**인공지능 HW2** -이수원 교수님-

# 20150439 소혜빈

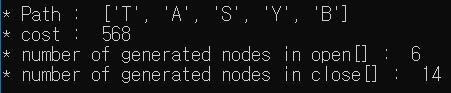
제출 날짜 : 2019.4.14

사용언어 : Python3.7.3

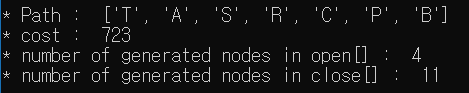
## 첨부 자료에서 Timisoara에서 Bucharest로 가는 path를 찾는 알고리즘을 다음 4가지 기법들을 사용하여 구현하고, 다음 질문에 답하시오

## 질문:

### Breadth-first search에 의하여 **가장 먼저** 생성된 path의 cost 및 생성된 node의 수는? (Closed, open 각각에 대하여 포함되었던 노드의 수)



### (b) Depth-first search에 의하여 **가장 먼저** 생성된 path의 cost 및 생성된 node의 수는? (Closed, open 각각에 대하여 포함되었던 노드의 수)

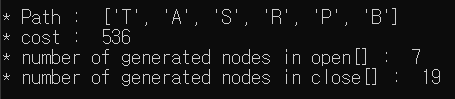


### (c) **Shortest path**및 그 cost는?



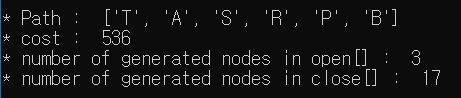
### (d) Shortest path를 찾는 경우, 각각의 알고리즘에 의하여 생성되는 node의 수는? (Closed, open 각각에 대하여 비교하시오)

Breadth-first search :



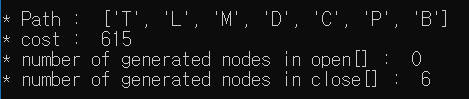
19개의 노드를 방문하고 optimal solution을 찾았다.

Depth-first search :



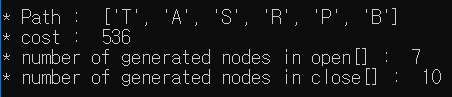
17개의 노드를 방문하고 optimal solution을 찾았다.

Hill-Climbing :



Local maxima 문제에 빠져 shortest path를 찾지 못했다.

Best-first search :



단 10개의 노드를 방문하고 가장 먼저 optimal solution을 찾았다.

### (e) 결과를 분석하시오 (discussion)

Breadth-First Search 나 Depth-First Search 의 경우, brute-force 방식으로서 가장 먼저 찾는 path가 optimal solution이라는 보장은 없으나 모든 path를 다 찾으므로 결국 그 중 하나는 optimal solution이다.

반면, Hill-Climbing 이나 Best-First Search 의 경우, Heuristic Evaluation Function( F = G+H)을 사용한다. 특히, Best-First Search 는 open 리스트를 F(최소)값을 기준으로 priority Queue로 관리하며 최적의 경로를 계속 업데이트하여 search하고, 최적화 순서대로 모든 path를 다 찾는다. 언제나 가장 먼저 발견하는 답이 optimal solution 이며 그러므로 Best-First Search 는 admissible 하다.

그러나 Hill-Climbing 의 경우, greedy approach 방식으로 Local maxima 문제가 발생하여 optimal solution을 찾지 못하고 search가 끝날 수 있다.

