# GS2402 Data Structures and Algorithms

Homework 3

Searching Algorithms

(Due date: Friday November 27th 2015, 23:59)

*Instructions:*

1. Login to Newton (**gs.gist.ac.kr** / **210.107.178.136**).
2. Create your program using any text editor of your choice (e.g. **vim / nano / emacs**)
3. Name your files using the following convention:

| | |
|---|---|
| Class definition files | Graph.h, List.h |
| Main program file | *student-id+"hw"+number+".cpp"* (e.g. **20145678hw03.cpp**) |

4. Compile your program and test that it works correctly (e.g. **g++ 20145678hw03.cpp –o prog**)
5. Submit all your c++ source code files and header files (3 separate files) (Type **~gs2402/bin/checkin HW03 20145678hw03.cpp Graph.h List.h**)

---

## Finding driving directions

**Pathfinding** or **pathing** is the plotting, by a computer application, of the shortest route between two graph points. Pathfinding is used in many areas, like GPS navigation, CAD systems (auto-routing in design of circuit boards), and applications of artificial intelligence, computer games, virtual reality, military forces or robotics.

Two primary problems of pathfinding are to find a path between two nodes in a graph and to find the optimal shortest path. Basic algorithms such as breadth-first and depth-first search address the first problem by exhausting all possibilities; starting from the given node, they iterate over all potential paths until they reach the destination node. These algorithms run in $O(|V| + |E|)$, or linear time, where V is the number of vertices, and E is the number of edges between vertices.

The more complicated problem is finding the optimal path. The exhaustive approach in this case is known as the Bellman–Ford algorithm, which yields a time complexity of O(|V||E|), or quadratic time. However, it is not necessary to examine all possible paths to find the optimal one. Algorithms such as A* and Dijkstra's algorithm strategically eliminate paths, either through heuristics or through dynamic programming. By eliminating impossible paths, these algorithms can achieve time complexities as low as O(|E|log|V|).

The file **drivingdistances.txt** contains some of the intercity driving distances (in km) between the 20 most highly-populated cities in South Korea. If an entry between two particular cities is not there, it means that the road connecting them is not accessible. A separate file **straightlinedistances.txt** contains the straight line distances (“as the crow flies”) from each of the 20 cities to Seoul.

Write a program to find the optimal driving path from Gwangju to Seoul. Your output should contain the sequence of the cities visited, and the total distance travelled.