

A* 알고리즘을 이용한 최단 경로 탐색

2019144060 신승우

2021111012 이승연

2020147051 장 건

2021171001 한준희

CONTENTS

01 목적

02 Knowledge-Base 구축

03 Prolog 코드

04 의의 및 한계

목적

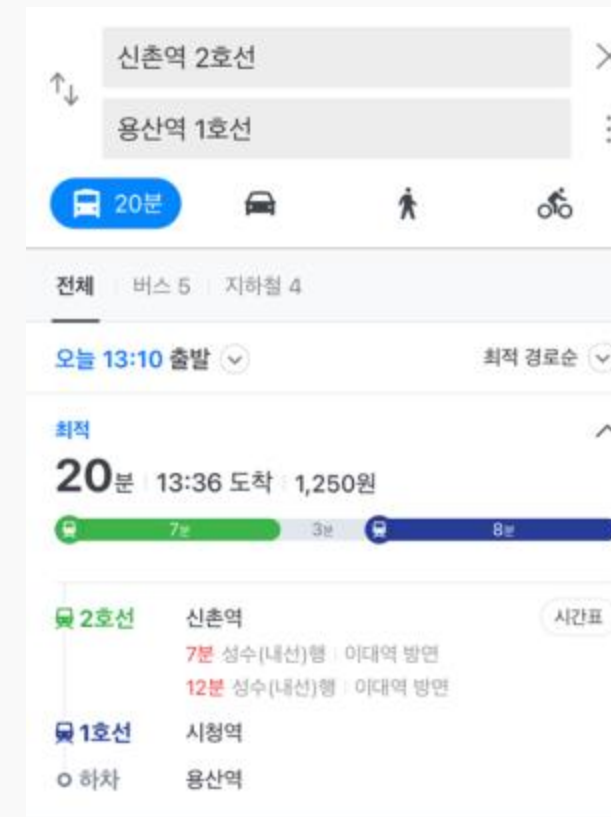
Knowledge-Base 구축

Prolog 코드

의의 및 한계

- 실생활에서 널리 활용되는 '네이버 지도'나 '카카오맵'에서의 '길찾기' 기능

➡ 대중교통 이용 시 최단경로에 대한 정보 획득



- 휴리스틱을 이용한 프롤로그를 통해 이러한 시스템을 실현할 수 있는가?