## 엑셀에서 도시 정보 읽어 오기

**\*도시 이름의 첫 대문자만 사용**

**\*코드 내 주석 참고**

**\* citys**

Type : dict {key : value}

key : 도시 이름의 첫 대문자 (str)

value : Bucharest까지의 직선거리 (int)

**\* nodes**

Type : dict {key : value}

key : 현재도시 이름의 첫 대문자 (str)

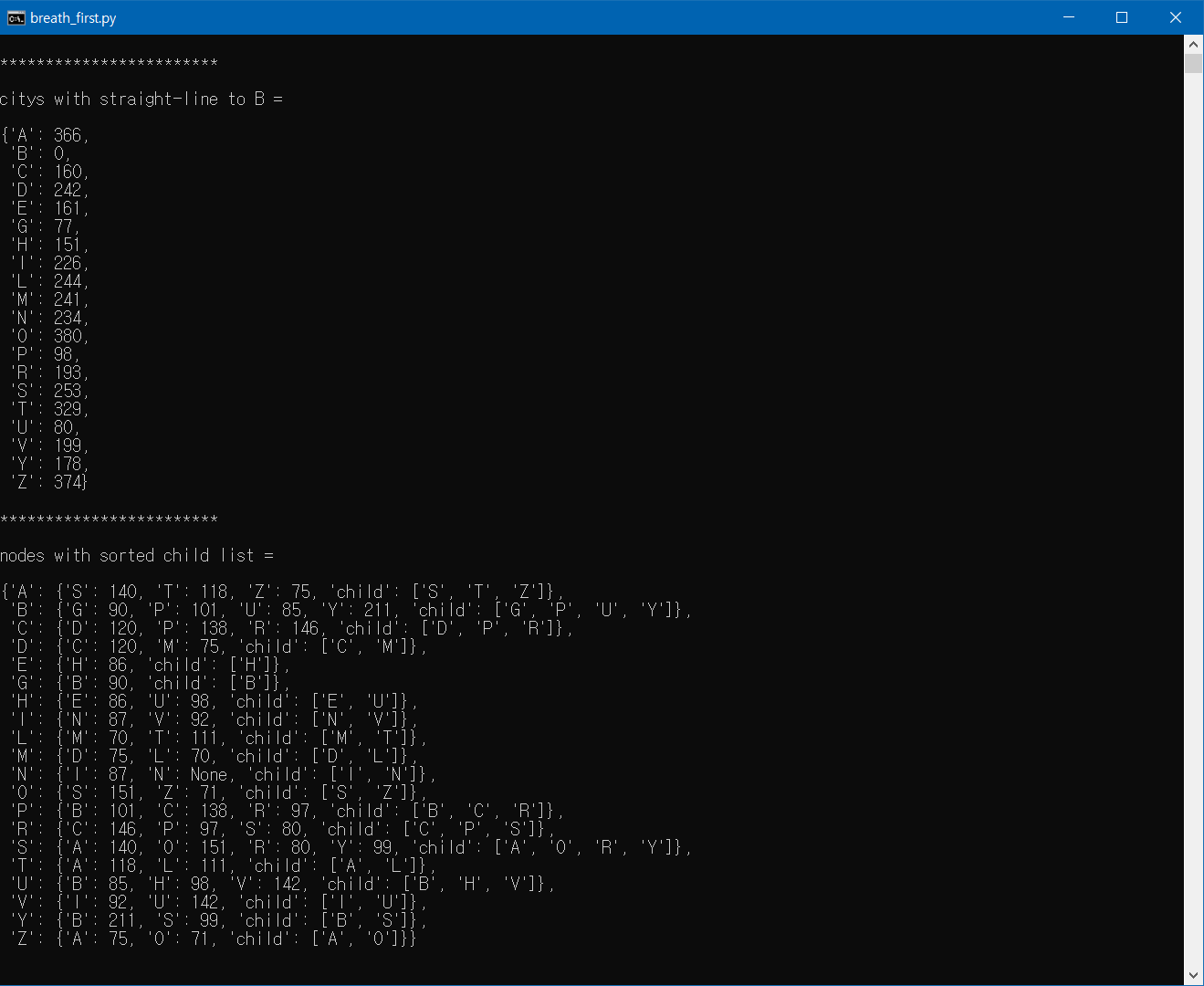
value : {key’ : value’ , … , ‘child’ : [ sorted key’ list ] }

key’ : 현재도시의 이웃한 도시 이름의 첫 대문자 (str)

