CS224W Recitation: A Tutorial of SNAP

Chenguang Zhu

What is SNAP?

- **S**tanford **N**etwork **A**nalysis **P**roject (SNAP)
- A network analysis and graph mining library
- C++ based
- Manipulates large graphs, calculates structural properties, generates graphs, and supports attributes on nodes and edges
- More info on http://snap.stanford.edu

Content

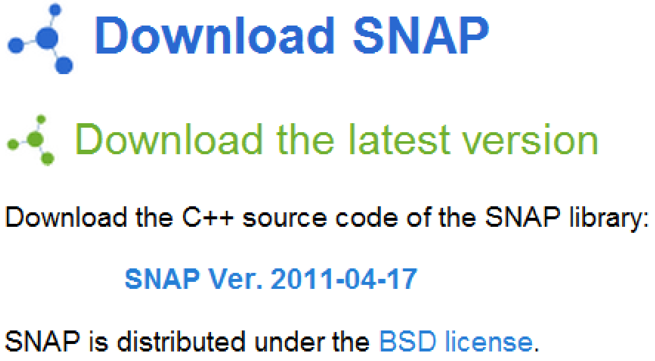
- Installation
- Data structures in SNAP
- Graph manipulation in SNAP
- Datasets in SNAP
- Plotting in SNAP
- Q&A

Content

- **Installation**
- Data structures in SNAP
- Graph manipulation in SNAP
- Datasets in SNAP
- Plotting in SNAP
- Q&A

Installation

- 1. Go to http://snap.stanford.edu/snap/download.html



Download SNAP

Download the latest version

Download the C++ source code of the SNAP library:

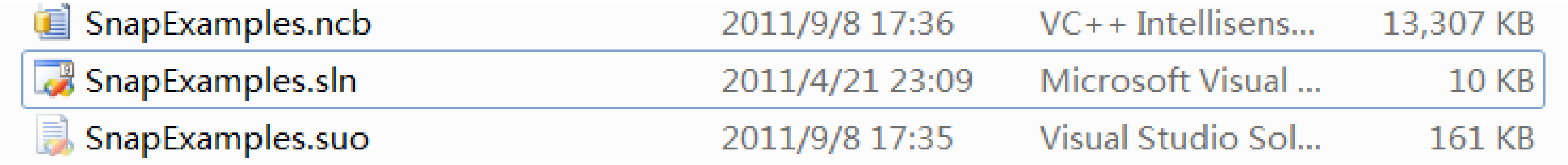
SNAP Ver. 2011-04-17

SNAP is distributed under the BSD license.

- 2. Download the latest SNAP version

Installation

- 3. Unzip
- 4. Go to subfolder “examples”
- 5. Open project “SnapExamples.sln” (Visual Studio required)



SnapExamples.ncb	2011/9/8 17:36	VC++ Intellisens...	13,307 KB
SnapExamples.sln	2011/4/21 23:09	Microsoft Visual ...	10 KB
SnapExamples.suo	2011/9/8 17:35	Visual Studio Sol...	161 KB

Installation

- If your system is Linux-based, use the Makefile in the same folder

- You can refer to any Makefile in folders in "*examples*", e.g. *examples/kronfit/Makefile*

SNAP under Linux

```
## Linux  (uncomment the 2 lines below for compilation on Linux)
#CXXFLAGS += -std=c++98 -Wall
#LDFLAGS += -lrt

## CygWin (uncomment the 2 lines below for compilation on CygWin)
#CXXFLAGS += -Wall
#LDFLAGS +=

## Main application file
MAIN = kronfit

all: $(MAIN)

opt: CXXFLAGS += -O4
opt: LDFLAGS += -O4
opt: $(MAIN)

# COMPILE
$(MAIN): $(MAIN).cpp Snap.o
	g++ $(LDFLAGS) -o $(MAIN) $(MAIN).cpp ../../snap/kronecker.cpp Snap.o -I../../glib -I../../snap

Snap.o:
	g++ -c $(CXXFLAGS)  ../../snap/Snap.cpp -I../../glib -I../../snap

clean:
	rm -f *.o  $(MAIN)  $(MAIN).exe
	rm -rf Debug Release
```