A* 알고리즘을 활용한 프롤로그(PROLOG)를 활용한
지하철 최단경로 제공 시스템 구현

목적

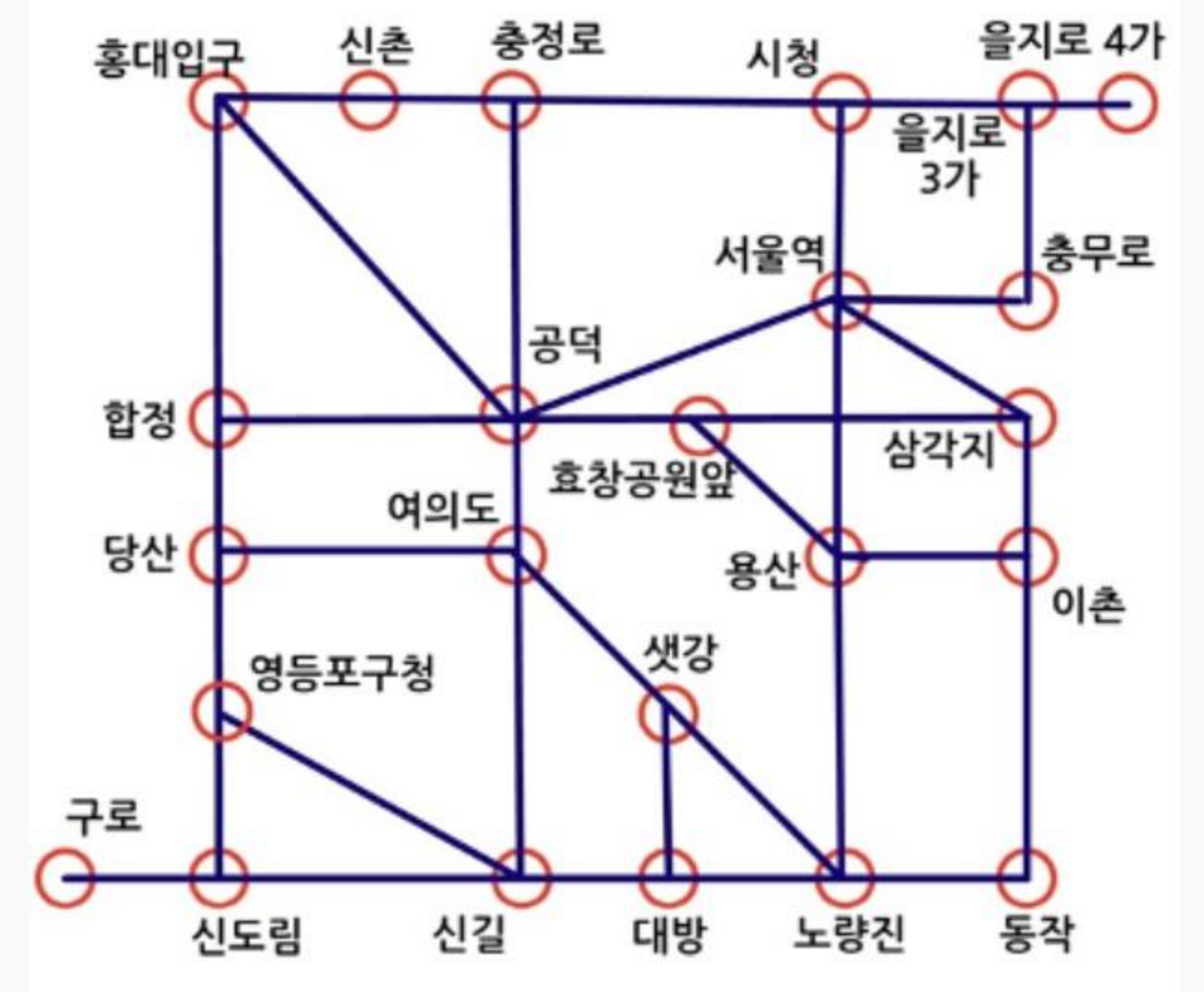
Knowledge-Base 구축

Prolog 코드

의의 및 한계

지하철역 선정

- 서울역을 중심으로 여러 호선이 혼재하고 있는 인근 지역을 주된 범위 설정
➡ 서울역을 포함한 환승역 23곳 + 연세대학교와 근접한 신촌역 = 총 24곳



목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

지하철 호선 및 역간거리

- 서울교통공사에서 제공하는 사이트를 통해 최단거리 검색, 24개 역간 거리를 기록
각 역과 역 사이의 선로에 해당하는 호선 또한 기입

데이터 출처: <http://www.seoulmetro.co.kr/kr/cyberStation.do#wayInfo>



```
road(영등포구청, 당산, 1.1), subway(영등포구청, 당산, '2호선').
```

* 두 역 간에 복수의 경로가 있고 역간 거리가 같은 경우, 모든 호선을 병기

** 두 역 간에 복수의 경로가 있고 역간 거리 또한 다른 경우,

최단경로 관점이기때 역간 거리가 짧은 노선만 기입

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

지하철역의 좌표

- `hvalue(Station, Destination, Distance)`에서
역 간의 직선거리를 활용할 수 있도록 각 역의 위경도 좌표를 프롤로그 코드에 기입

데이터 출처: https://github.com/henewsuh/subway_crd_line_info/blob/main/%EC%A7%80%ED%95%98%EC%B2%A0%EC%97%AD_%EC%A2%8C%ED%91%9C.csv

```
coordinates(37.503039, 126.881966, 구로).  
coordinates(37.508725, 126.891295, 신도림).  
coordinates(37.52497, 126.895951, 영등포구청).  
coordinates(37.517122, 126.917169, 신길).  
coordinates(37.53438, 126.902281, 당산).  
coordinates(37.521624, 126.924191, 여의도).  
coordinates(37.517274, 126.928422, 샛강).
```

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

A* 알고리즘

- $f(n) = g(n) + h(n)$

- $g(n)$ 함수: 출발점에서 측정 시점 노드까지의 거리

지하철 역간거리 사용

- $h(n)$ 함수: 측정 시점 포인트에서 목표 노드까지의 거리

유클리디안 거리 사용

cf) 맨해튼 거리

h-value로 활용할 수 있는 또 다른 지표 후보:

