BOOST

Quoc-Tuan Le

Optimal seminar group, HUS

Overview

Introduction to BOOST

What is BOOST

- Boost provides free peer-reviewed portable C++ source libraries that work well with the C++ Standard Library.

What is BOOST

- Boost provides free peer-reviewed portable C++ source libraries that work well with the C++ Standard Library.
- Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

What is BOOST

- Boost provides free peer-reviewed portable C++ source libraries that work well with the C++ Standard Library.
- Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.
- Boost works on almost any modern operating system, including UNIX and Windows variants.

What is BOOST

- Boost provides free peer-reviewed portable C++ source libraries that work well with the C++ Standard Library.
- Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.
- Boost works on almost any modern operating system, including UNIX and Windows variants.
- Boost welcomes and thrives on participation from a variety of individuals and organizations. Many avenues for participation are available in the Boost Community.

Get Boost

The most reliable way to get a copy of Boost is to download a distribution from SourceForge:

1. Download boost_1_67_0.tar.bz2.
2. In the directory where you want to put the Boost installation, execute

```
tar --bzip2 -xf /path/to/boost\_1\_67\_0.tar.bz2
```

Header-Only Libraries

The first thing many people want to know is, "how do I build Boost?" The good news is that often, there's nothing to build.

Nothing to Build?

Most Boost libraries are header-only: they consist entirely of header files containing templates and inline functions, and require no separately-compiled library binaries or special treatment when linking.

Build a Simple Program Using Boost

The following program reads a sequence of integers from standard input, uses Boost.Lambda to multiply each number by three, and writes them to standard output:

```
1  #include <boost/lambda/lambda.hpp>
2  #include <iostream>
3  #include <iterator>
4  #include <algorithm>
5
6  int main()
7  {
8      using namespace boost::lambda;
9      typedef std::istream_iterator<int> in;
10
11     std::for_each(
12     in(std::cin), in(), std::cout << (_1 * 3) << " " );
13 }
```

Build a Simple Program Using Boost

Copy the text of this program into a file called example.cpp.

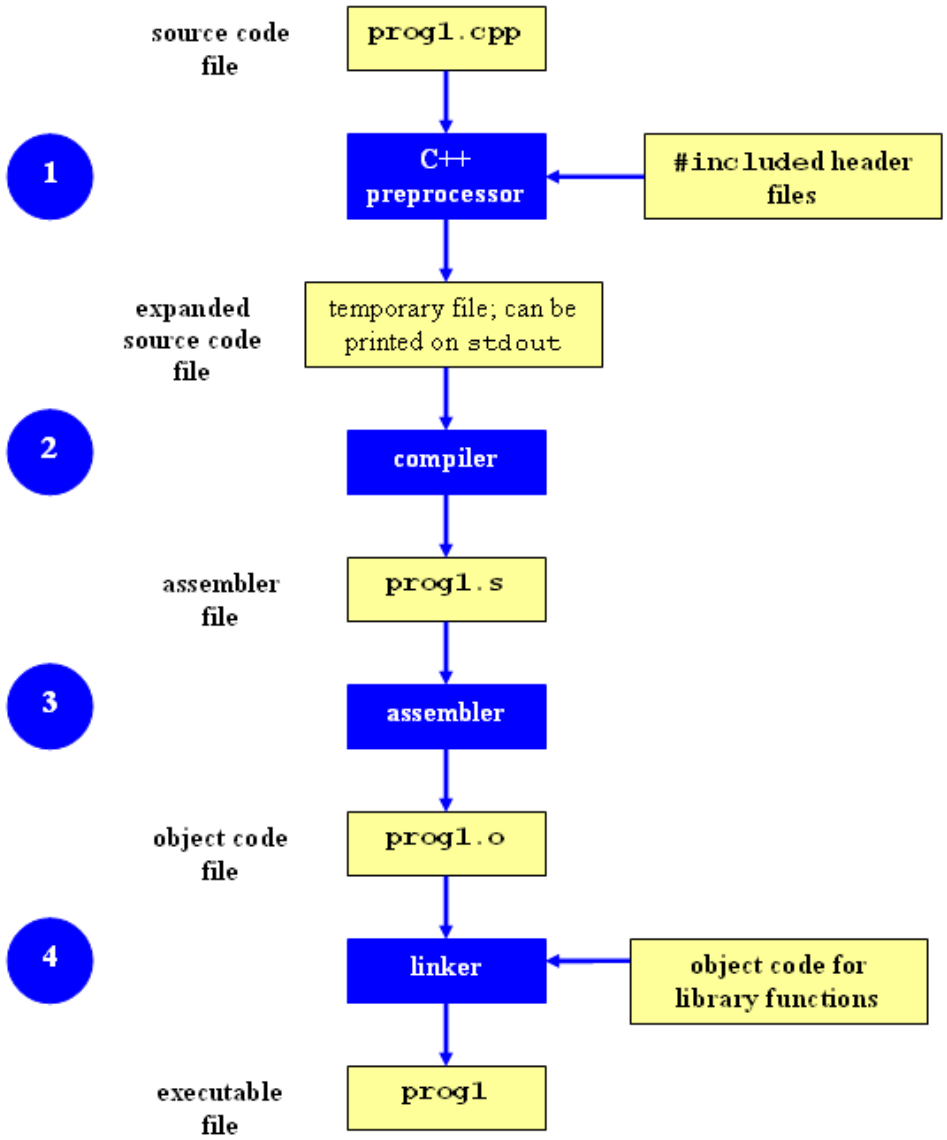
Now, in the directory where you saved example.cpp, issue the following command:

```
g++ -I path/to/boost_1_67_0 example.cpp -o example
```

