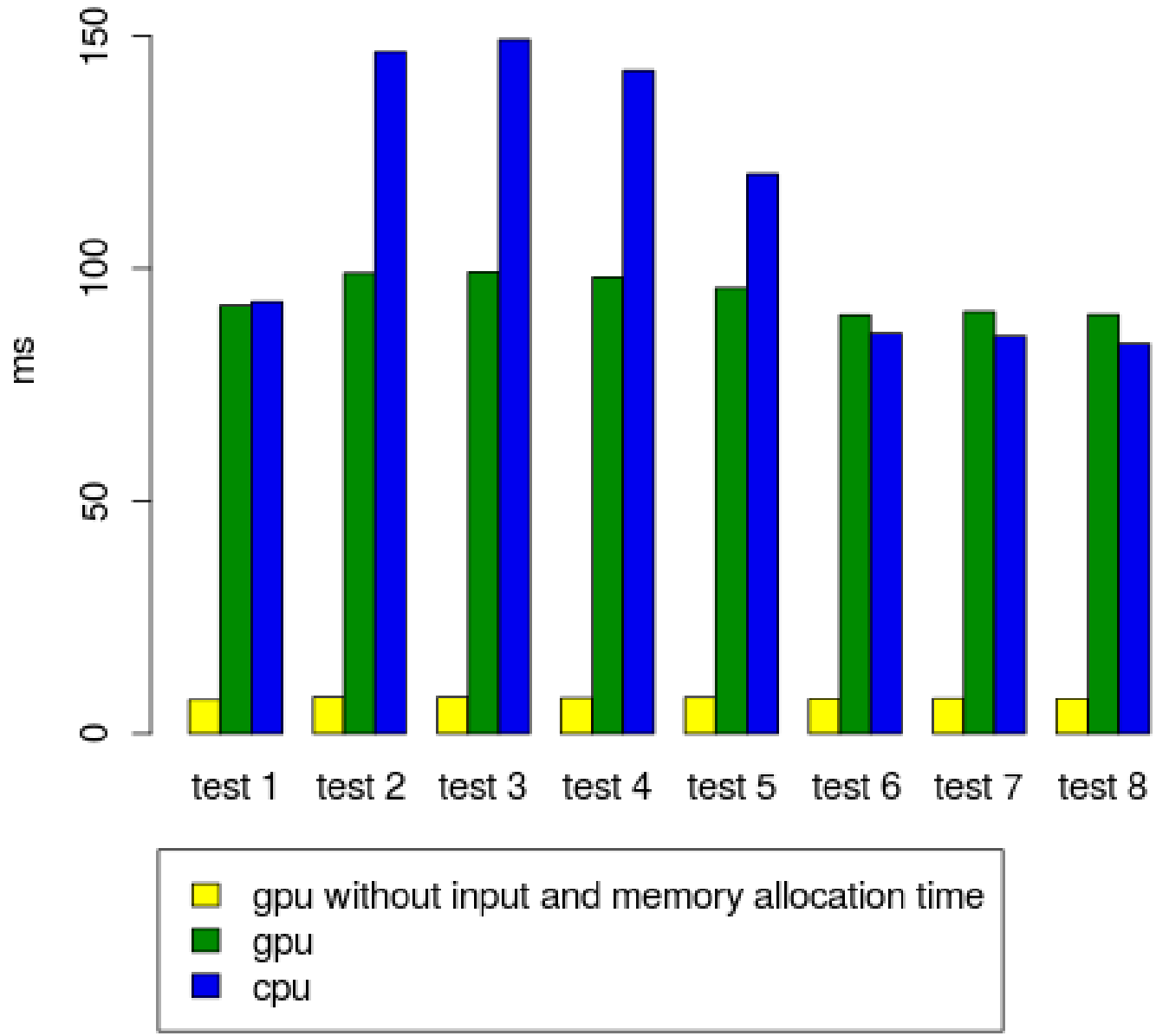
Performance test results: cliques



Performance test results: dense graphs, $M = O(N^2)$



Performance test results: graphs with random size of M



Summary

The sequential algorithm takes a **O(N+M)** time.
The parallel algorithm takes a **O(N)** time and is not sensitive to the size of the graph shown on the entrance.

For more details please see the lexBFS-chordalityTest project at:

https://bitbucket.org/agalup/

Thank you for your attention