GENERATION OF MINIMUM SPANNING TREES

on undirected, edgeweighted graphs with the algorithms of Kruskal, Prim and Boruvka

Cottbus, 2015-10-02

Paul Jähne
Why?

• why not?
• MST useful in real life
• practice C, MPI and associated tools
• project for module "Algorithm Engineering"
What?

- comparison of algorithms for MST generation
- Kruskal's algorithm: $O(|E| \log |E|)$
- Prim's algorithm: $O(|E| + |V| \log |V|)$ or $O(|E| \log |V|)$
- Boruvka's algorithm: $O(|E| \log |V|)$
"Quite frankly, even if the choice of C were to do nothing but keep the C++ programmers out, that in itself would be a huge reason to use C."
– Linus Torvalds

Use the source, Luke!
Implementation of Kruskal's algorithm

- parallelized merge sort with small modification
- union find data structure with path compression and union by rank
Implementation of Prim's algorithm

- binary heap with array
- Fibonacci heap with doubly linked list
- adjacency list
Implementation of Boruvka's algorithm

- union find data structure with path compression and union by rank
- parallelized search
In short

- written by myself
- C with MPI
- Open Source (LGPLv3)
- https://github.com/SethosII/minimum-spanning-tree
Wherewith?
## List of real graphs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vertices</th>
<th>Edges</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Full USA</td>
<td>23,947,347</td>
<td>58,333,344</td>
<td>2.03E-7</td>
</tr>
<tr>
<td>CTR</td>
<td>Central USA</td>
<td>14,081,816</td>
<td>34,292,496</td>
<td>3.45E-7</td>
</tr>
<tr>
<td>W</td>
<td>Western USA</td>
<td>6,262,104</td>
<td>15,248,146</td>
<td>7.77E-7</td>
</tr>
<tr>
<td>E</td>
<td>Eastern USA</td>
<td>3,598,623</td>
<td>8,778,114</td>
<td>1.35E-6</td>
</tr>
<tr>
<td>LKS</td>
<td>Great Lakes</td>
<td>2,758,119</td>
<td>6,885,658</td>
<td>1.81E-6</td>
</tr>
<tr>
<td>CAL</td>
<td>California and Nevada</td>
<td>1,890,815</td>
<td>4,657,742</td>
<td>2.60E-6</td>
</tr>
<tr>
<td>NE</td>
<td>Northeast USA</td>
<td>1,524,453</td>
<td>3,897,636</td>
<td>3.35E-6</td>
</tr>
</tbody>
</table>

... and more

9. DIMACS Implementation Challenge - Shortest Paths
## List of generic graphs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vertices</th>
<th>Edges</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>Graph with density 1</td>
<td>10,000</td>
<td>49,995,000</td>
<td>1.00E0</td>
</tr>
<tr>
<td>HALF</td>
<td>Graph with density 0.5</td>
<td>10,000</td>
<td>24,997,500</td>
<td>5.00E-1</td>
</tr>
<tr>
<td>QUAR</td>
<td>Graph with density 0.25</td>
<td>10,000</td>
<td>12,498,750</td>
<td>2.50E-1</td>
</tr>
<tr>
<td>GRID</td>
<td>Grid graph</td>
<td>10,000</td>
<td>19,800</td>
<td>3.96E-4</td>
</tr>
<tr>
<td>PATH</td>
<td>Path graph</td>
<td>10,000</td>
<td>9,999</td>
<td>2.00E-4</td>
</tr>
</tbody>
</table>
Results?
Runtime for real data (sequential)
Runtime as a function of density (sequential)
Comparison to asymptotic runtime

![Graph showing comparison to asymptotic runtime.](image)

- Kruskal
- Prim (Fib.)
- Prim (Bin.)
- Boruvka

Axes:
- Y-axis: Scaled runtime
- X-axis: Edges

Values range from 0.00E+00 to 3.50E-08.
Other papers on comparing minimum spanning tree algorithms

- only generic data and not publicly available
- implementation not publicly available
- no parallelization
- Prim's algorithm fastest
Conclusion

Boruvka's algorithm outperforms others in speed and memory usage

→ contrary to other papers

Why do we have differing results? I couldn't say!
Backup
Test system

- Dell R720 Server, two Intel Xeon E5-2680 v2 CPUs, 512 GB RAM
- Operating System: CentOS 6.6, Linux Kernel 2.6.32
- Compiler GNU C Compiler 4.8.1
- MPI: OpenMPI 1.8.2
- mpicc -std=c11 -O3 -march=corei7-avx -lm -o Release/MST src/main.c
Measurement

• with MPI_Wtime
• average of 30 runs
• no parallel use of the server