Questions

1. What is the runtime of Luby’s algorithm, and on how many processors?
2. Name one library in which the Tsukiyama algorithm is implemented.
3. Name two problems related to independent sets
Meet Terry Henderson

- From Hiram, Georgia
- Studies IE: Optimization
- Advisor: Jim Ostrowski
- Masters in Math from Wake Forest
- Interests:
  - Story-telling
  - Video Games
  - Movies/TV
Meet Ben Olson

- Grew up in Memphis, TN
- Advisor: Dr. Michael Jantz
- Studies memory management
- Enjoys video games, hiking, systems programming
Outline

● Overview
● Definitions
● History
● Applications
● Algorithms
● Implementation and Results
A review of some basic terminology:

- **Independent set** - a collection of vertices with no (pairwise) edges between them
- **Clique** - a complete (sub)graph, i.e. a set of vertices all of which have an edge between them
Independent Set Definition
Definitions

**Complement** - A graph is the complement of a graph $G$ if its vertices are adjacent if and only if the vertices are not adjacent in $G$.

Independent sets in $G$ are cliques in $G$ complement.
Complement Definition
Independent sets and multipartite graphs

**Multipartite graphs** or **k-partite graphs** are graphs that can be divided into k sets of nodes with no edges between them.

These k partite sets are by definition independent sets!

Thus discussion of multipartite graphs is exactly discussion about its independent sets.

**K-cliques** are k-partite graphs with all possible edges between partite sets.
An Interesting Relationship

Partite sets in a k-clique correspond to disjoint (maximal) cliques in the complement.
Matchings

**Matching** - a set of edges without common vertices (sometimes called independent edge set)

A **perfect matching** is a matching that covers all vertices.

If we restrict to a perfect matching we have a bipartite graph (two independent sets).
A Perfect Matching
Maximal v. Maximum

- A maximum is the largest such subgraph in question
  - In our case the independent set with the most vertices
- A maximal subgraph is one to which no additional vertices/edges can be added.
  - An independent set for which each of the vertices in the set is adjacent to a vertex outside the set
- A maximum subgraph is always a maximal subgraph but the reverse may not be true.
Maximum and Maximal Independent Sets

Maximal *and* Maximum

Maximal
Problems related to Independent sets

- Cliques
- K-partite graphs
- Graph coloring
- Dominating sets
- Vertex cover
1669 - Athanasius Kircher publishes an extension of the work of Llull (Ars Magna Scienede Sive Combinatoria - The Great Art of Knowledge) in which he provides illustrations of complete bipartite graphs.
History

1914 - Dénes Konig presents results on regular bipartite graphs to the Congress of Mathematical Philosophy in Paris

1931 - Konig's Thm published: the size of the maximum independent set plus the size of the maximum matching is equal to the number of vertices.
History

1941 - Turan's theorem:
   If G is K_r free the number of edges is at most \((1-1/(r-1)) n^2/2\)

1949 - Luce and Perry coin the word "clique" in graph theory

1960 - Miller and Muller give bound on maximum number of maximal independent sets in a graph:

\[ |\text{MIS}(G)| \leq g(n) \]

\[ g(n) := \begin{cases} 
3^{n/3} & \text{if } n \equiv 0 \pmod{3} \\
4 \cdot 3^{(n-4)/3} & \text{if } n \equiv 1 \pmod{3} \\
2 \cdot 3^{(n-2)/3} & \text{if } n \equiv 2 \pmod{3}.
\end{cases} \]
History

1973 - Bron Kerbosch algorithm for maximal clique enumeration
1978 - Yannakakis:
   Odd cycle transversal is an NP-complete algorithmic problem that asks, given a graph $G = (V,E)$ and a number $k$, whether there exists a set of $k$ vertices whose removal from $G$ would cause the resulting graph to be bipartite.
1979 - Garey, Johnson:
   Given a bipartite graph, testing whether it contains a complete bipartite subgraph $K_{i,i}$ for a parameter $i$ is an NP-complete problem.
Determining whether a general graph is k-partite for $k>2$ is NP complete.

**Building the complement** - $O(n^2)$

**Nick Class complexity hierarchy**

$NC_i$ is the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors.

It is known that finding a maximal independent set is $NC_2$ using Luby’s algorithm which we discuss a little later.
Why Independent Sets v. Cliques?

$O(n^2)$ to find the complement is a costly conversion, so when should we use MIS algorithms and when should we use Clique algorithms?

