The following materials have been collected from the numerous sources including my own and my students over the years of teaching and experiences of programming. Please help me to keep this tutorial up-to-date by reporting any issues or questions. Please send any comments or criticisms to idebtor@gmail.com. Your assistances and comments will be appreciated.

# PSet – Graph

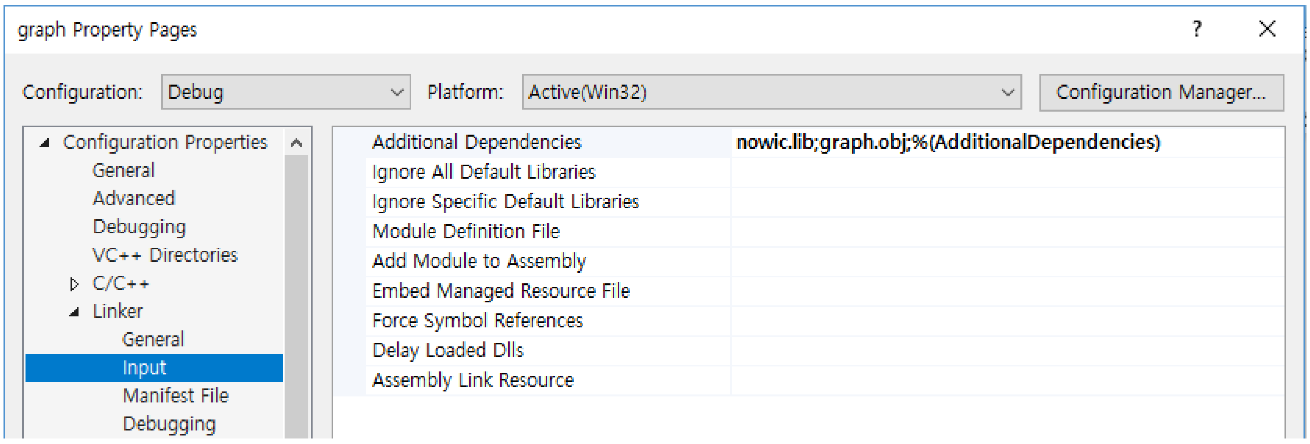
## Files provided

As a warming up, you build a project called graph and display a graph menu and a graph. The following files are provided. Build the project with lib/nowic.lib and include/nowic.h. You may use gcc for this project as well.

| | |
|---|---|
| • graph.h | - Don't change this file |
| • graph.o_mac | - for mac and g++ command line |
| • graph.o | - **for pc and g++ command line** |
| • graph.obj | **- for visual studio** |
| • graphDriver.cpp | - Most of your work goes this file |
| • antenna.txt | – graph files, place them in VS project folder. |
| • graph?.txt | - some graph files for your testing |
| • graphx.exe | - a solution example to compare with **for pc** |
| • graphx | - a solution example to compare with **for mac** |
| • psetgraph.docx | - instructions, **test and self-grading**: |

## How to compile

**For visual studio,** graph.obj goes to lib folder where nowic.lib is. Using project properties, add **graph.obj** where nowic.lib is as shown below.


graph Property Pages

Configuration: Debug Platform: Active(Win32)

Linker > Input

Additional Dependencies: nowic.lib;graph.obj;%(AdditionalDependencies)

**Using pc and g++ on console**, use the following commands:

```
g++ -std=c++11 graphDriver.cpp graph.o -I../include -L../lib -lnowic -o graph
```

Using Mac and g++, use the following commands:

```
g++ -std=c++11 graphDriver.cpp graph.o_mac -I../include -L../lib -lnowic_mac -o graph
```

Using xcode, you let me know once you figure it out such that I can add them here.

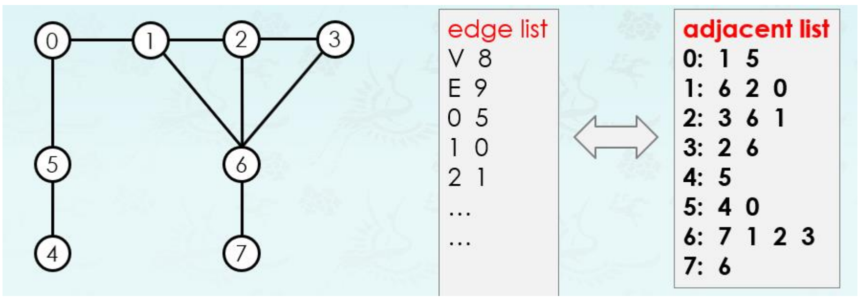
## Step 1: cyclic_check()

Code cyclic_check() in graphDriver.cpp. This function is invoked just right after you enter 'q' to quit in the menu, but before exiting the program.

