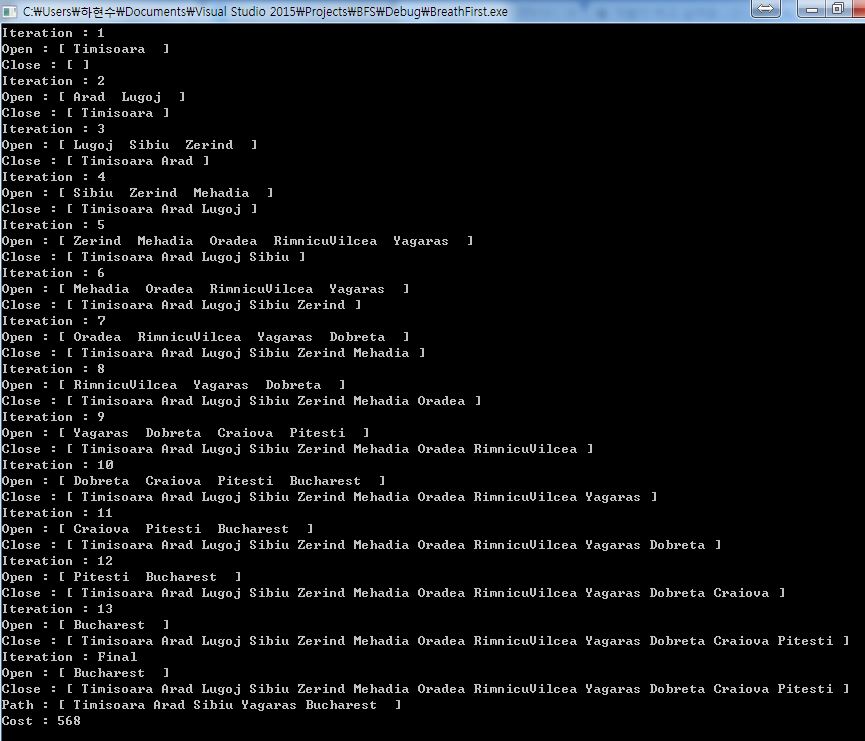
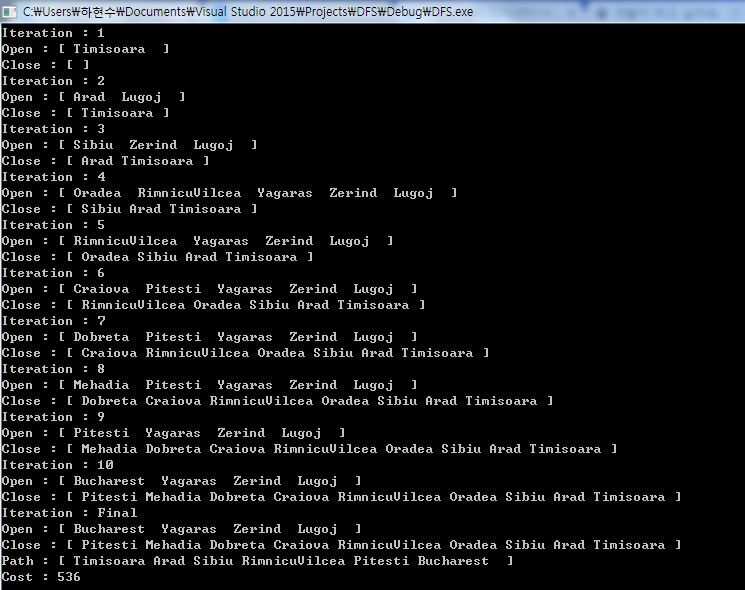
인공지능 Assignment #2