- When we already have the complement
  - I.e. we have a graph, but we care about properties of the complement
- When taking the complement gives favorable properties
  - E.g. we are interested in matrix operations but our adjacency matrix is very dense
- When graphs lend themselves to a particular algorithm
  - E.g. clique algorithms are better at chordal graphs, and MIS algorithms are very bad at these types of graphs
Applications

- Parallelization
  - Graphs with multiple independent sets lend themselves to parallelization (especially when reduced to a perfect matching)

- Molecular Biology
  - Protein docking (clique detection)
  - Genome Mapping Data Integration (maximal Cliques)

- Scheduling
- (Weighted) Set Packing
- Channel routing in physical design automation
- Memory Management
Applications: Molecular Biology

- **Protein Docking**
  - Goal is to find whether proteins interact to form a stable complex
  - Problem is fundamental to all aspects of biological function
  - One approach uses a clique detection algorithm to find hydrogen bond donor/acceptor pairs

- **Integration of Genome Mapping Data**
  - Finding “virtual probes” is equivalent to finding the maximal cliques of a graph
    - Virtual probes are sets of mutually overlapping data
    - Each set determines a site in the genome corresponding to the region which is common among clones of the set
Applications: Scheduling

- Model each object that needs to be scheduled as a vertex
- Vertices are adjacent if there is a conflict (i.e. cannot be scheduled at the same time)
- Find the maximal independent sets
Applications: Weighted Set Packing

- The WSP problem is:
  - Given a set $S$ of $m$ elements and a collection of weighted subsets of $S$, find a subcollection of disjoint sets of maximum total weight
  - Has a variety of applications in practical optimization problems
    - Paper gives the examples of “formation of coalitions in multiagent systems” and “model[ing] multi-unit combinatorial auctions”
  - A weighted set packing can be transformed into a weighted independent set instance on $n$ vertices
    - Approximations of WIS correspond to approximations of WSP
    - (WIS takes graph with vertex weights and gives the independent set with maximum total weight)
Applications: Channel Routing

- Utilizes (vertex) transitive graphs:
  - Any two vertices are equivalent under some automorphism
    - i.e. a bijection from the vertices to the vertices
- Example: k-layer routing for Printed Circuit Boards
  - Constructs Horizontal and Vertical Constraints Graph
  - Independent sets in the VCG may be placed “on the same track but on different layers”
    - Vertices may also be assigned weights reflecting priorities on their selection
Applications: Memory Management
Applications: Memory Management
Algorithms

- Three problems:
  - Find a maximal independent set (called MIS problem)
  - Find all maximal independent sets (also called MIS problem)
  - Find the maximum independent set
Finding one maximal independent set can be solved in polynomial time, $O(\log(n)^2)$ with $O(m)$ processors, by a greedy algorithm [Luby].

All maximal independent sets can be found in worst-case $O(3^{n/3})$ [Tomita].

All maximal independent sets can be found in $O(nmu)$ [Tsukiyama].
Luby’s Algorithm

Until graph is empty:
  For each vertex:
    Put in S with prob. 1/2d
  For each edge:
    If both vertices are in S:
      Remove v with lower degree from S and graph
  For each vertex in S:
    Remove vertex from graph

S: 0 1
Luby’s Algorithm

Until graph is empty:
  For each vertex:
    Put in $S$ with prob. $1/2d$
  For each edge:
    If both vertices are in $S$:
      Remove $v$ with lower degree from $S$ and graph
  For each vertex in $S$:
    Remove vertex from graph

S: 0 1
Luby’s Algorithm

Until graph is empty:
For each vertex:
  Put in S with prob. 1/2d
For each edge:
  If both vertices are in S:
    Remove v with lower degree from S and graph
For each vertex in S:
  Remove vertex from graph
Luby’s Algorithm

Until graph is empty:
  For each vertex:
    Put in S with prob. 1/2d
  For each edge:
    If both vertices are in S:
      Remove v with lower degree from S and graph
    For each vertex in S:
      Remove vertex from graph

S: 0 2
Luby’s Algorithm

Until graph is empty:
  For each vertex:
    Put in $S$ with prob. $1/2d$
  For each edge:
    If both vertices are in $S$:
      Remove $v$ with lower degree from $S$ and graph
  For each vertex in $S$:
    Remove vertex from graph

$S$: 0 2
Luby’s Algorithm

Until graph is empty:
  For each vertex:
    Put in $S$ with prob. $1/2d$
  For each edge:
    If both vertices are in $S$:
      Remove $v$ with lower degree from $S$ and graph
  For each vertex in $S$:
    Remove vertex from graph