- Don't use graph_by_file(), but use Graph() and addEdge().
- Run DFS and BFS at v = 0, then print results saved in the graph structure. Don't use print_DFS() and print_BFS().

## Step 2: antenna.txt and some graph files

Using a graph called "antenna.txt" provided, complete the file which can represent the following graph.

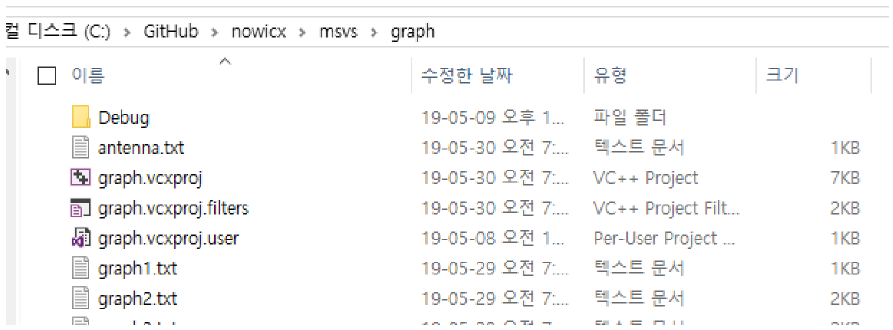


edge list
V 8
E 9
0 5
1 0
2 1
…
…

adjacent list
0: 1 5
1: 6 2 0
2: 3 6 1
3: 2 6
4: 5
5: 4 0
6: 7 1 2 3
7: 6

Since the antenna.txt requires to have an edge list, you must come up with an edge list that produces the adjacent list given above.

1. Figure out an edge list that generates the adj-list
2. Add the edge list in antenna.txt. In visual studio, this file is located where the project file is as shown below:



컴 디스크 (C:) › GitHub › nowicx › msvs › graph

| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| Debug | 19-05-09 오후 1... | 파일 폴더 | |
| antenna.txt | 19-05-30 오전 7:... | 텍스트 문서 | 1KB |
| graph.vcxproj | 19-05-30 오전 7:... | VC++ Project | 7KB |
| graph.vcxproj.filters | 19-05-30 오전 7:... | VC++ Project Filt... | 2KB |
| graph.vcxproj.user | 19-05-08 오전 1... | Per-User Project ... | 1KB |
| graph1.txt | 19-05-29 오전 7:... | 텍스트 문서 | 1KB |
| graph2.txt | 19-05-29 오전 7:... | 텍스트 문서 | 2KB |

3. Run the graphx.exe and check that your antenna.txt file produces the exact adj-list. Use the output of testing to complete antenna.txt to fill out the results algorithms such as DFS, CCID, BFS, DistTo.... etc.

## Step 3: case p – path between two vertices

Code "case p" in the main() of graphDriver.cpp. It should function as shown graphx.exe provided.

## Step 4: case a – bipartite using adj-list coloring

Code **"case a"** in the main() of graphDriver.cpp. It should function something like results shown in **"case t"** in graphx.exe provided.

## Submitting your solution

- Include the following line at the top of your every file with your name signed.
  On my honour, I pledge that I have neither received nor provided improper assistance in the completion of this assignment. Signed: _____________
- Make sure your code **compiles** and **runs** right before you submit it. Don't make "a tiny last-minute change" and assume your code still compiles. You will not receive sympathy for code that "almost" works.
- If you only manage to work out the homework partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- Place your source files in the folder you and I are sharing.
- After submitting, if you realize one of your programs is flawed, you may fix it and submit again as long as it is **before the deadline**. You may submit as often as you like. **Only the last version** you submit before the deadline will be graded.

## Files to submit

- graphDriver.cpp, psetgraph.docx with self-graing filled.

## Due and Grade points

- Due: June 6, 11:55pm
- Grade points: 5 points
  - 1 point per step except step 4 for two points

## Test and Self-Grading

Name: __________________ Student Number:____________________ Section: ________
NOTE: 20% penality for incorrect test and self-grading.

1. Compare the results with graphx.exe – Your point _____________
2. Compare the results with graphx.exe – Your point _____________

3. Compare the results with graphx.exe – Your point _____________
4. Compare the results with graphx.exe – Your point _____________

NOTE: Describe anything or potention bugs grader or instructor need to pay attention.