**The study A\* Algorithm to use MATLAB**

|  |  |  |  |
| --- | --- | --- | --- |
| 항목 | MATLAB with AI | 이름 | Simon |
| 날짜 | 20200922 | 번호 | 1 |

1. **What is A\* Algorithm**
   1. Informed Search

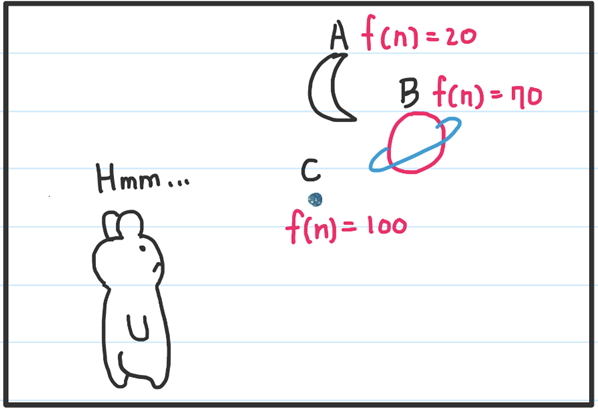
Search, 탐색에 관하여 알아봅시다. 지금 가고 싶은 곳을 떠올려 봅시다. 그리고 현재위치에서 목적지까지 시간을 얼마나 걸릴까요? 비용은 얼마나 들까요? 또 어디를 거쳐가야 할까요? 인터넷에 검색해보면 빠르게 답을 찾을 수 있습니다. 그런데 사실 다른 방법으로도 갈 수 있고 그 방법이 더 빠르거나 저렴하기도 할 수 있습니다.

Informed search는 우리가 여행계획 짜는 것과 비슷합니다. 목적지도 알고 중간에 어디를 거쳐야할 지 그리고 각 구간마다 비용도 혹은 거리도 다 알고 있습니다. 그래서 Informed search는 나중에 얘기할 Uninformed search 혹은 Blind search에 비하면 방향도 정해져 있으며 불 필요한 지점을 들리지 않습니다.

Informed search의 대표적인 알고리즘은 Best-First Search(BFS)입니다. 그리고 Informed search의 알고리즘은 대부분 BFS를 뿌리로 두고 있습니다. BFS의 가장 큰 특징은 평가 함수(Evaluation function) F(n)을 이용합니다.

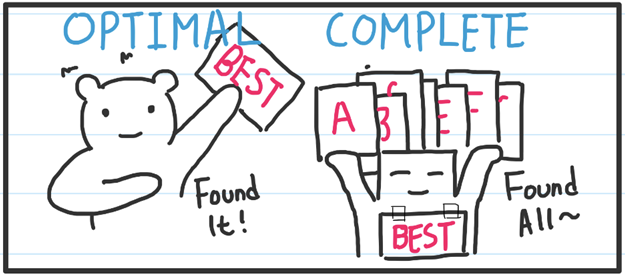
* 1. Evaluation function

Evaluation function은 BFS에서 중요한 요소입니다. 함수니까 앞으로 F(n)이라고 얘기하겠습니다. 탐색을 시작할 때 당장 갈 수 있는 거리에 대해서 F(n)을 따져봅니다. A, B 그리고 C에 갈 수 있을 때, F(n)이 제일 낮은 곳부터 가는 것만 따집니다. 즉 BFS에서는 F(n)없이 한발짝도 움직이지 못합니다.