To test the result, type:

```
echo 1 2 3 | ./example
```

Prepare to Use a Boost Library Binary



Prepare to Use a Boost Library Binary

If you want to use any of the separately-compiled Boost libraries, you'll need to acquire library binaries.

Issue the following commands in the shell:

```
cd path/to/boost_1_67_0
./bootstrap.sh --help
```

Select your configuration options and invoke ./bootstrap.sh again without the –help option. Unless you have write permission in your system's /usr/local/ directory, you'll probably want to at least use

```
./bootstrap.sh --prefix=path/to/installation/prefix
```

to install somewhere else. Also, consider using the –show-libraries and –with-libraries=library-name-list options to limit the long wait you'll experience if you build everything. Finally,

```
./b2 install
```

Link Your Program to a Boost Library

we'll use the following simple program that extracts the subject lines from emails. It uses the Boost.Regex library, which has a separately-compiled binary component.

```
1  #include <boost/regex.hpp>
2  #include <iostream>
3  #include <string>
4
5  int main(){
6      std::string line;
7      boost::regex pat( "^Subject: (Re: |Aw: )*(.*)" );
8
9      while (std::cin){
10         std::getline(std::cin, line);
11         boost::smatch matches;
12         if (boost::regex_match(line, matches, pat))
13         std::cout << matches[2] << std::endl;
14     }
15 }
```

Link Your Program to a Boost Library

There are two main ways to link to libraries:

1. You can specify the full path to each library:

```
g++ -I path/to/boost_1_67_0 example.cpp -o example \
~/boost/stage/lib/libboost_regex-gcc34-mt-d-1_36.a
```

2. You can separately specify a directory to search (with -Ldirectory) and a library name to search for (with -llibrary,2 dropping the filename's leading lib and trailing suffix (.a in this case):

```
g++ -I path/to/boost_1_67_0 example.cpp -o example \
-L~/boost/stage/lib/ -lboost_regex-gcc34-mt-d-1_36
```

BOOST's performance

Performance[1]

Graph type	Algorithm	Sparse graph	Dense graph
LEMON	LEMON	3.27s	1.13s
LEMON	BGL	4.36s	1.07s
BGL	LEMON	3.55s	1.56s
BGL	BGL	4.90s	2.08s

Table 1: Benchmark results for the largest instances of the shortest path problem combining LEMON and BGL implementations.

[1]The benchmark tests were performed on a machine with AMD Opteron Dual Core 2.2 GHz CPU and 16 GB memory (1 MB cache), running openSUSE 10.1 operating system. The codes were compiled with GCC version 4.1.0 using -O3 optimization flag.