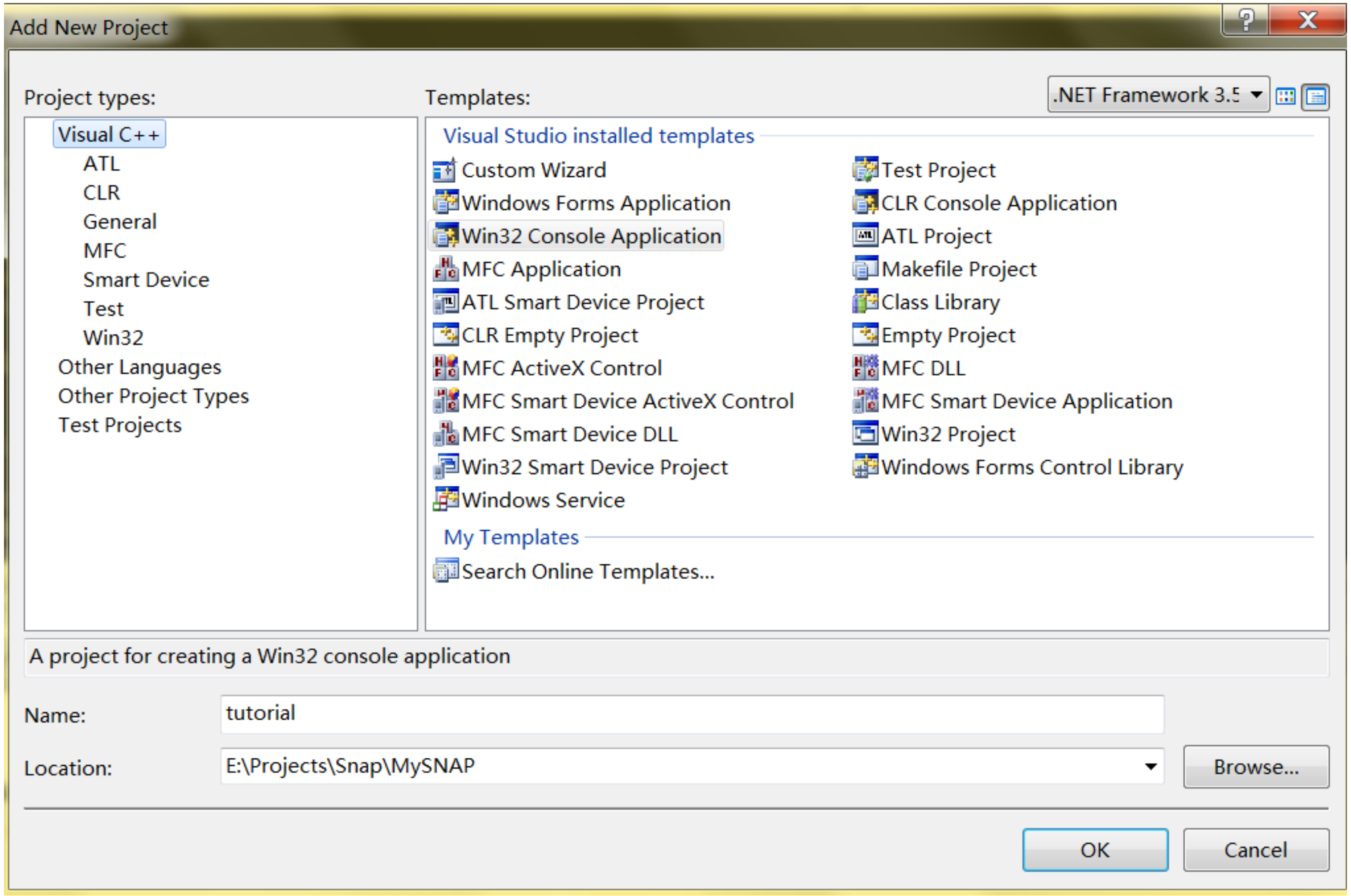
SNAP under Linux

```
## Linux  (uncomment the 2 lines below for compilation on Linux)
#CXXFLAGS += -std=c++98 -Wall
#LDFLAGS += -lrt

## CygWin (uncomment the 2 lines below for compilation on CygWin)
#CXXFLAGS += -Wall
#LDFLAGS +=

## Main application file
MAIN = kronfit

all: $(MAIN)

opt: CXXFLAGS += -O4
opt: LDFLAGS += -O4
opt: $(MAIN)

# COMPILE
$(MAIN): $(MAIN).cpp Snap.o
	g++ $(LDFLAGS) -o $(MAIN) $(MAIN).cpp ../../snap/kronecker.cpp Snap.o -I../../glib -I../../snap

Snap.o:
	g++ -c $(CXXFLAGS) ../../snap/Snap.cpp -I../../glib -I../../snap

clean:
	rm -f *.o  $(MAIN)  $(MAIN).exe
	rm -rf Debug Release
```

Demo: Make in LINUX

Create Your Own Project

- Open Visual Studio and create a project



Add New Project

Project types:

- Visual C++
  - ATL
  - CLR
  - General
  - MFC
  - Smart Device
  - Test
  - Win32
- Other Languages
- Other Project Types
- Test Projects

Templates: .NET Framework 3.5

Visual Studio installed templates

Custom Wizard	Test Project
Windows Forms Application	CLR Console Application
Win32 Console Application	ATL Project
MFC Application	Makefile Project
ATL Smart Device Project	Class Library
CLR Empty Project	Empty Project
MFC ActiveX Control	MFC DLL
MFC Smart Device ActiveX Control	MFC Smart Device Application
MFC Smart Device DLL	Win32 Project
Win32 Smart Device Project	Windows Forms Control Library
Windows Service	

My Templates

Search Online Templates...