* 1. Complete? Optimal? Technology terms

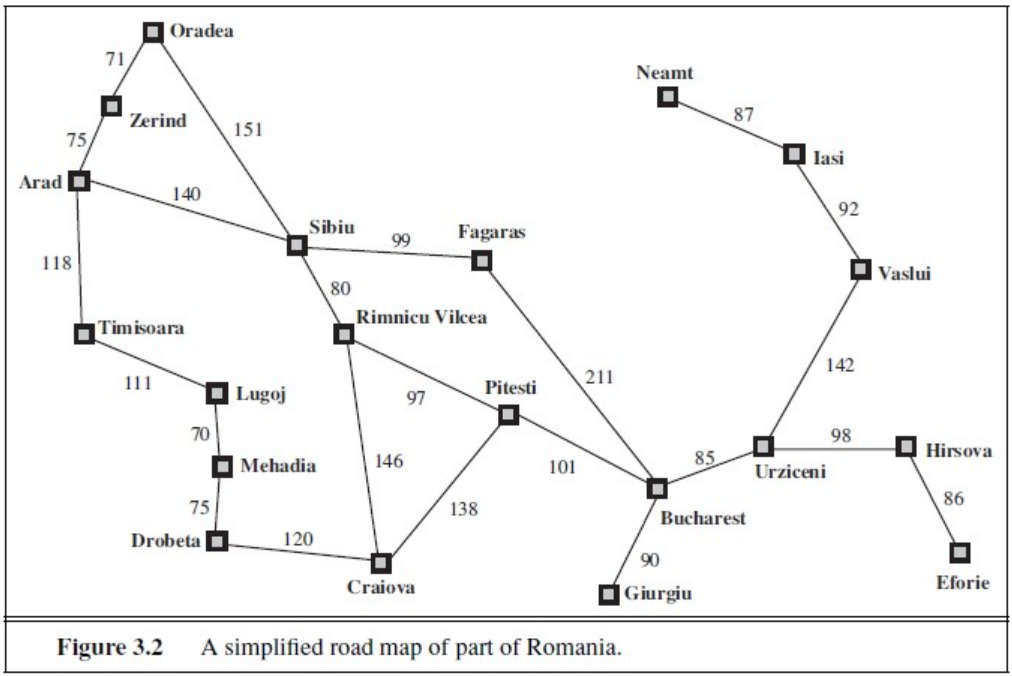
지금까지 얘기했던 내용에는 State와 탐색에 대한 Complete or Optimal solution, Big-O notation 등 여러가지 기술용어들이 있습니다. 몰라도 뭐든지 할 수 있으니 쉽게 이해할 수 있게 쓰겠습니다.

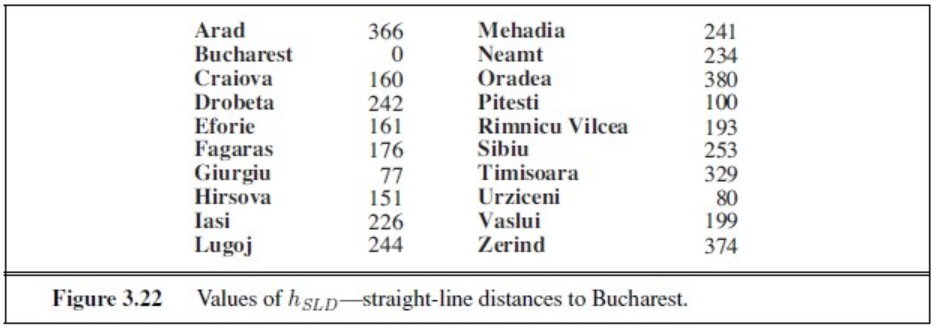


* 1. A\* algorithm

A\* Algorithm은 Evaluation function 설계는 아래와 같이 이루어집니다. 먼저 출발지로부터 목적지까지의 비용과 우리가 들리는 지점에서 목적지까지 드는 비용입니다. 거리로 비유하자면 출발지에서 목적지까지의 직선거리와 각 지점에서 목적지까지 거리 혹은 출발지에서 각 지점까지 거리(내가 걸어온 거리)도 됩니다. 이 점을 기억하셔서 실습해봅시다.

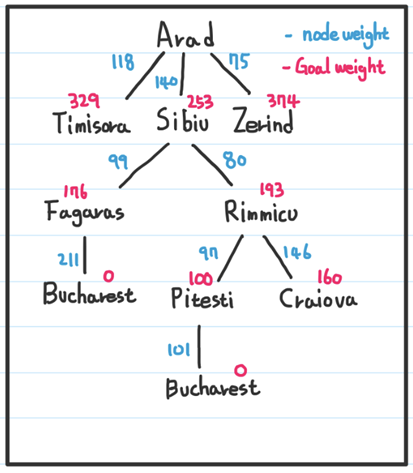
1. **Implement**
   1. A simplified road map of part of Romania