Heap performance

Type \ n	10	100	1000	10000	100000
BinHeap	0.0001065	0.00076785	0.0084887	0.0862004	1.05576
Dheap	9.975e-05	0.0006841	0.0082312	0.0861992	1.05127
FibHeap	0.00011345	0.000767	0.0073001	0.0875208	1.05497

Table 2: Results for the Dijkstra algorithm (one to all) compiling with BOOST heap options.

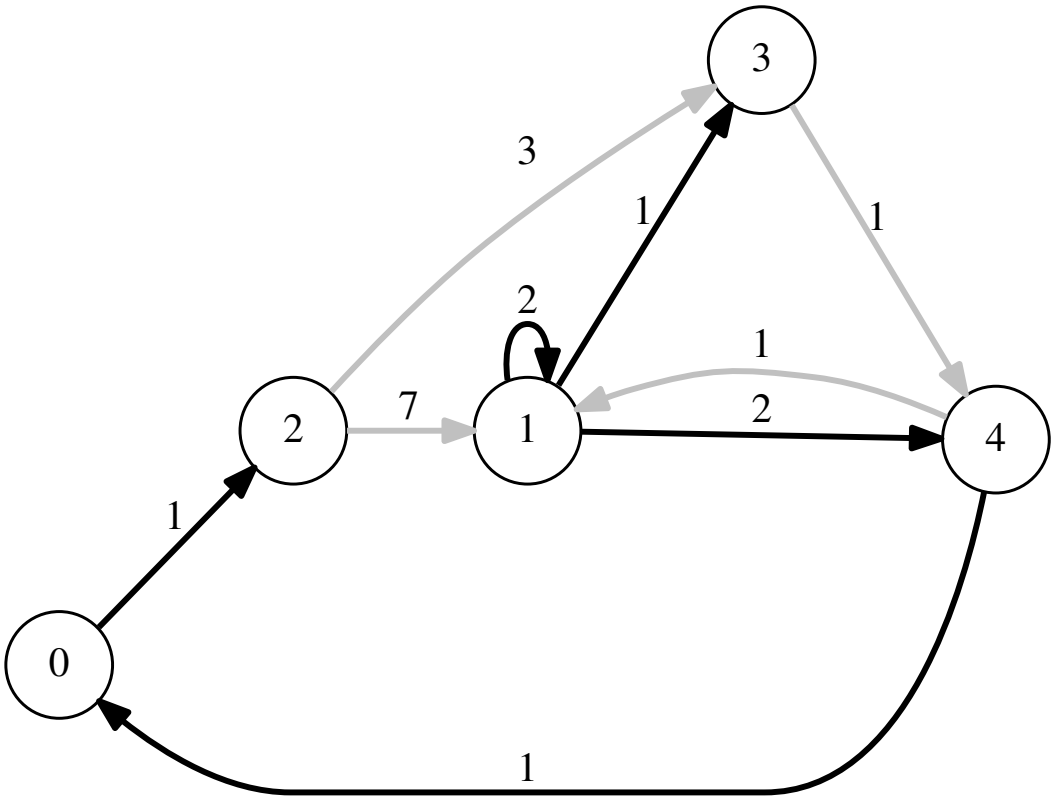
Heap performance

Type \ n	10	100	1000	10000	100000
BinHeap	0.0001766	0.0010599	0.0097628	0.123566	1.46321
Dheap	0.0001497	0.00069505	0.00607185	0.0729028	0.819103

