Boost.Graph for Beginners

- How do I define a graph?
- What algorithms can I use?
- How (the heck) do I use algorithms?
- Where do I find more information?

Boris Schäling, boris@highscore.de

C++Now, Aspen, 16 May 2013

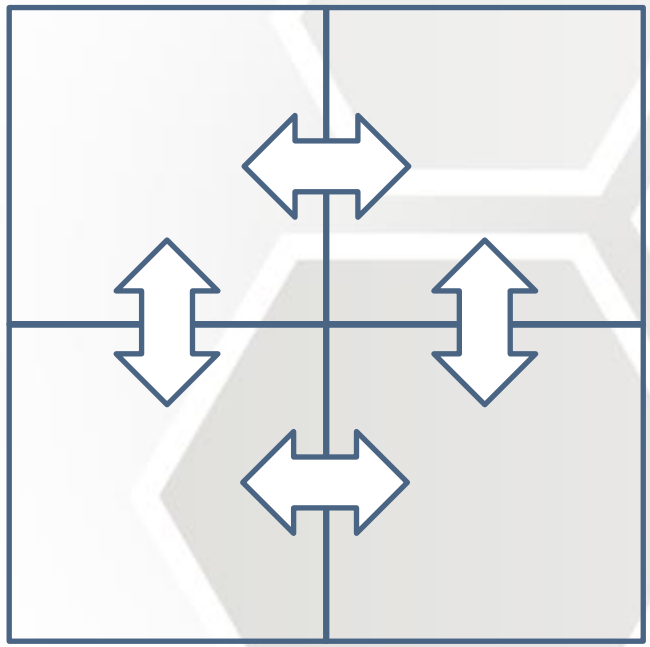
Overview

Boost.Graph provides classes to define graphs and algorithms to search in graphs

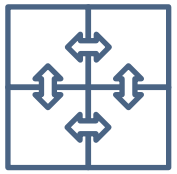
- Most important class is adjacency_list
- Various adaptors for non-Boost.Graph types
- Two most important core search algorithms:
  - void breadth_first_search(...)
  - void depth_first_search(...)
- Visitors which do something when algorithms visit vertices and edges
- Various shortest path and many more algorithms

Sample graph

All code examples in this presentation are based on the following graph:



Graph definition



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

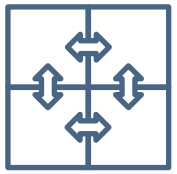
```
#include <boost/graph/adjacency_list.hpp>

using namespace boost;

adjacency_list<> g;

// adds four vertices to the graph
adjacency_list<>::vertex_descriptor v1 = add_vertex(g);
adjacency_list<>::vertex_descriptor v2 = add_vertex(g);
adjacency_list<>::vertex_descriptor v3 = add_vertex(g);
adjacency_list<>::vertex_descriptor v4 = add_vertex(g);
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Graph definition



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// gets vertices
typedef adjacency_list<>::vertex_iterator iterator;
std::pair<iterator, iterator> p = vertices(g);

// prints vertices: 0, 1, 2, 3
for (auto it = p.first; it != p.second; ++it)
  std::cout << *it << std::endl;

// prints std::size_t
std::cout << typeid(v1).name() << std::endl;
```

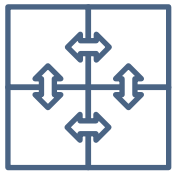
100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Graph definition



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

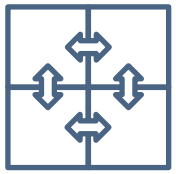
```
std::pair<adjacency_list<>::edge_descriptor, bool> p;

// adds four edges to the graph
p = add_edge(v1, v2, g);
p = add_edge(v2, v3, g);
p = add_edge(v3, v4, g);
p = add_edge(v4, v1, g);

// accesses and prints the edges: (0,1), (1,2), (2,3), (3,0)
auto p = edges(g);
for (auto it = p.first; it != p.second; ++it)
  std::cout << *it << std::endl;
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Graph definition



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// defines a graph with undirected edges
adjacency_list<vecS, vecS, undirectedS> g;

enum { topLeft, topRight, bottomRight, bottomLeft };

// adds edges without explicitly adding vertices
add_edge(topLeft, topRight, g);
add_edge(topRight, bottomRight, g);
add_edge(bottomRight, bottomLeft, g);
add_edge(bottomLeft, topLeft, g);
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Algorithms

There are several choices to make when using algorithms from Boost.Graph:

- Core vs. specialized algorithms
- Named vs. non-named parameters
- Internal vs. external properties
  - Internal: Lists vs. bundled

Non-named parameters

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
queue<graph::vertex_descriptor> q;
// null visitor which doesn't do anything
bfs_visitor<null_visitor> vis;
// graph has 4 vertices – colormap needs 4 elements
std::array<default_color_type, 4> colormap;

// pointer is automatically used as a property map
breadth_first_search(g, topLeft, q, vis, colormap.data());
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Named parameter

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

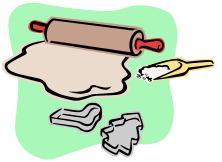
```
// null visitor which doesn't do anything
bfs_visitor<null_visitor> vis;

// visitor() gives the parameter a name
breadth_first_search(g, topLeft, visitor(vis));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

User-defined null visitor



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// my own null visitor which doesn't do anything
struct my_null_visitor
{
  typedef on_no_event event_filter;
  template <class T, class Graph>
  void operator()(T, Graph&) {}
};

// turns algorithm-independent visitor into a BFS visitor
bfs_visitor<my_null_visitor> vis;

breadth_first_search(g, topLeft, visitor(vis));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Discover visitor