value’ : 현재도시에서 이웃한 도시까지의 비용 (int)

sorted key’ list : sorting 된 key’들의 list

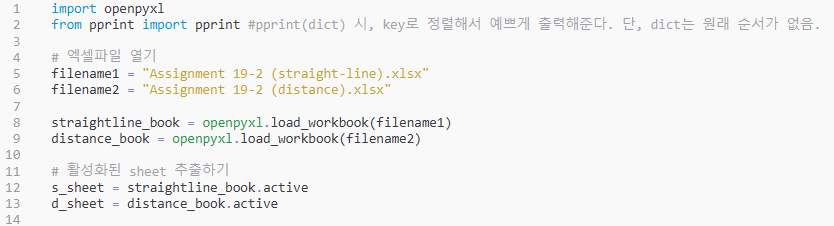
**\*읽어온 엑셀파일을 각각 citys 와 nodes에 담아서 pprint로 출력한 결과화면**



**[source code]**

**\* s\_sheet :** straight-line 정보가 담긴 엑셀파일의 활성화된 sheet

**\* d\_sheet :** distance 정보가 담긴 엑셀파일의 활성화된 sheet



**\* s\_sheet -> citys**



**\* d\_sheet -> nodes**

nodes1 :

A열에 입력된 도시를 nodes의 key로, B열에 입력된 도시를 nodes의 value의 key’로

nodes2 :

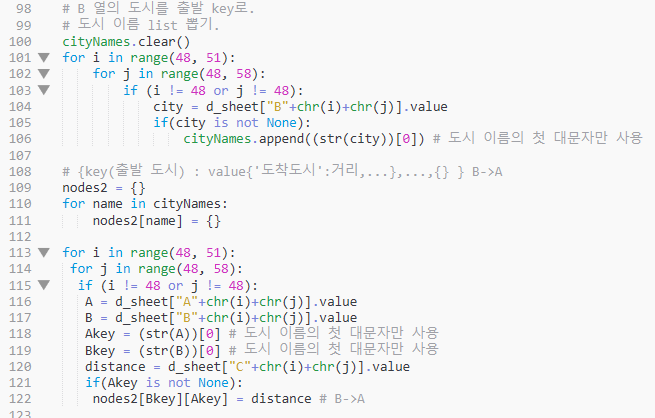
B열에 입력된 도시를 nodes의 key로, A열에 입력된 도시를 nodes의 value의 key’로

nodes :

nodes1과 nodes2를 중복된 key 에 대해 key’를 update하고,

nodes1에 없는 key에 대해 nodes2의 key를 추가하며 합친다.