A project for creating a Win32 console application

Name: tutorial

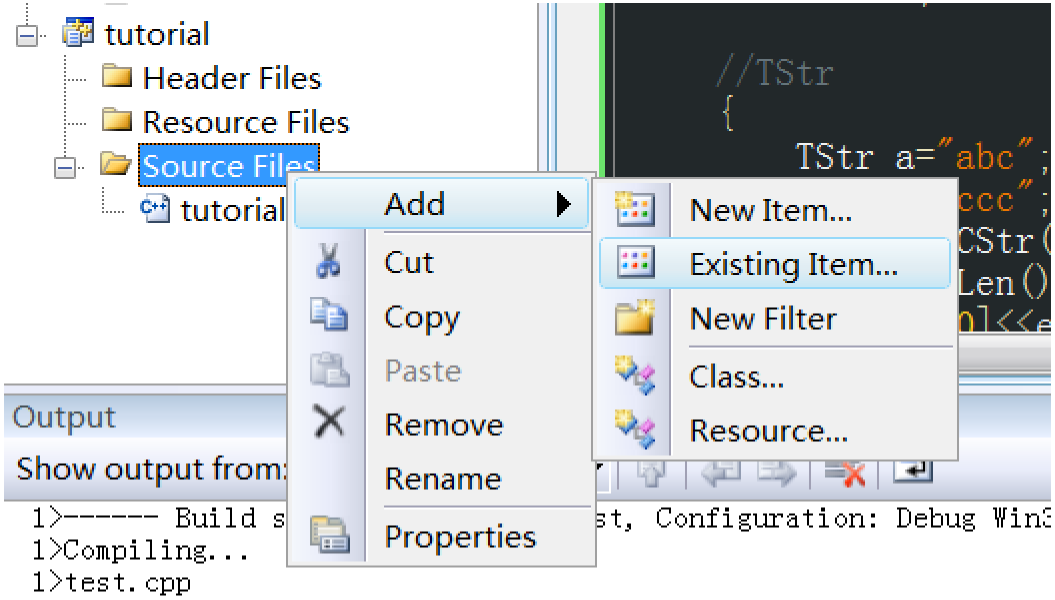
Location: E:\Projects\Snap\MySNAP

Create Your Own Project

- Add line "#include YOUR_SNAP_PATH/snap/Snap.h" into your main program

```
#include <stdio.h>
#include "../../snap/Snap.h"
…
```

- Include YOUR_SNAP_PATH/snap/Snap.h/cpp into your project

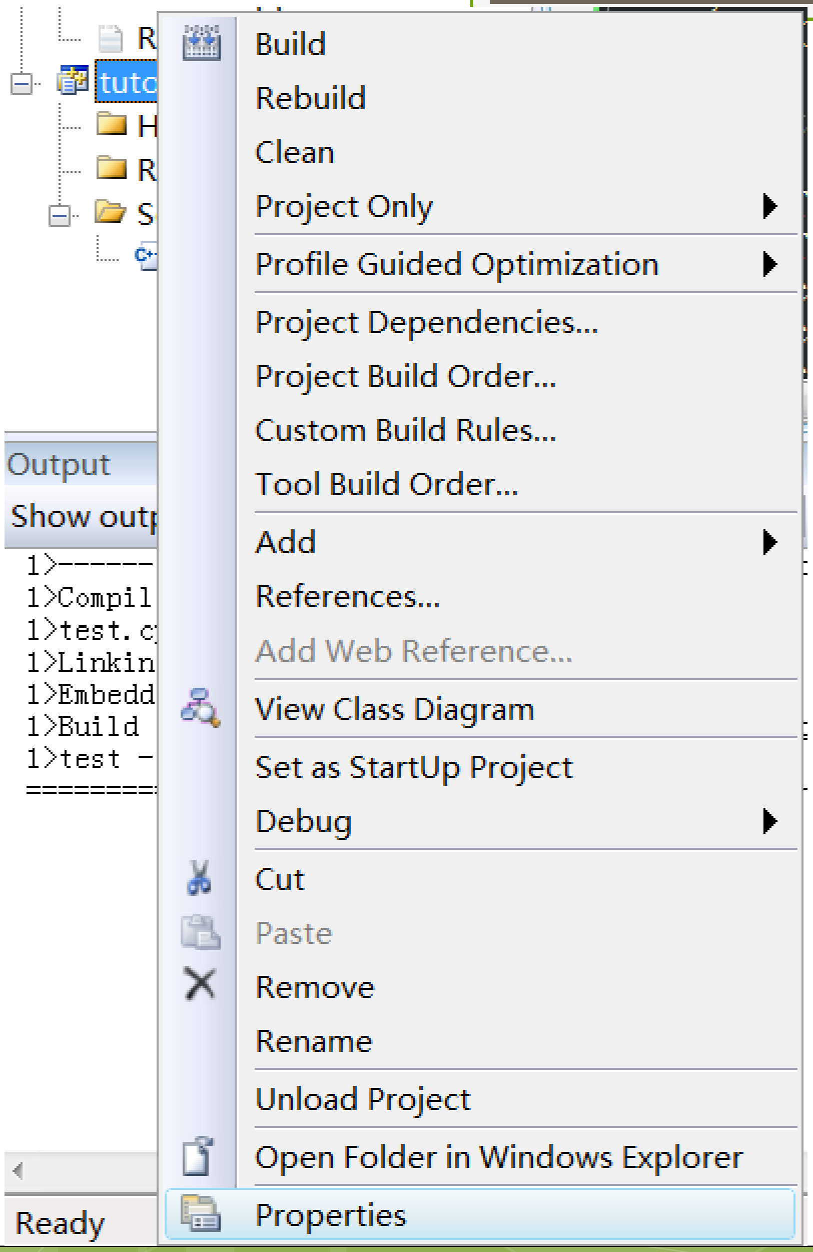


```
//TStr
{
    TStr a="abc";
          ccc";
          CStr(
          Len()
          nl<<e
```