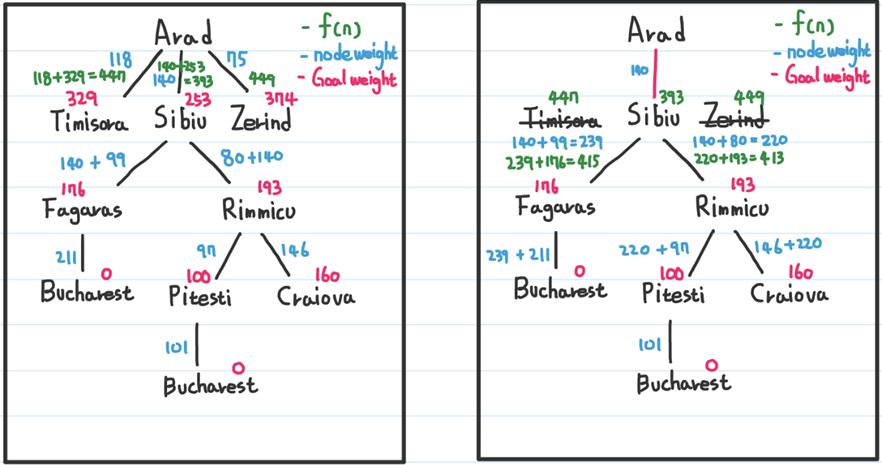


A\* Algorithm에 이용할 루마니아 지도입니다. 좌측 중단의 Arad에서 시작해서 목적지는 Bucharest입니다. Figure 3.22은 목적지까지의 직선거리를 적은 것이고 각 지점별로 걸리는 이동거리는 Figure 3.2에서 확인할 수 있습니다. 실습을 하기위해서 가져온 지도를 조금 바꿔보겠습니다.

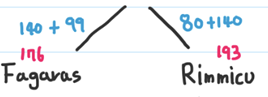
* 1. Romania Tree

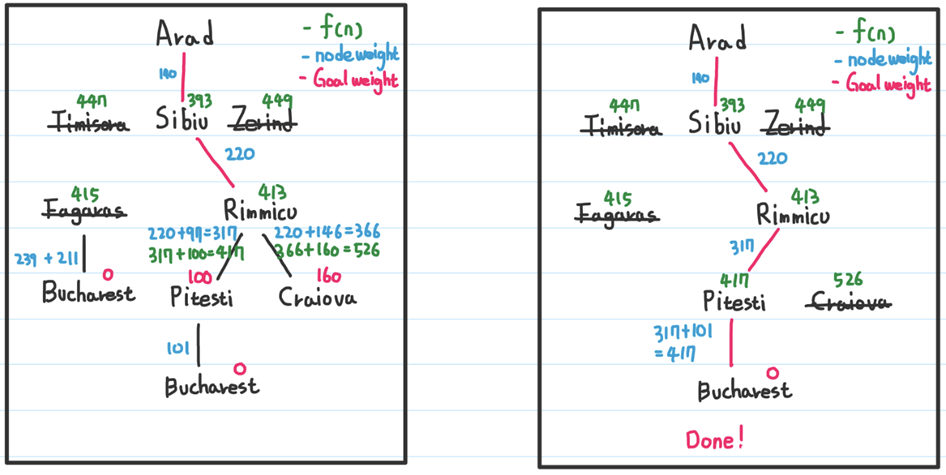


핵심을 알아야 하니 필요 없는 구간은 지웠으며 빨간색이 직선거리, 파란색이 지점별 거리입니다. Arad에서 출발하겠습니다.



녹색은 지점별 거리와 직선거리를 더한 값입니다. 즉 A\* algorithm의 핵심인 평가함수를 직접 계산하고 있습니다. 좌측 이미지를 보아하니 Sibiu가 393으로 제일 낮은 값을 얻었네요. Sibiu로 넘어갑시다.

이제부터는 제가 걸어온 거리까지 더합니다! 사실 아까는 걸어온 거리가 0이여서 계산할 이유가 없었어요. 오른쪽 이미지를 보니 Rimmicu가 낮은 값을 얻었네요! Rimmicu로 가겠습니다.



Rimmicu로 넘어와보니 Pitesti가 낮은 값을 받았고 Pitesti를 도착하면서 목적지인 Bucharest가 보이기 시작합니다. 이 것을 끝으로 A\* Algorithm을 손수 해볼 수 있었습니다.

* 1. MATLAB code

코드 올릴 예정

clc, clear

import java.util.LinkedList

%Straight line distances to Buchrest

StraightDist = py.dict(pyargs(...

'Arad',366,'Bucharest',0,'Craiova',160,'Drobeta',242,'Eforie',161, ...

'Fagaras',176,'Giurgiu',77,'Hirsova',151,'Iasi',226,'Lugoj',244, ...

'Mehadia',241,'Neamt',234,'Oradea',380,'Pitesti',100,'Rimmicu',193, ...

'Sibiu',253,'Timisoara',329,'Uziceni',80,'Vaslui',199,'Zerind',374));

%Node Weight

NodeDist = py.dict(pyargs(...

'Arad\_Zerind',75,'Arad\_Sibiu',140,'Arad\_Timisoara',118, ...

'Zerind\_Arad',75,'Zerind\_Oradea',71, ...

'Sibiu\_Fagaras',99,'Sibiu\_Rimmicu',80, 'Sibiu\_Arad',140, ...

'Timisoara\_Arad',118,'Timisoara\_Lugoj',111, ...

'Oradea\_Zerind',71,'Oradea\_Sibiu',151, ...