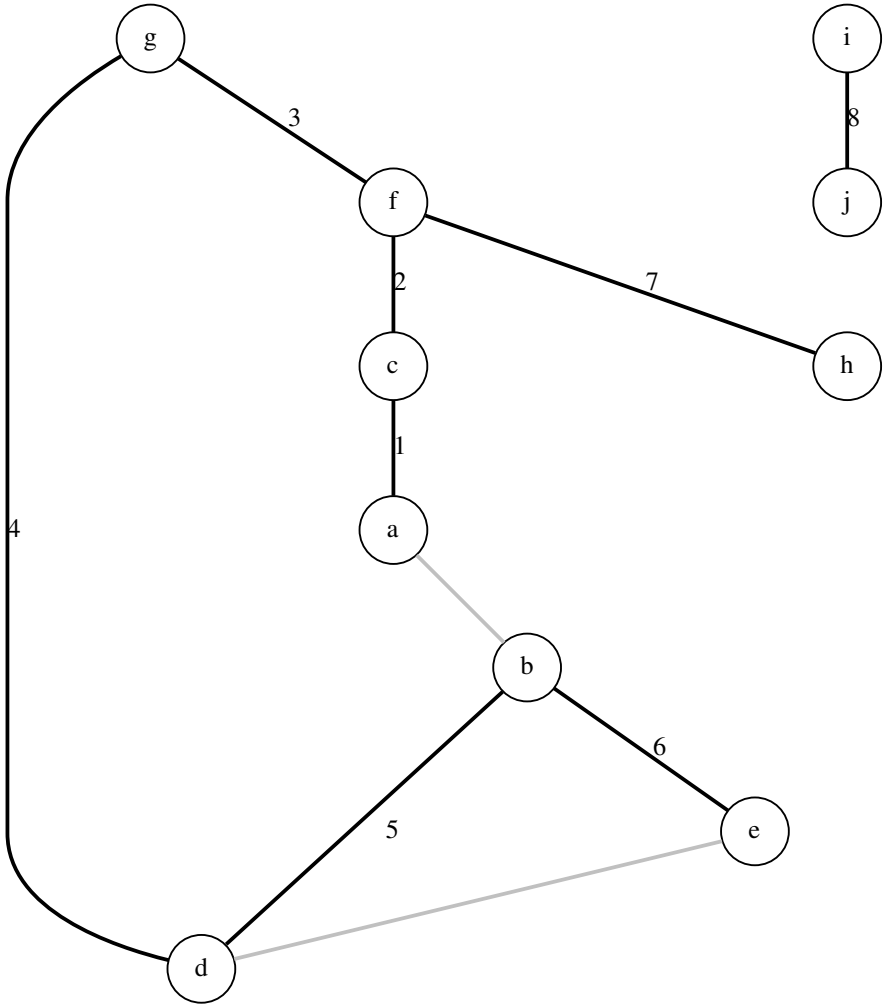
Table 3: Results for the Dijkstra algorithm (one to all) compiling with BOOST heap options.