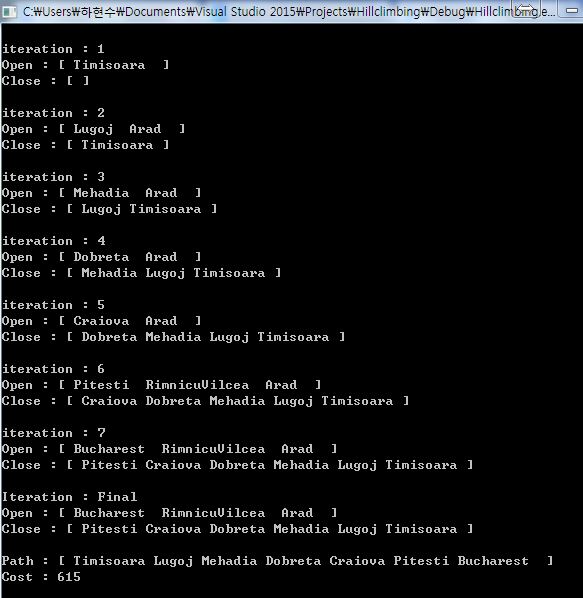
학번 : 20150291

학부 : 소프트웨어학부

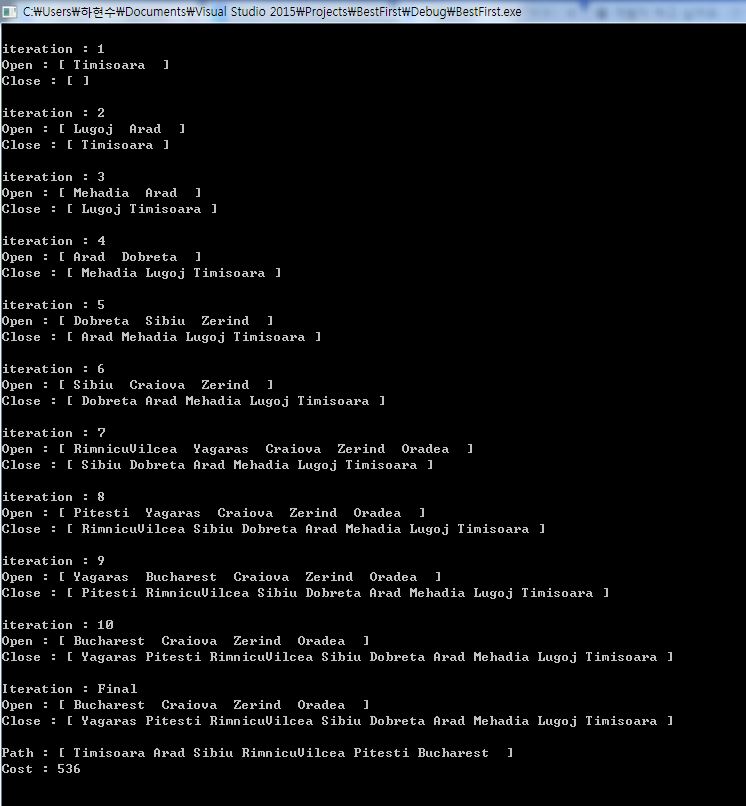
이름 : 하현수

구현 프로그램 스크린샷

1. Breadth-First Search
2. Depth-First Search
3. Hill-Climbing

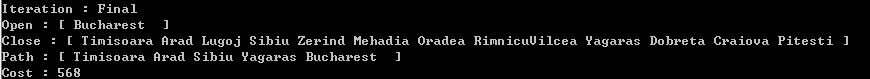


1. Best-First Search



Question :

1. : Breadth-First Search에 의하여 가장 먼저 생성된 path의 cost 및 생성된 node의 수는?



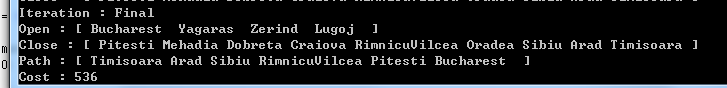
Cost : 568

Node의 개수, Open : 1개, Close : 12개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 13개.

1. : Depth-First Search에 의하여 가장 먼저 생성된 Path의 cost및 생성된 node의 수는?



Cost : 536

Node의 개수, Open : 4개, Close : 9개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 13개

1. Shortest Path 및 그 Cost는?

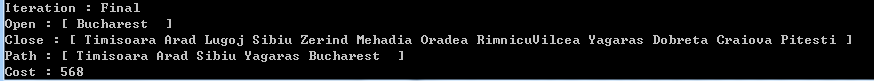
직접 손으로 따라그려보며 계산한 결과

최소거리는 Timisoara – Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest로

Cost는 536입니다.

1. Shortest Path를 찾는 경우, 각각의 알고리즘에 의하여 생성되는 node의 수는?

Breadth-First Search의 경우

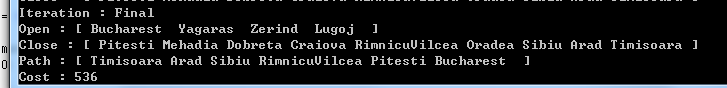


Node의 개수, Open : 4개, Close : 9개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 13개의 최종노드를 가지고있으며 최단 경로를 찾지 못했습니다.

Depth-First Search의 경우

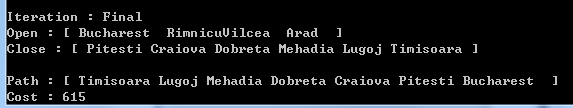


Node의 개수, Open : 4개, Close : 9개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 13개의 최종노드를 가지며 최단 경로를 찾아냈습니다.

Hill-Climbing의 경우

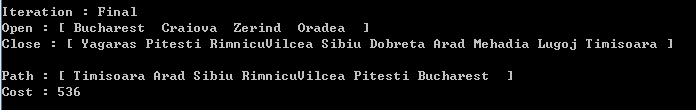


Node의 개수, Open : 3개, Close : 6개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 9개의 최종 노드를 가지지만, 최단 경로를 찾아내지 못했습니다.

Best-First Search의 경우



Node의 개수, Open : 4개, Close : 9개

(//Bucharest가 Open에서 뽑히면 루프를 탈출하는 방식이라서 Close안으로 넣지 않았습니다.)

총 13개의 최종노드를 가지며, 최단경로를 찾아내었습니다.

1. 결과를 분석하시오

4가지 알고리즘을 모두 사용해 본 결과, 최단경로를 찾아낸 알고리즘은 Depth-first Search와, Best-First Search 뿐이었습니다.

Hill Climbing은 가장 적은 최종 노드를 가지며, 가장 적은 반복으로 경로를 찾아 내었지만 정작 Cost값은 가장 컸고,

Breadth-first Search는 가장 많은 반복횟수를 가졌으나 최단 경로를 구해내지 못했습니다.

이에 반해, Depth-first Search와 Best-first Search는 비교적 적은 Iteration으로 최단 경로를 구해냈습니다.

이러한 결과를 바탕으로, 이 4가지 알고리즘은 문제의 형식과 주어진 f(n)에 따라서 최단 경로를 찾을 수도, 없을 수도 있다는 것을 알 수 있으며, 알고리즘의 효율성 또한 해결해야 할 문제에 따라 다르다는 것을 알 수 있었습니다.

확실한 것은, 이 4가지 알고리즘 모두 어떤 문제에서든지 시작노드에서 목표노드까지의 최단거리를 구할 수 있다고 보장할 수 없다는 것입니다.