격자 상태에서의 최단 거리를 의미하여 두 지점(두 역) 간 최단 거리를 보장 X

➡ 맨해튼 거리를 제외한 직선(최단)거리를 보장하는 유클리디안 거리만을 활용

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
hvalue(Station, Destination, Distance) :-  
    coordinates(X1, Y1, Station),  
    coordinates(X2, Y2, Destination),  
    Dlat is (X2 - X1) * pi / 180,  
    Dlon is (Y2 - Y1) * pi / 180,  
    A is sin(Dlat / 2) ** 2 + cos(X1 * pi / 180) * cos(X2 * pi / 180) * sin(Dlon / 2) ** 2,  
    C is 2 * atan2(sqrt(A), sqrt(1 - A)),  
    Distance is C * 6371.0.
```

- hvalue(Station, Destination, Distance): 두 노드의 유클리디안 거리 측정 위한 predicate
- 위경도 좌표를 이용하여 *하버사인 공식* 통해 거리 측정

- 하버사인 공식

$$a = \sin^2(\Delta\varphi / 2) + \cos \varphi_1 \times \cos \varphi_2 \times \sin^2(\Delta\lambda / 2)$$

$$c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \times c$$

($\Delta\varphi$ 는 두 역 간의 위도차, $\Delta\lambda$ 는 두 역간의 경도차, R = 지구의 반지름(6,371 km))

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

작동 원리

```
r(X, Y, Z):-road(X, Y, Z).  
r(X, Y, Z):-road(Y, X, Z).
```

```
sb1(X, Y, Z):-subway(X, Y, Z).  
sb1(X, Y, Z):-subway(Y, X, Z).
```

- 역(노드)간 양방향성을 위한 predicate

예시) subway(영등포구청, 당산, '2호선'). / subway(당산, 영등포구청, '2호선').

road(영등포구청, 당산, 1.1). / road(당산, 영등포구청, 1.1).

- 경로의 방향만 다른 동일한 경로

➡ sb(X,Y,Z) 와 r(X,Y,Z)를 사용하여 방향에 따라 2번 입력해야 하는 번거로움 해결

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
shortest_path(Point, Destination):-  
    astar([[0,Point]],Destination,ReversePath),  
    reverse(ReversePath, Path),  
    write('최단경로: '), print_path(Path,Subways),  
    write('경로의 역간 이용 지하철 호선: '),print_subways(Subways),!.
```

- 전체적인 코드의 흐름

- 1) A* 알고리즘을 이용하여 경로를 찾기 (astar)
- 2) 찾은 경로를 원하는 형식으로 출력하기 위해 변형하고 출력
(reverse, print_path, print_subways)

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
astar(Paths, Destination, [C, Destination | Path]) :-  
    member([C, Destination | Path], Paths),  
    choosebest(Paths, Destination, [C1 | _]),  
    C1 == C.
```

- A*알고리즘의 전체적인 흐름

- 1) 현재까지 찾은 경로 중에서 도착역을 포함한 경로 찾기

이 경로가 현재까지 찾은 경로들을 대상으로 휴리스틱 함수값 $f(n)$ 이 가장 적은 경로인지 확인

일치할 경우, 그 경로가 A* 알고리즘을 이용한 최적의 경로 / 아닌 경우, 2번째 과정 진행

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
astar(Paths, Destination, BestPath):-  
    choosebest(Paths, Destination, Best),  
    delete(Paths, Best, PrevPaths),  
    expand(Best, NewPaths),  
    append(PrevPaths, NewPaths, L),  
    astar(L, Destination, BestPath).
```

2) 기존 경로들 중에서 도착역을 포함한 경로가 없기 때문에 기존 경로들의 최근 역(노드)을 기준으로 휴리스틱 함수값 $f(n)$ 이 가장 작은 값을 선택
이 경로의 가장 최근 역(노드)에서 지나온 역(노드)을 제외하고 갈 수 있는 모든 노드를 찾고 이들의 가짓수만큼 새 경로 생성

이후 $f(n)$ 이 가장 작아서 선택된 경로를 제외한 기존경로 + 생성된 새로운 경로들을 대상으로 출발지로부터 최신역(노드)까지 이동한 총 거리를 갱신
목적지까지의 최단거리 경로를 찾을 때까지 A*알고리즘 반복 진행