BOOST's graphic

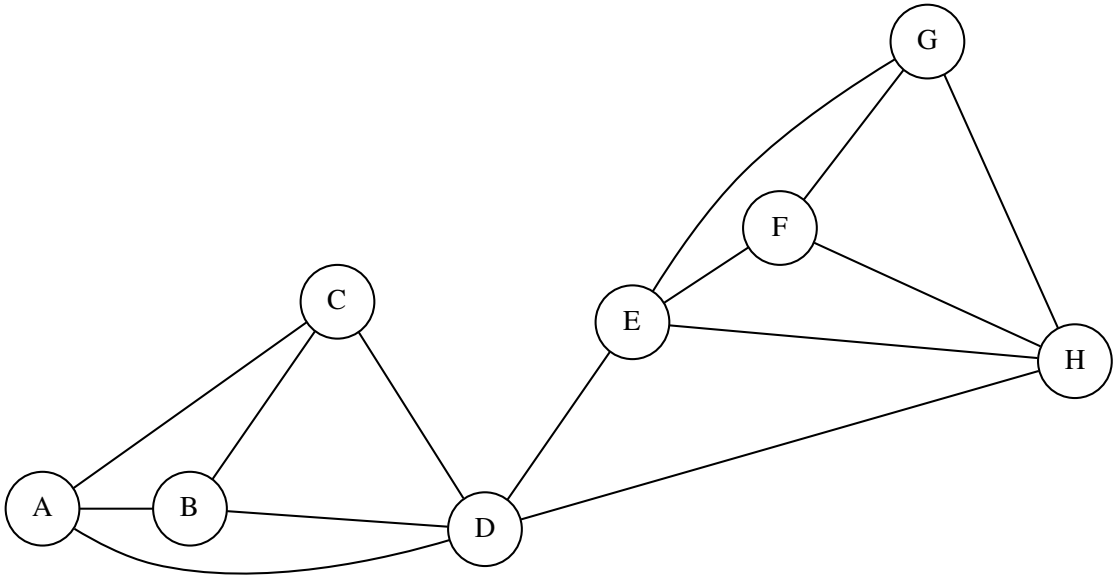
Graphic



Graphic



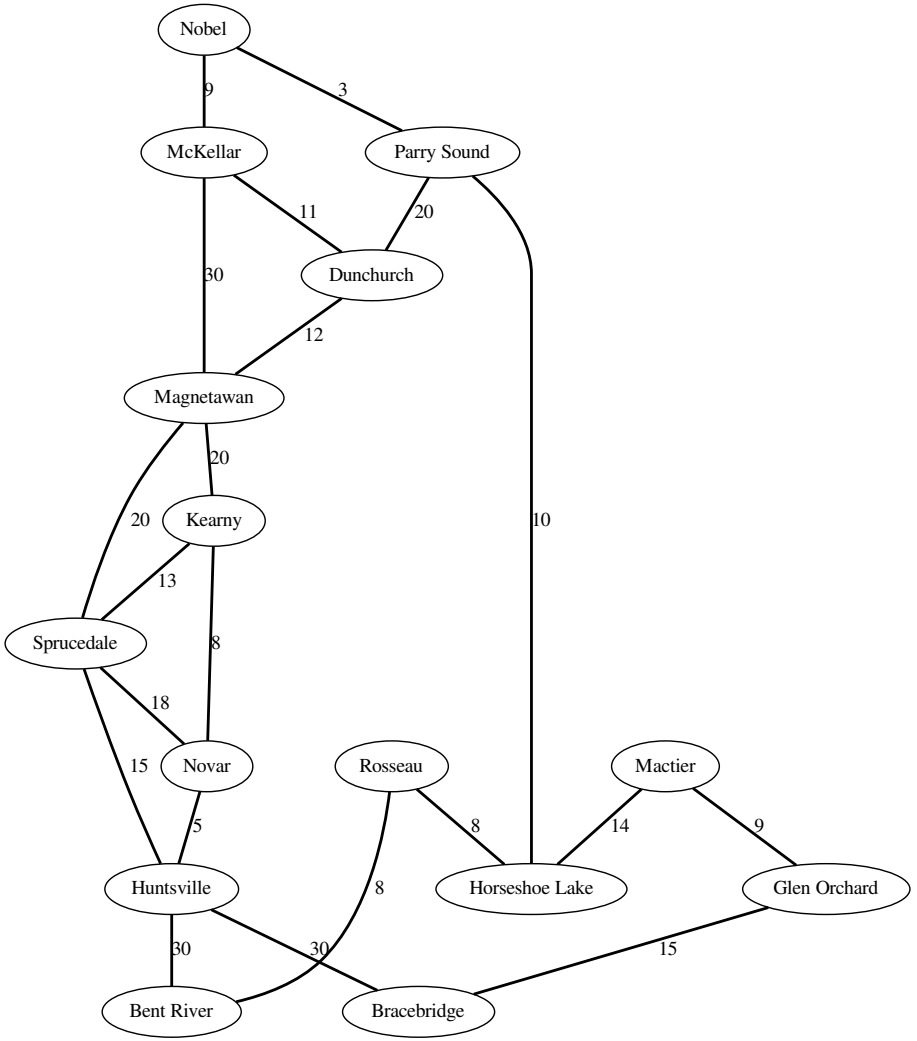
Graphic



Graphic


www.boost.org
anubis.dkuug.dk
sourceforge.net
www.lsc.nd.edu
www.hp.com
www.yahoogroups.com
www.lam-mpi.org
weather.yahoo.com
nytimes.com
www.boston.com

Graphic


Nobel
9
3
McKellar
Parry Sound
11
20
30
Dunchurch
12
Magnetawan
20
20
Kearny
10
13
Sprucedale
8
18
15
Novar
Rosseau
Mactier
5
8
14
9
Huntsville
8
Horseshoe Lake
Glen Orchard
30
30
15
Bent River
Bracebridge