Guide to Matchbox-10.12

For Linux Users

Step 1

Go to the matchbox root directory and run "make" for the debug version or "make optimized" for the optimized one. Executables are generated in /examples/debug or /examples/optimized.

For Mac Users

Step 1

Install llvm with brew: brew install llvm

In Makefile.gnu_32:

Change all g++ to clang++.

Remove –lrt flags.

Step 2

Go to the matchbox root directory and run "make" for the debug version or "make optimized" for the optimized one. Executables are generated in /examples/debug or /examples/optimized.

Possible Compiling Issues and Solutions

1. unsupported option '-fopenmp' / '-lrt'

Solution: Step 1 above.

2. ld: library not found for -lomp

Solution:

Get version number of llvm: llvm-config –-version

Create a link for libomp.dylib:

cd /usr/local/lib

ln -s /usr/local/Cellar/llvm/<version number>/lib/libomp.dylib libomp.dylib

3. 'wchar.h' file not found

Solution:

Before running "make", run the following command first:

export
CPLUS_INCLUDE_PATH="/Applications/Xcode.app/Contents/Developer/Platforms
/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include

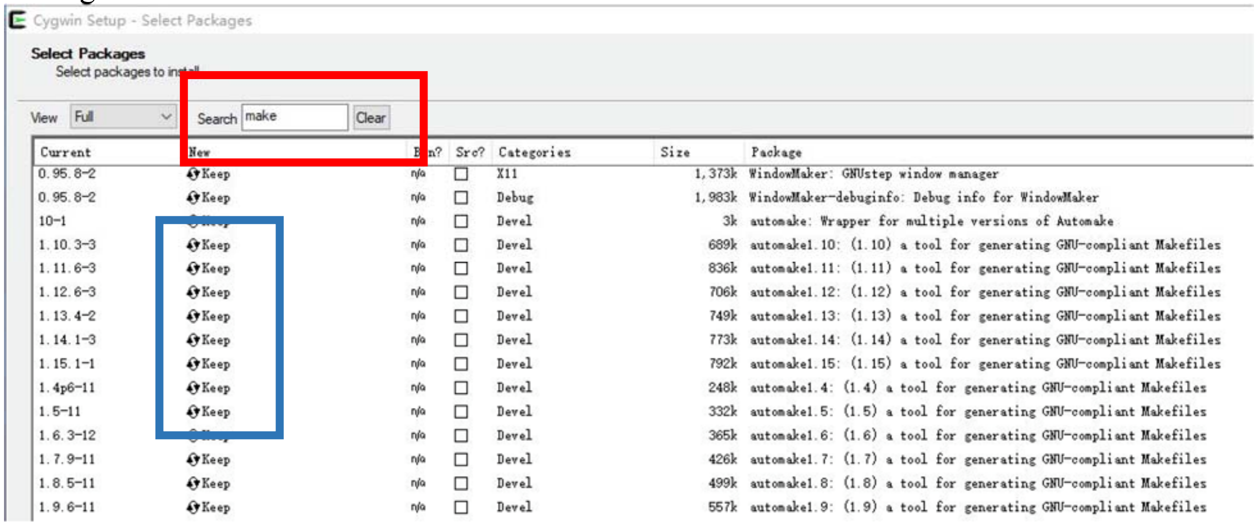
For Windows Users

Step 1

Download the latest version of Cygwin from http://www.cygwin.com.

Step 2

During installation:



Cygwin Setup - Select Packages

Select Packages
Select packages to install

View: Full   Search: make   Clear

Current	New	Bin?	Src?	Categories	Size	Package
0.95.8-2	Keep	n/a	☐	X11	1,373k	WindowMaker: GNUstep window manager
0.95.8-2	Keep	n/a	☐	Debug	1,983k	WindowMaker-debuginfo: Debug info for WindowMaker
10-1	Keep	n/a	☐	Devel	3k	automake: Wrapper for multiple versions of Automake
1.10.3-3	Keep	n/a	☐	Devel	689k	automake1.10: (1.10) a tool for generating GNU-compliant Makefiles
1.11.6-3	Keep	n/a	☐	Devel	836k	automake1.11: (1.11) a tool for generating GNU-compliant Makefiles
1.12.6-3	Keep	n/a	☐	Devel	706k	automake1.12: (1.12) a tool for generating GNU-compliant Makefiles
1.13.4-2	Keep	n/a	☐	Devel	749k	automake1.13: (1.13) a tool for generating GNU-compliant Makefiles
1.14.1-3	Keep	n/a	☐	Devel	773k	automake1.14: (1.14) a tool for generating GNU-compliant Makefiles
1.15.1-1	Keep	n/a	☐	Devel	792k	automake1.15: (1.15) a tool for generating GNU-compliant Makefiles
1.4p6-11	Keep	n/a	☐	Devel	248k	automake1.4: (1.4) a tool for generating GNU-compliant Makefiles
1.5-11	Keep	n/a	☐	Devel	332k	automake1.5: (1.5) a tool for generating GNU-compliant Makefiles
1.6.3-12	Keep	n/a	☐	Devel	365k	automake1.6: (1.6) a tool for generating GNU-compliant Makefiles
1.7.9-11	Keep	n/a	☐	Devel	426k	automake1.7: (1.7) a tool for generating GNU-compliant Makefiles
1.8.5-11	Keep	n/a	☐	Devel	499k	automake1.8: (1.8) a tool for generating GNU-compliant Makefiles
1.9.6-11	Keep	n/a	☐	Devel	557k	automake1.9: (1.9) a tool for generating GNU-compliant Makefiles