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
choosebest([X],_,X):-!.
choosebest([[C1,Station1|Y],[C2,Station2|_] | Z], Destination,Best):-
    hvalue(Station1, Destination,H1),
    hvalue(Station2, Destination,H2),
    H1 + C1 <= H2 + C2,
    choosebest([[C1,Station1|Y] | Z], Destination,Best).
choosebest([[C1,Station1|_]],[C2,Station2|Y] | Z], Destination,Best):-
    hvalue(Station1, Destination, H1),
    hvalue(Station2,Destination, H2),
    H1 + C1 > H2 + C2,
    choosebest([[C2,Station2|Y] | Z], Destination,Best).
```

- A* 알고리즘의 세부 부분 (직접 정의한 predicate 설명)
- choosebest predicate는 휴리스틱 함수를 기준으로 계산하여 그 값이 가장 작은 경로를 선택
- 경로들 중 앞에서 2개를 골라 각 경로들의 가장 최신역(노드)를 기준으로 휴리스틱 함수 f(n) 도출

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
hvalue(Station, Destination, Distance) :-  
    coordinates(X1, Y1, Station),  
    coordinates(X2, Y2, Destination),  
    Dlat is (X2 - X1) * pi / 180,  
    Dlon is (Y2 - Y1) * pi / 180,  
    A is sin(Dlat / 2) ** 2 + cos(X1 * pi / 180) * cos(X2 * pi / 180) * sin(Dlon / 2) ** 2,  
    C is 2 * atan2(sqrt(A), sqrt(1 - A)),  
    Distance is C * 6371.0.
```

- hvalue predicate를 사용하여 h(n)을 구하고, g(n)은 지금까지 이동해온 총 거리 사용
- 이렇게 구한 f(n)을 서로 비교하여 더 작은 값을 가지는 경로 선택
- 이 과정을 반복하여 하나의 경로만 남았다면 그 경로 선택

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
expand( [ Cost, Station | Path ], Paths ) :-  
    findall( [ Cost, NewStation, Station | Path ],  
        ( r( Station, NewStation, _ ),  
          not( member( NewStation, Path ) ) ),  
        L ),  
    update_costs( L, Paths ).
```

- expand predicate는 하나의 경로에서 가장 최신역(노드)를 기준으로 경로에 이미 있는 노드를 제외한(재방문 방지) 갈 수 있는 모든 노드를 찾고 이를 경로에 하나씩 추가하여 갈 수 있는 노드만큼 새 경로 생성
- 생성된 경로들의 총 거리를 update_costs predicate를 사용해 갱신

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
update_costs([],[]):-!.  
update_costs([[Total_Cost,Station1,Station2|Path]|Y],[[NewCost_Total,Station1,Station2|Path]|Z]):-  
    r(Station2, Station1, Distance),  
    NewCost_Total is Total_Cost + Distance,  
    update_costs(Y,Z).
```

- update_costs predicate는 최신역(노드)에 새로 추가된 새로운 역(노드)사이의 거리를 불러와 기존 경로의 총 거리에 더해줌으로서 값 갱신

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
print_path([Station_Destination, Cost], []):- write(Station_Destination), write('.'), write(' '), nl,  
                                                write('경로의 총 거리: '), write(Cost), write(' km'), nl.  
  
print_path([Station1, Station2 | Y], Subways):-  
    sb1(Station1, Station2, Subway),  
    append([Subway], R, Subways),  
    write(Station1), write(', '),  
    print_path([Station2 | Y], R).
```

- 찾은 경로를 원하는 형식으로 출력하기 위해 변형하고 출력하기 위한 부분
- astar predicate에서 찾은 경로를 reverse predicate를 이용하여 원하는 형태로 변경
- 이후 print_path predicate를 사용해 경로 상의 역들을 출발역에서부터 시작하여 하나씩 출력하면서 역간 지하철 호선을 모아놓은 리스트 함수를 생성

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