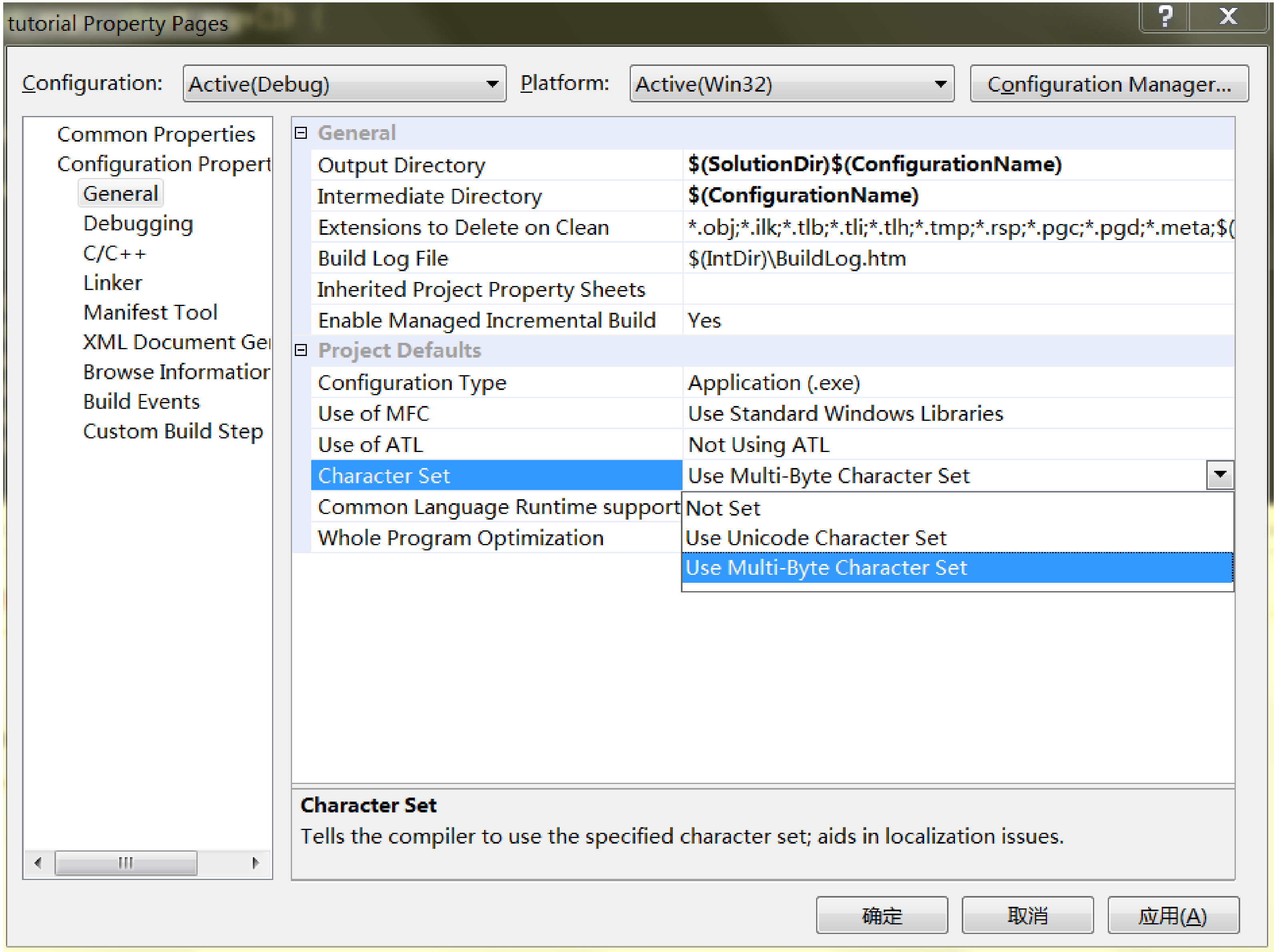
```
1>------ Build s                    st, Configuration: Debug Win3
1>Compiling...
1>test.cpp
```

Create Your Own Project

- Due to the settings of SNAP, the character set must be set to Multi-byte
  - Right-click on your project and go to "Properties"
  - Go to *Configuration Properties* → *General* → *Projects Defaults* → *Character Set* → *Select "Use Multi-Byte Character Set"*



```
1>------
1>Compil
1>test.c
1>Linkin
1>Embedd
1>Build
1>test -
========
```



Configuration: Active(Debug)

Platform: Active(Win32)

General	
Output Directory	**$(SolutionDir)$(ConfigurationName)**
Intermediate Directory	**$(ConfigurationName)**
Extensions to Delete on Clean	*.obj;*.ilk;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;*.pgc;*.pgd;*.meta;$(
Build Log File	$(IntDir)\BuildLog.htm
Inherited Project Property Sheets	
Enable Managed Incremental Build	Yes
Project Defaults	
Configuration Type	Application (.exe)
Use of MFC	Use Standard Windows Libraries
Use of ATL	Not Using ATL
Character Set	Use Multi-Byte Character Set
Common Language Runtime support	
Whole Program Optimization	

Character Set

Tells the compiler to use the specified character set; aids in localization issues.