'Fagaras\_Sibiu',99,'Fagaras\_Bucharest',211, ...

'Rimmicu\_Sibiu',80,'Rimmicu\_Pitesti',97, 'Rimmicu\_Craiova',146, ...

'Lugoj\_Timisoara',111,'Lugoj\_Mehadia',70, ...

'Pitesti\_Rimmicu',97,'Pitesti\_Bucharest',101,'Pitesti\_Craiova',138, ...

'Mehadia\_Lugoj',70,'Mehadia\_Drobeta',75, ...

'Drobeta\_Mehadia',75,'Drobeta\_Craiova',120, ...

'Craiova\_Drobeta',120,'Craiova\_Rimmicu',146, 'Craiova\_Pitesti',138, ...

'Bucharest\_Fagaras',211,'Bucharest\_Pitesti',101, 'Bucharest\_Giurgiu',90, 'Bucharest\_Uziceni',85, ...

'Uziceni\_Bucharest',85,'Uziceni\_Hirsova',98, 'Uziceni\_Vaslui',142, ...

'Hirsova\_Uziceni',98,'Hirsova\_Eforie',86, ...

'Vaslui\_Uziceni',142,'Vaslui\_Iasi',92, ...

'Iasi\_Vaslui',92,'Iasi\_Neamt',87, ...

'Giurgiu\_Bucharest',90,'Eforie\_Hirsova',86, 'Neamt\_Iasi',87));

%Site List

SiteNames = {'Arad' 'Bucharest' 'Craiova' 'Drobeta' 'Eforie' ...

'Fagaras' 'Giurgiu' 'Hirsova' 'Iasi' 'Lugoj' ...

'Mehadia' 'Neamt' 'Oradea' 'Pitesti' 'Rimmicu' ...

'Sibiu' 'Timisoara' 'Uziceni' 'Vaslui' 'Zerind'};

InitialState = 'Arad';

GoalState = 'Bucharest';

%Queue

StateSpace = LinkedList();

StateSpace.addLast(InitialState)

%Visualization

GPSData = py.dict(pyargs(...

'Arad',[46.185401, 21.322492],'Bucharest',[44.434308, 26.092571],'Craiova',[44.327497 23.786745],'Drobeta',[44.638060 22.661103],'Eforie',[44.049730 28.652512], ...

'Fagaras',[45.842674 24.970539],'Giurgiu',[43.894407 25.965605],'Hirsova',[44.688055 27.946515],'Iasi',[47.166790 27.583530],'Lugoj',[45.688312 21.904419], ...

'Mehadia',[44.906961 22.367557],'Neamt',[46.935242 26.362483],'Oradea',[47.058607 21.942244],'Pitesti',[44.858139 24.870755],'Rimmicu',[45.106551 24.359350], ...

'Sibiu',[45.796382 24.154599],'Timisoara',[45.753437 21.224136],'Uziceni',[44.718908 26.644935],'Vaslui',[46.645329 27.728033],'Zerind',[46.622826 21.516294]));

Run = true;

Cumulative = 0;

while Run

%Dequeue

Visit = StateSpace.removeFirst()

%Plot geography

temp = GPSData{Visit}.double;

lat = temp(1);

lon = temp(2);

geoplot(lat,lon, 'r-O')

hold on

text(lat+0.1,lon+0.1,Visit);

% Done?

if strcmp(Visit,GoalState)

geolimits([42 48],[19 30])

hold off

return;

end

% Generate child nodes

MinFnSpace = 0;

MinIndex = 0;

TempQ = GenChildNodes(Visit, SiteNames, NodeDist, StateSpace);

for i=1:TempQ.size

% Calculate Evaluation Function - f(n) = g(n) + h(n)

Containers = TempQ.removeFirst();

MinFnSpace(i) = NodeDist{Visit + "\_" + char(Containers)} + Cumulative; %g(n)

MinFnSpace(i) = MinFnSpace(i) + StraightDist{char(Containers)}; %h(n)

if (MinIndex >= MinFnSpace(i)) || (MinIndex == 0)

MinIndex = MinFnSpace(i);

NextSite = char(Containers);

end

end

%Choice next site

StateSpace.addLast(NextSite);

temp = GPSData{NextSite}.double;

lat2 = temp(1);

lon2 = temp(2);

geoplot([lat lat2],[lon lon2], 'r-o')

clearvars MinFnSpace MinSite;

end

function y = GenChildNodes(InState, InSpace, Dict, Queue)

for i=1:length(InSpace)

try

ParamName = InState + "\_" + InSpace(i);

Dict{ParamName};

Queue.addLast(InSpace(i));

catch

end

end

y = Queue;

end