```
print_subways( [X] ) :- write(X), nl, nl.  
print_subways( [X|Y] ) :-  
    write(X), write(' - '),  
    print_subways(Y).
```

- 앞의 리스트 함수와 `print_subways` predicate를 이용하여
역간 거쳐야 하는 지하철 호선을 출발역에서 도착역 방향으로 하나씩 출력

목적

Knowledge-Base 구축

Prolog 코드


의의 및 한계

실행 예시

- Swish prolog 홈페이지에 코드를 붙여넣고 *shortest_path(출발역,도착역).* 를 기입
- '신촌역 - 이촌역'에 대한 최단경로

'신촌-홍대입구-공덕-효창공원앞-삼각지-이촌'

서울교통공사의 정보와 동일, 최단거리 및 이용 호선도 동일

 `shortest_path(신촌, 이촌).`


최단경로: 신촌, 홍대입구, 공덕, 효창공원앞, 삼각지, 이촌.
경로의 총 거리: 8.2 km
경로의 역간 이용 지하철 호선: 2호선 - 경의중앙선/공항철도 - 6호선 - 6호선 - 4호선


true

Next 10 100 1,000 Stop


?- `shortest_path(신촌, 이촌).`

최소시간	최단거리	최소환승
출발	신촌(지하)	
경유		
도착	이촌(국립중앙박물관)	

 경로탐색

 새로고침

7개 정차역 | 3회 환승
전체 8.2 km | 약 42분 소요



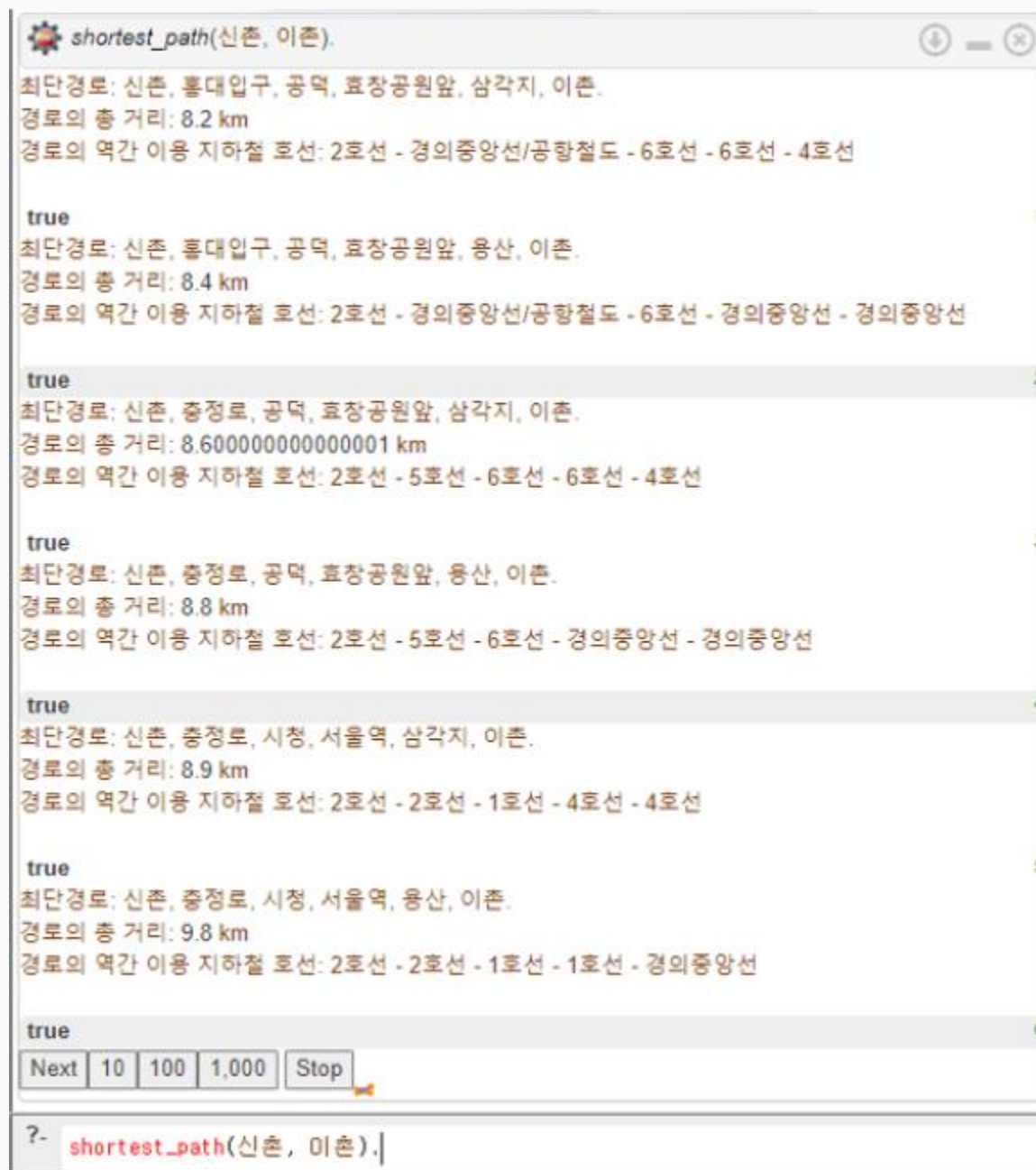
목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

- **프롤로그 결과창의 Next를 활용하여, 최단거리 뿐만 아니라 출발역에서 도착역까지의 가능한 모든 경로 중 N번째로 짧은 경로와 그때의 총 이동거리도 확인 가능**
- **'신촌역-이촌역'에 대해 최단경로 이외의 경로들을 총 거리가 작은 순서대로 제시**



목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

● 실행 예시 영상

The screenshot shows a Google search interface with the query "swish prolog". The search results are displayed below the search bar, showing approximately 128,000 results in 0.28 seconds. The first result is from swi-prolog.org, with the URL "https://swish.swi-prolog.org". The title of the result is "SWISH -- SWI-Prolog for SHaring", and the snippet below it says "Use the Examples menu from the navigation bar · Use the Program or Notebook button above." Below the title, there are several links: "SWISH" (with a snippet "Type, Name, Tags, User, Modified. puzzleEM, Erika Mackin, 2015 ..."), "Prolog-family-tree" (with a snippet "Example programs; Prolog tutorials; SWISH tutorials ..."), "Example" (with a snippet "Knowledge bases - Lists - Sudoku (clp(fd)) - GraphViz renderer"), and "Prolog tutorials" (with a snippet "This notebook collects Prolog tutorials provided as SWISH ..."). At the bottom of the search results, there is a link "swi-prolog.org 검색결과 더보기 >". The bottom of the screenshot shows the address bar with the URL "https://swish.swi-prolog.org" and the text "SWI-Prolog".

목적

Knowledge-Base 구축

Prolog 코드

의의 및 한계

- '샛강 - 효창공원앞'에 대한 최단경로
코드 '샛강-노량진-용산-효창공원앞' vs 서울교통공사 '샛강-여의도-공덕-효창공원앞'
➡ 두 가지의 경로의 거리의 합은 5.3(km)으로 동일
➡ 작성된 코드가 정상적으로 작동