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// prints 0, 1, 3, 2
struct my_discover_visitor
{
  typedef on_discover_vertex event_filter;
  template <class T, class Graph>
  void operator()(T t, Graph&) { std::cout << t << std::endl; }
} vis;

// make_bfs_visitor() helper function used
breadth_first_search(g, topLeft,
  visitor(make_bfs_visitor(vis)));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Discover visitor

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

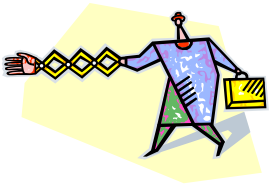
```
// prints 0, 1, 2, 3
struct my_discover_visitor
{
  typedef on_discover_vertex event_filter;
  template <class T, class Graph>
  void operator()(T t, Graph&) { std::cout << t << std::endl; }
} vis;

// make_dfs_visitor() helper function used
depth_first_search(g, visitor(make_dfs_visitor(vis)));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Recording distances



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
std::array<int, 4> distances = {{ 0 }};

// record_distances() to write distances to property map
breadth_first_search(g, topLeft,
  visitor(
    make_bfs_visitor(
      record_distances(distances.data(),
        on_tree_edge())));

// prints 0, 1, 2, 1
for (auto d : distances)
  std::cout << d << std::endl;
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Recording predecessors

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
std::array<int, 4> predecessors;
predecessors[bottomRight] = bottomRight;

// record_predecessors() to write predecessors to
// property map
breadth_first_search(g, bottomRight,
  visitor(
    make_bfs_visitor(
      record_predecessors(predecessors.data(),
        on_tree_edge()))));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Recording predecessors

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

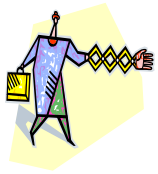
```
// prints 0, 1, 2
int p = topLeft;
while (p != bottomRight)
{
  std::cout << p << std::endl;
  p = predecessors[p];
}
std::cout << p << std::endl;
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now



Distances & predecessors

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
std::array<int, 4> distances = {{ 0 }};
std::array<int, 4> predecessors;
predecessors[bottomRight] = bottomRight;

breadth_first_search(g, bottomRight,
  visitor(
    make_bfs_visitor(
      std::make_pair(
        record_distances(distances.data(),
          on_tree_edge()),
        record_predecessors(predecessors.data(),
          on_tree_edge())))));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Property list

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// weight added as an edge property
typedef adjacency_list<vecS, vecS, undirectedS, no_property,
  property<edge_weight_t, int>> graph;

graph g(edges.begin(), edges.end(), 4);

std::array<int, 4> predecessors;
dijkstra_shortest_paths(g, bottomRight,
  predecessor_map(predecessors.data()));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

Property list

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
// weight added as an edge property
typedef adjacency_list<vecS, vecS, undirectedS, no_property,
  property<edge_weight_t, int>> graph;

// defining weights and initializing properties in the graph
std::array<int, 4> weights = {{ 2, 1, 1, 1 }};
graph g(edges.begin(), edges.end(), weights.data(), 4);

std::array<int, 4> predecessors;
dijkstra_shortest_paths(g, bottomRight,
  predecessor_map(predecessors.data()));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Property list

50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

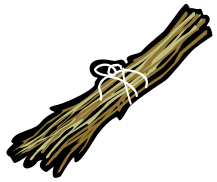
```
typedef adjacency_list<vecS, vecS, undirectedS, no_property,
  property<edge_weight_t, int>> graph;
graph g(edges.begin(), edges.end(), 4);

// accessing property map and setting weight per edge
property_map<graph, edge_weight_t>::type edge_weight_map =
  get(edge_weight_t(), g);
auto it = boost::edges(g).first;
put(edge_weight_map, *it, 2);
put(edge_weight_map, *++it, 1);
put(edge_weight_map, *++it, 1);
put(edge_weight_map, *++it, 1);
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Bundled properties



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

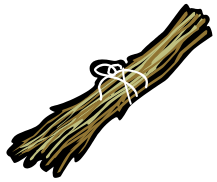
```
// bundled edge properties: simply a struct
struct edge_properties { int weight; };
typedef adjacency_list<vecS, vecS, undirectedS,
  no_property, edge_properties> graph;

std::array<edge_properties, 4> weights = {{ 2, 1, 1, 1 }};
graph g(edges.begin(), edges.end(), weights.data(), 4);

std::array<int, 4> predecessors;
dijkstra_shortest_paths(g, bottomRight,
  predecessor_map(predecessors.data()).weight_map(
    get(&edge_properties::weight, g)));
```

100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

Bundled properties



50BoostLibraries - Microsoft Visual C++ 2010 Express

File Edit View Project Debug Tools Window Help

main.cpp

(Global Scope)

```
struct edge_properties { int weight; };
typedef adjacency_list<vecS, vecS, undirectedS,
  no_property, edge_properties> graph;
graph g(edges.begin(), edges.end(), 4);

// accessing property map and setting weight per edge
auto it = edges(g).first;
g[*it].weight = 2;
g[*++it].weight = 1;
g[*++it].weight = 1;
g[*++it].weight = 1;
```

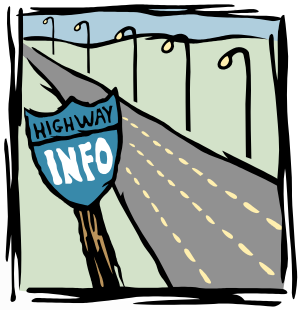
100 %

Error List Output Find Symbol Results

Ready Ln 1 Col 1 Ch 1 INS

C++ now

More information



- Boost.Graph documentation:

  http://www.boost.org/libs/graph/

- Books:

  The Boost Graph Library: User Guide and Reference Manual (from 2001)