Create Your Own Project

- Now you are free to go!
- Program whatever your want, and enjoy the powerful arsenal of SNAP!

Content

- Installation
- **Data structures in SNAP**
- Graph manipulation in SNAP
- Datasets in SNAP
- Plotting in SNAP
- Q&A

What’s In SNAP?

- **Data structures (In subfolder “glib”):**
  - STL-like library
  - Contains basic data structures, like vectors, hash-tables and strings
  - Provides serialization for loading and saving
- Network analysis library (In subfolder “snap”)
  - Network generation, manipulation
- Example applications (In subfolder “examples”)
  - Small sample applications that demonstrate functionality

Data Structures

- In subfolder “glib”
- More info in glib/ds.h
- Numbers:
  - Integers: TInt
  - Real number: TFlt
  - Example:
    - TInt a=5; cout<<a<<endl;
    - Note: in C style, use printf(“%d\n”, a.Val);

Basic Structures

- String: TStr
  - Examples:
    - TStr a="abc";
    - TStr b="ccc";
    - cout<<a.CStr()<<endl; (char*) --- abc
    - cout<<a.Len()<<endl; --- 3
    - cout<<a[0]<<endl; --- a
    - cout<<(a==b)<<endl; --- 0

Combination

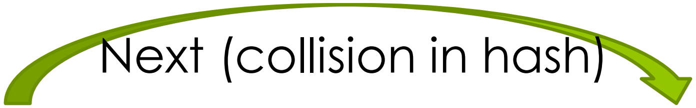
- Pair
  - TPair<Type1, Type2> (Type can also be complex structures like TVec, TPair…)
    - E.g. TPair<TInt, TFlt> a; a.Val1=…; a.Val2=…;
  - List of shorthand (in ds.h)
    - typedef TPair<TInt, TInt> TIntPr;
    - typedef TPair<TInt, TIntPr> TIntIntPrPr;
- Triple
  - TTriple<Type1, Type2, Type3>

Vectors

- TVec<Type>
  - Example:
    - TVec<TInt> a;
    - a.Add(10);
    - a.Add(20);
    - a.Add(30);
    - cout<<a[0]<<endl; --- 10
    - cout<<a.Len()<<endl; --- 3
  - Similarly, “Type” can be complex structures like TVec< TVec< TVec<TFlt> > >

Hash Tables

- THash<key type, value type>
  - Key is the unique index, value is associated with key



Next (collision in hash)

KeyId	0	1	2
Key	“David”	“Ann”	“Jason”
Value	100	89	95

Hash Tables

- Example:

```
THash<TInt, TStr> a;
a.AddDat(12)=“abc”;
a.AddDat(34)=“def”;
cout<<a.GetKey(0)<<endl;             ----  12
cout<<a[0].CStr()<<endl;             ----  abc
cout<<a.GetKeyId(12)<<endl;          ---- 0
cout<<a.GetDat(34).CStr()<<endl;    ----- def
```

Hash Tables

- When key is of string type: THash<TStr, …>, a more space-efficient way is to use TStrHash<…>
  - Example: TStrHash<TInt>

- Uses string pool, saves more space

Hash Sets

- In case only key is needed, use THashSet
- Example:
  - THashSet<TInt> a;
  - a.AddKey(12);
  - a.AddKey(34);
  - a.AddKey(56);
  - cout<<a.GetKey(2)<<endl;   --- 56

Saving and Loading

- Binary files
- Much quicker to save/load
- Memory efficient
- Save:
  - {TFOut fout("a.bin");
  - a.Save(fout);}
- Load:
  - {TFIn fin("a.bin");
  - a.Load(fin);}

Useful Data Structure(1): Time

- TSecTm
- Manipulates time
- Supports comparison, calculation in different time units, obtaining current time…
- **DEMO: TSecTm_example.cpp**

Useful Data Structure(2): Generate Distribution

- TRnd class
- Generate lots of distributions
- Example:

```
 TRnd a;
//exponential distribution
for (int i=0; i<10; ++i)
         cout<<a.GetExpDev(1)<<endl;
```

- **DEMO: TRnd_example.cpp**

Useful Data Structure(3): Calculating Statistics

- In glib/xmath.h
- Multiple classes
- Calculating moments, correlation coefficients, t-test …
- **Demo: XMath_example.cpp**

Content

- Installation
- Data structures in SNAP
- **Graph manipulation in SNAP**
- Datasets in SNAP
- Plotting in SNAP
- Q&A

What's In SNAP?