```
shortest_path(샛강,효창공원앞).  
최단경로: 샛강, 노량진, 용산, 효창공원앞.  
경로의 총 거리: 5.3 km  
경로의 역간 이용 지하철 호선: 9호선 - 1호선 - 경의중앙선  
  
true  
?- shortest_path(샛강,효창공원앞).
```

최소시간	최단거리	최소환승
출발	샛강	
경유		
도착	효창공원앞	

경로탐색

새로고침

5개 정차역 | 2회 환승
전체 5.3 km | 약 21분 소요



목적

Knowledge-Base 구축

Prolog 코드

▣ 의의 및 한계



- A* 알고리즘을 활용하며 h-value를 역간 직선거리로 설정, 역간 실제 거리를 활용
➡ 최단거리에 기반한 최단경로를 제공하는 시스템 구현
- 기존 : 다익스트라 알고리즘을 이용한 최단경로 길 찾기 기능
➡ 본 프로젝트 : **A* 알고리즘**으로 최단경로 길 찾기 기능 실현
- 프롤로그의 Next 기능을 활용, 경로의 총 거리를 기준으로 출발지로부터 목적지까지
가능 경로 중 최단 경로뿐만 아니라 **N번째로 짧은거리 경로** 또한 제시 가능

목적

Knowledge-Base 구축

Prolog 코드

▣ 의의 및 한계



- 최단경로 제공에 있어 **최단거리**만을 고려
- 실제 길 찾기에 있어 거리만큼 중요한 요소 "**시간**"
 - ➡ 시간에 대한 고려 부족
- 이와 같은 문제를 보완 위해 역간 환승거리, 배차간격, 혼잡도, 시계열 등 고려
 - ➡ **최단시간**의 판단 근거로 활용



감사합니다