$S$: 0
Luby’s Algorithm Analysis

- Intended to be a parallel algorithm running on $O(m)$ processors
- Requires shared memory device, constant communication
- Requires constant deletion of vertices, adding to a set in parallel
Tsukiyama Algorithm

- A New Algorithm for Generating All the Maximal Independent Sets by Tsukiyama, Ide, Ariyoshi, and Shirakawa
- Written in 1977
- Fortran, NEAC 2200/700
- Also the common algorithm used in graph libraries (i.e. igraph)
- $O(nmu)$
Let’s try this algorithm out on an extremely small graph:
Tsukiyama Algorithm

Buckets:

IS: 0 0 0

C = 1
Tsukiyama Algorithm

Buckets:

IS: 0 1 0

C = 1
Tsukiyama Algorithm

Buckets:

IS: 0 1 0

C = 1
Tsukiyama Algorithm

![Diagram of the Tsukiyama Algorithm with nodes 0, 1, and 2 connected and bucket and IS label with values 0, 1, 1, and C = 1]
Tsukiyama Algorithm
Tsukiyama Algorithm

MIS: 0 1 1

Buckets:

IS: 0 1 0

C = 1
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

**Buckets:**

<table>
<thead>
<tr>
<th>IS:</th>
<th>0 1 0</th>
</tr>
</thead>
</table>

**C = 1**

**Y = 0**

**F = 1**
Tsukiyama Algorithm

MIS: 0 1 1

Buckets: | 0 |

IS: 0 0 0

C = 1
Y = 0
F = 1
Z = 1
MIS | 0 1 1
---|---

Tsukiyama Algorithm

C = 1
Y = 0
F = 0
Z = 1

Buckets:

IS: | 0 0 0
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

| IS: | 1 0 0 |

| Buckets: | 0 | 1 |

| C = 1 |
| Y = 1 |
| F = 0 |
| Z = 1 |
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

0 1 1

0 1

0 0 0

C = 1
Y = 1
F = 0
Z = 0
Tsukiyama Algorithm

MIS: 0 1 1

Buckets:

C = 1
Y = 1
F = 0
Z = 0

IS: -1 0 0
Tsukiyama Algorithm

MIS | 0 1 1
--- | ---

Buckets:

IS: | 0 1 0
--- | ---

C = 1
Y = 1
F = 0
Z = 0
Tsukiyama Algorithm

MIS

| 0 | 1 | 1 |

0

1

2

Buckets:

IS:

| 0 | 1 | 0 |

C = 1

Y = 1

F = 1
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

Graph:

- Node 0
- Node 1
- Node 2

Buckets:

| IS: | 0 0 0 |

Properties:

- C = 1
- Y = 1
- F = 1
Tsukiyama Algorithm

MIS: 0 1 1

Buckets:

IS: 1 0 0

C = 1
Y = 1
F = 1
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

- **Buckets:** 0
- **IS:** 1 0 0
- **C:** 1
- **Y:** 1
- **F:** 1
Tsukiyama Algorithm

MIS: 0 1 1

Buckets:

IS: 1 0 0

C = 2
Y = 1
F = 1
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0 1 1</th>
</tr>
</thead>
</table>

**Buckets:**

- **C = 2**
- **F = 1**

**Is:**

- 1 0 2
Tsukiyama Algorithm

<table>
<thead>
<tr>
<th>MIS</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Buckets:**
  - 0

- **IS:**
  - 1 0 2

- **C:** 2
- **Y:** 1
- **F:** 1
Tsukiyama Results

**Number of Vertices vs. Runtime**

**Density vs. Runtime**
Some Open Issues and Future Directions

● Is maximal independent set problem NC\(_1\)?
● Erdos-Faber-Lovasz conjecture:
  ○ If k complete graphs having exactly k vertices have the property that every pair of complete graphs share at most one vertex, then the union of the graphs can be colored with k colors.
● Almost-independent sets (almost-complete graphs)
  ○ Similar to almost symmetries in graphs
References

- Halldorsson, Approximations of Weighted Independent Set and Hereditary Subset Problems, http://www.emis.de/journals/JGAA/accepted/00/Halldorsson00.4.1.pdf
References

- Barbosa, *Network Conduciveness with Application to the Graph-Coloring and Independent-Set Optimization Transitions*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2900201/
Discussion
Questions Round 2

1. What is the runtime of Luby’s algorithm, and on how many processors?
2. Name one library in which the Tsukiyama algorithm is implemented.
3. Name two problems related to independent sets