- Data structures (In subfolder "glib"):
  - STL-like library
  - Contains basic data structures, like vectors, hash-tables and strings
  - Provides serialization for loading and saving
- **Network analysis library (In subfolder "snap")**
  - Network generation, manipulation
- Example applications (In subfolder "examples")
  - Small sample applications that demonstrate functionality

Graph Type

- **TUNGraph**: undirected graph with no multi-edge
- **TNGraph**: directed graph with no multi-edge
- **TNEGraph**: directed graph with multi-edge

Network Type

- **TNodeNet<TNodeData>:** directed graph with TNodeData object for each node
- **TNodeEDatNet<TNodeData, TEdgeData>:** directed graph with TNodeData on each node and TEdgeData on each edge
- **TNodeEdgeNet<TNodeData, TEdgeData>:** directed multi-edge graph with TNodeData on each node and TEdgeData on each edge

Network Type

- **TNodeNet<TNodeData>:** directed graph with TNodeData object for each node
- **TNodeEDatNet<TNodeData, TEdgeData>:** directed graph Data on each node and dge
- **TNodeE Data>:** directe TNodeD TEdgeDa

When you wanna use saving/loading function, you have to write Save and Load

Demo: NodeNet.cpp

Example

Smart pointer: Count the number of pointers to an object. Release things automatically when the count→0

- Use smart pointer whenever possible
- ***typedef TPt<TNGraph> PNGraph***
- Add node before edges
- Example:
  - PNGraph Graph = TNGraph::New();
  - Graph->AddNode(1);
  - Graph->AddNode(5);
  - Graph->AddEdge(1,5);

Example

- Use smart pointer whenever possible
- ***typedef TPt<TNGraph> PNGraph***
- Add node before edges
- Example:
    - PNGraph Graph = TNGraph::N
    - Graph->AddNode(1);
    - Graph->AddNode(5);
    - Graph->AddEdge(1,5);

Demo: Gnm.cpp

Establish A Graph

- Generate graph with specific properties
- Use TSnap::Gen…
  - TSnap::GenRndGnm (G$_{nm}$ (Erdős–Rényi) graph)
  - TSnap::GenForestFire (Forest Fire Model)
  - TSnap::GenPrefAttach (Preferential Attachment)
- Example:
  - // create a directed random graph on 100 nodes and 1k edges
    - PNGraph Graph = TSnap::GenRndGnm<PNGraph>(100, 1000);

- 1. Traverse a graph

- **// traverse the nodes**

```
for (TNGraph::TNodeI NI=Graph->BegNI(); NI<Graph->EndNI(); NI++)
   printf("%d %d %d\n",  NI.GetId(), NI.GetOutDeg(), NI.GetInDeg());
```

- **// traverse the edges**

```
for (TNGraph::TEdgeI EI=Graph->BegEI(); EI<Graph->EndEI(); EI++)
   printf("edge (%d, %d)\n", EI.GetSrcNId(), EI.GetDstNId());
```

- **// we can traverse the edges also like this**

```
for (TNGraph::TNodeI NI=Graph->BegNI(); NI<Graph->EndNI(); NI++)
   for (int e = 0; e < NI.GetOutDeg(); e++)
     printf("edge (%d %d)\n", NI.GetId(), NI.GetOutNId(e));
```

- 2. Get properties of a graph
  - ***// generate a network using Forest Fire model***

    -
      ```
      PNGraph G = TSnap::GenForestFire(1000, 0.35, 0.35);
      ```
  - ***// convert to undirected graph TUNGraph***

    -
      ```
      PUNGraph UG = TSnap::ConvertGraph<PUNGraph, PNGraph> (G);
      ```
  - ***// get largest weakly connected component of G***

    -
      ```
      PNGraph WccG = TSnap::GetMxWcc(G);
      ```
  - ***// get a subgraph induced on nodes {0,1,2,3,4}***

    -
      ```
      PNGraph SubG = TSnap::GetSubGraph (G, TIntV::GetV(0,1,2,3,4));
      ```

- 2. Get properties of a graph
  - ***// generate a network using Forest Fire model***
    - PNGraph G = TSnap::GenForestFire(1000, 0.35, 0.35);
  - ***// convert to undirected graph TUNGraph***
    - PUNGraph UG = TSnap::ConvertGraph<PUNGraph, PNGraph> (G);
  - ***// get largest weakly connected component of G***
    - PNGraph WccG = TSnap::GetM
  - ***// get a subgraph indu ,3,4}***
    - PNGraph SubG = TSnap::GetSu V(0,1,2,3,4));