View full list of packages. Search for ‘make’ and ‘vim’ packages (shown in red). If it says ‘Skip’ for the packages (shown in blue), select them and install them. Installed packages should have status ‘Keep’ instead of ‘Skip’. By default, the installer does not install these packages. The installer can be used to add packages to the existing installation.

Step 3

Run ‘vim [filename]’ to open the file. Press ‘i’ to turn on editing mode., ‘esc’ to exit editing mode. Enter ‘:w’ to save changes, ‘:q’ to exit the file or ‘:wq’ save and exit the file.

Go to the matchbox root directory and run "make" for the debug version or "make optimized" for the optimized one. Executables are generated in /examples/debug or /examples/optimized.

Notes

1. To run

Go to examples/debug or examples/optimized and run an executable file without arguments you will see what the arguments need to be. For example run ./BptMaxCard.

2. Input

-The graph file must be in mtx format and for non-bipartite algorithm the mtx file must be symmetric.

-Vertex weights file must be in the following format:

```
n
w(v1)
w(v2)
.
.
w(vn)
```

Where n is the number of vertices and w(vi) is the weight of the i-th vertex.

3. List of implemented algorithms

I. Cardinality matching algorithms

Algorithms for bipartite graphs

BptMaxCard: Exact algorithm for maximum cardinality matching.

BptAprxMaxCard: 1/2-appximation algorithm for maximum cardinality matching.

Algorithms for non-bipartite graphs

MaxCard: Exact algorithm for maximum cardinality matching.

II. Edge weighted matching algorithms

Algorithms for bipartite graphs

BptMaxEdgWght: Exact algorithm for maximum edge-weighted matching.

BptPrfMaxEdgWght: Exact algorithm for maximum edge-weighted perfect matching.

BptSemiPrfMaxEdgWght: Exact algorithm for maximum edge-weighted semi-perfect matching, BptHalfEdgWght: 1/2-approximation algorithm for maximum edge-weighted matching, using Greedy Algorithm.

Algorithms for non-bipartite graphs

HalfEdgWght: 1/2-approximation algorithm for maximum edge-weighted, using Greedy Algorithm.

III. Vertex weighted matching algorithms

Algorithms for bipartite graphs

Note: All bipartite vertex weighted matching problems are solved as two single set weighted sub problems and combine them using Mendelsohn-Dulmage technique.

BptMaxVtxWght: Exact algorithm for maximum vertex-weighted matching.

BptTwoThirdVtxWght: 2/3-approximation algorithm for maximum vertex-weighted, using sorting vertices in non-increasing order of weights and augmenting paths of length at most three that reach a heaviest vertex.

BptHalfVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Greedy Algorithm.

BptLocalVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Locally Dominant Algorithm.

BptSuitorVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Suitor Algorithm.

Algorithms for non-bipartite graphs

MaxVtxWght: Exact algorithm for maximum vertex-weighted matching.

ScaleApproxVtxWght: (1-eps)-approximation algorithm for maximum vertex-weighted, using Scaling Algorithm.

TwoThirdVtxWght: 2/3-approximation algorithm for maximum vertex-weighted, using sorting vertices in non-increasing order of weights and augmenting paths of length at most three that reach a heaviest vertex..

HalfVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Greedy Algorithm.

SuitorVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Suitor Algorithm.

ParSuitorVtxWght: OpenMP implementation of 1/2-approximation algorithm for maximum vertex-weighted, using Suitor Algorithm.
LocalVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Locally Dominant Algorithm.
PGDPVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Path Growing Algorithm with dynamic programming.
PGVtxWght: 1/2-approximation algorithm for maximum vertex-weighted, using Path Growing Algorithm.