Demo:
getCC.cpp

Play With A Graph

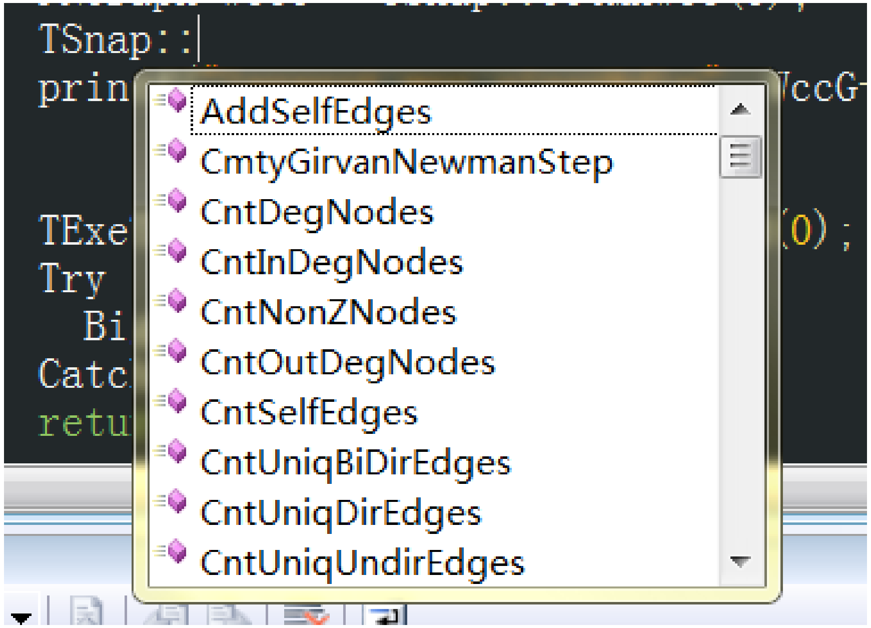
  - TVec<TPair<TInt, TInt> > CntV; // vector of pairs of integers (size, count)

- ***//get distribution of connected components (component size, count)***
  - TSnap::GetWccSzCnt(G, CntV);
- ***// get degree distribution pairs (degree, count)***
  - TSnap::GetOutDegCnt(G, CntV);

More…

As there's not much documentation to SNAP, it is vital to explore via reading source code for relevant functions&classes

- Explore namespace TSnap



```
TSnap::
prin                                              VccG
TExe                                              (0);
Try
  Bi
Catc
retu
```

What’s In SNAP?

- Data structures (In subfolder “glib”):
  - STL-like library
  - Contains basic data structures, like vectors, hash-tables and strings
  - Provides serialization for loading and saving
- Network analysis library (In subfolder “snap”)
  - Network generation, manipulation
- **Example applications (In subfolder “examples”)**
  - Small sample applications that demonstrate functionality

Example Applications

- **Cascades**: Simulate SI model on a network
- **Cliques:** Clique Percolation Method for detecting overlapping communities
- **ForestFire:** ForestFire graph generative model
- **TestGraph:** Demonstrates basic functionality of the library

Content

- Installation
- Data structures in SNAP
- Graph manipulation in SNAP
- **Datasets in SNAP**
- Plotting in SNAP
- Q&A

Datasets In SNAP

- http://snap.stanford.edu/data/index.html
- Some examples:
  - **Social networks:** online social networks, edges represent interactions between people
  - **Citation networks:** nodes represent papers, edges represent citations
  - **Collaboration networks:** nodes represent scientists, edges represent collaborations (co-authoring a paper)
  - **Amazon networks :** nodes represent products and edges link commonly co-purchased products
  - **Twitter and Memetracker :** Memetracker phrases, links and 467 million Tweets

Datasets in SNAP

- Example file (as20graph.txt in subfolder ***examples***)
  - # Directed Node Graph
  - # Autonomous systems (graph is undirected, each edge is saved twice)
  - # Nodes: 6474   Edges: 26467
  - # SrcNId   DstNId
  - 1 3
  - 1 6
  - 1 32
  - 1 48
  - 1 63
  - 1 70
  - …

Loading/Saving

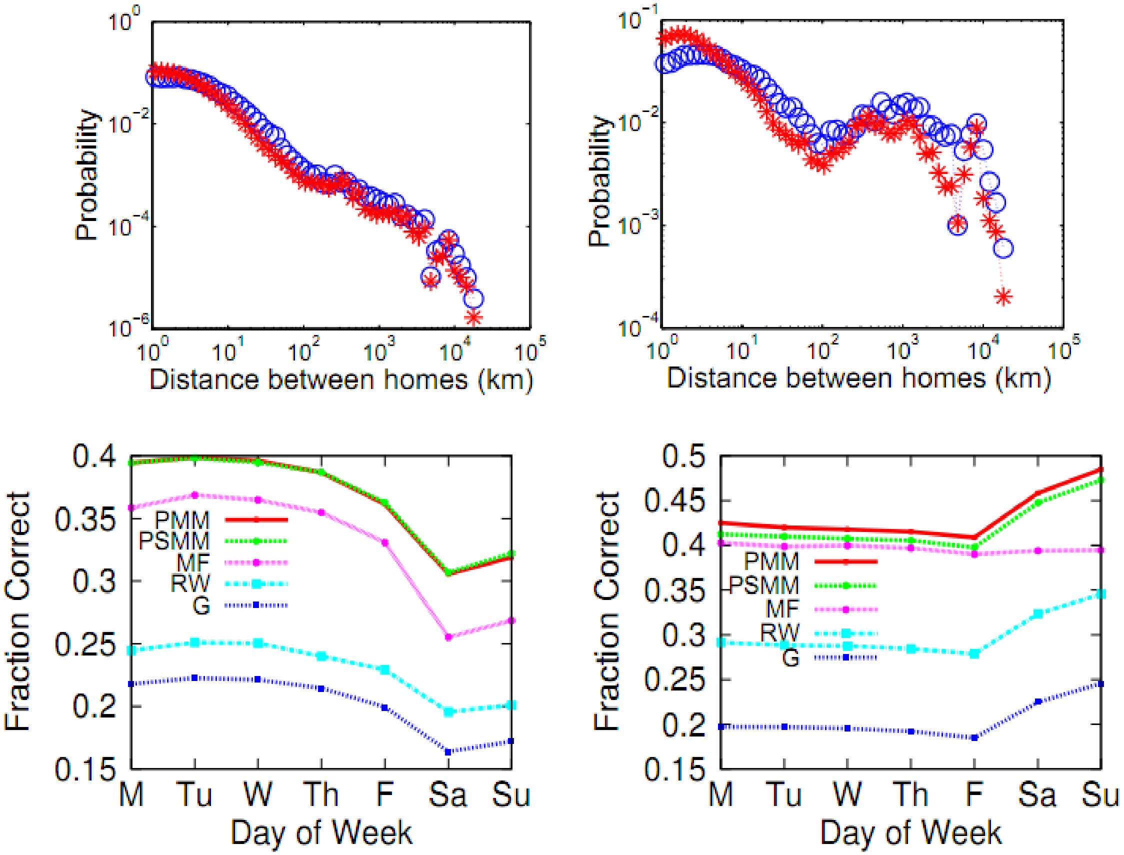
- Loading:
  - PUNGraph g=TSnap::LoadEdgeList<PUNGraph>(“as20graph.txt”,0,1);
  - 0 is the column id for source node
  - 1 is the column id for target node
- Saving
  - TSnap::SaveEdgeList<PUNGraph>(g, “as20graph.txt”, ””);
- Not as efficient as loading and saving in binary form
  - g->Save(TFOut(“graph.bin”));

Content

- Installation
- Data structures in SNAP
- Graph manipulation in SNAP
- Datasets in SNAP
- **Plotting in SNAP**
- Q&A

Want To Draw?

- Last topic: making a plot in SNAP

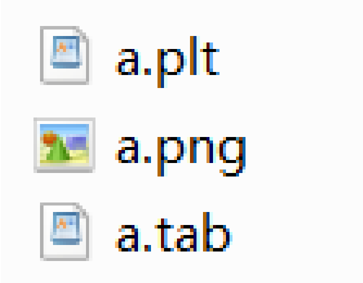


Want To Draw?

- Steps:
  - Install Gnuplot from http://www.gnuplot.info/
  - Make sure that the path containing wgnuplot.exe (for Windows) or gnuplot (for Linux) is in your environmental variable $PATH.
  - Example:
    - TVec<TPair<TFlt, TFlt > > XY1, XY2; …
    - TGnuPlot Gp(“file name”, “title name");
    - Gp.AddPlot(XY1, gpwLinesPoints, “curve1");
    - Gp.AddPlot(XY2, gpwPoints, “curve2");
    - Gp.SetXYLabel(“x-axis name", “y-axis name”);
    - Gp.SavePng(); //or Gp.SaveEps();

Gnuplot

- After executing, three files generated

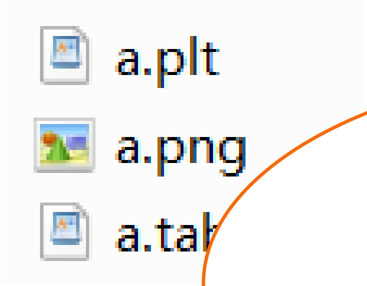


a.plt
a.png
a.tab

  - .plt file is the plotting command for gnuplot
  - .tab file contains the data
  - .png or .eps is the plot

Gnuplot

- After executing, three files generated



a.plt

a.png

a.ta

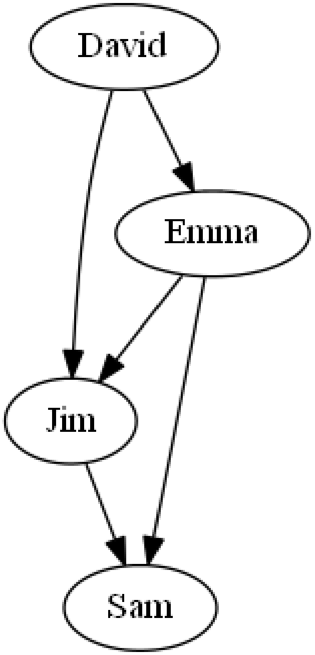
  - .plt file is the plotti
  - .tab file contains the
  - .png or .eps is the plot

**Demo:
Gnuplot_example.cpp**

Visualize Your Graph

- Use TGraphViz
- Need to install GraphViz software first
  http://www.graphviz.org/
- Add GraphViz path to environment variable
- Visualize graph with contents

```
PNGraph g=TNGraph::New();
g->AddNode(1); g->AddNode(2);
g->AddNode(3); g->AddNode(4);
g->AddEdge(1,2); g->AddEdge(2,3);
g->AddEdge(1,3); g->AddEdge(2,4);
g->AddEdge(3,4);
TIntStrH name;
name.AddDat(1)="David";
name.AddDat(2)="Emma";
name.AddDat(3)="Jim";
name.AddDat(4)="Sam";
TGraphViz::Plot<PNGraph>(g, gvlDot,
"gviz_plot.png", "", name);
```



- Q&A
- Thanks you!