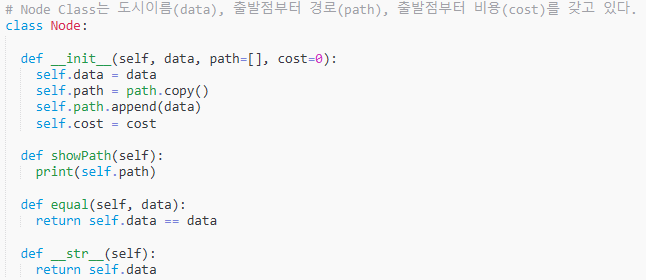






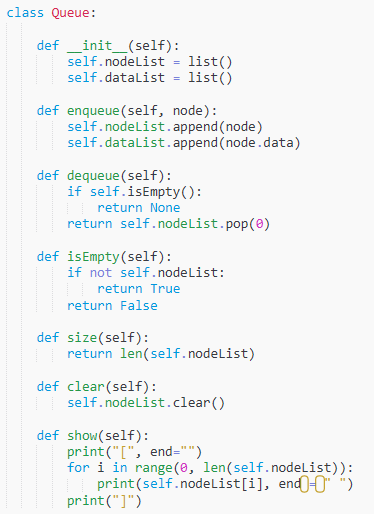
## RUN 결과

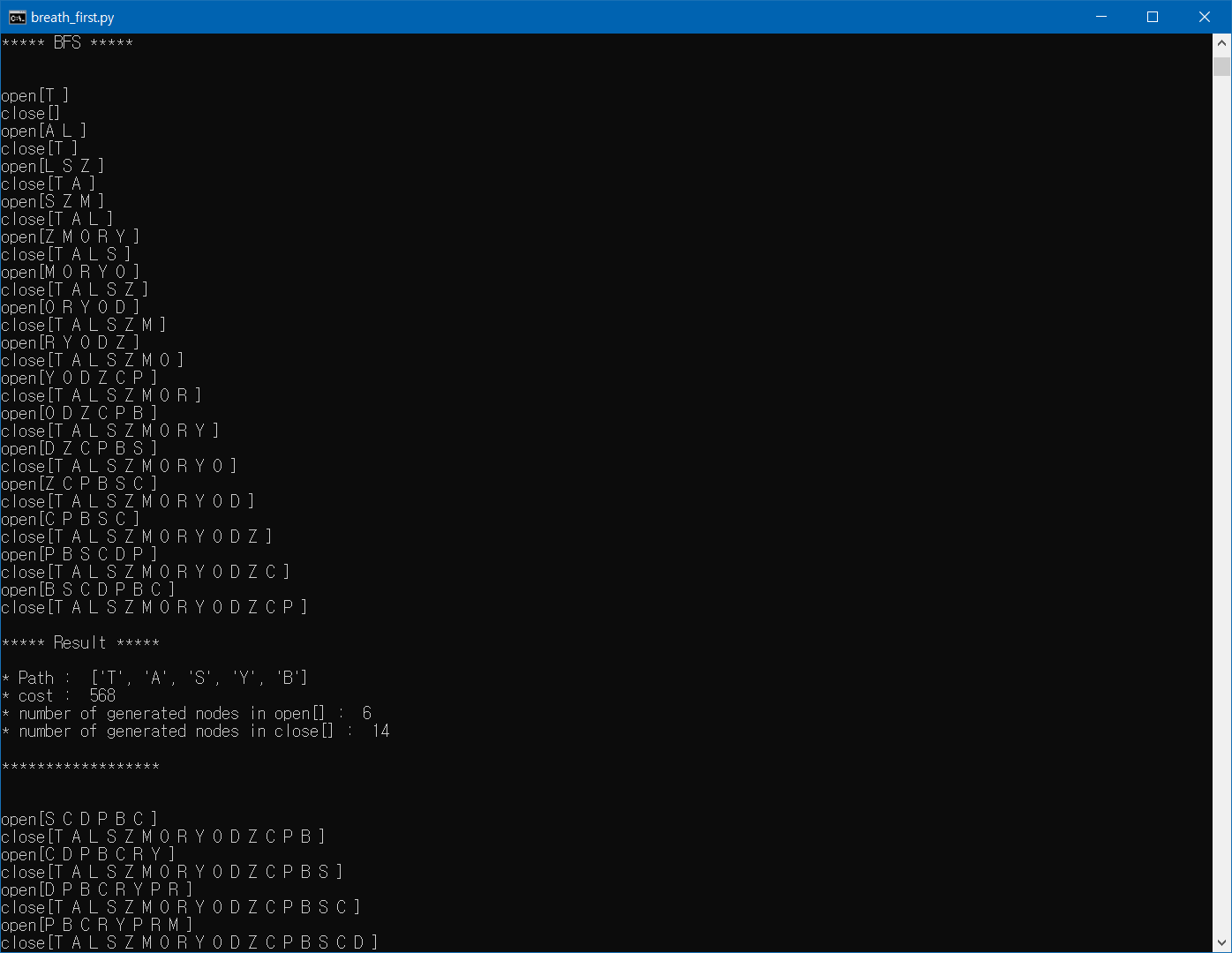
\* 모든 알고리즘에서 Node class 를 아래와 같이 정의하여 사용하였다.

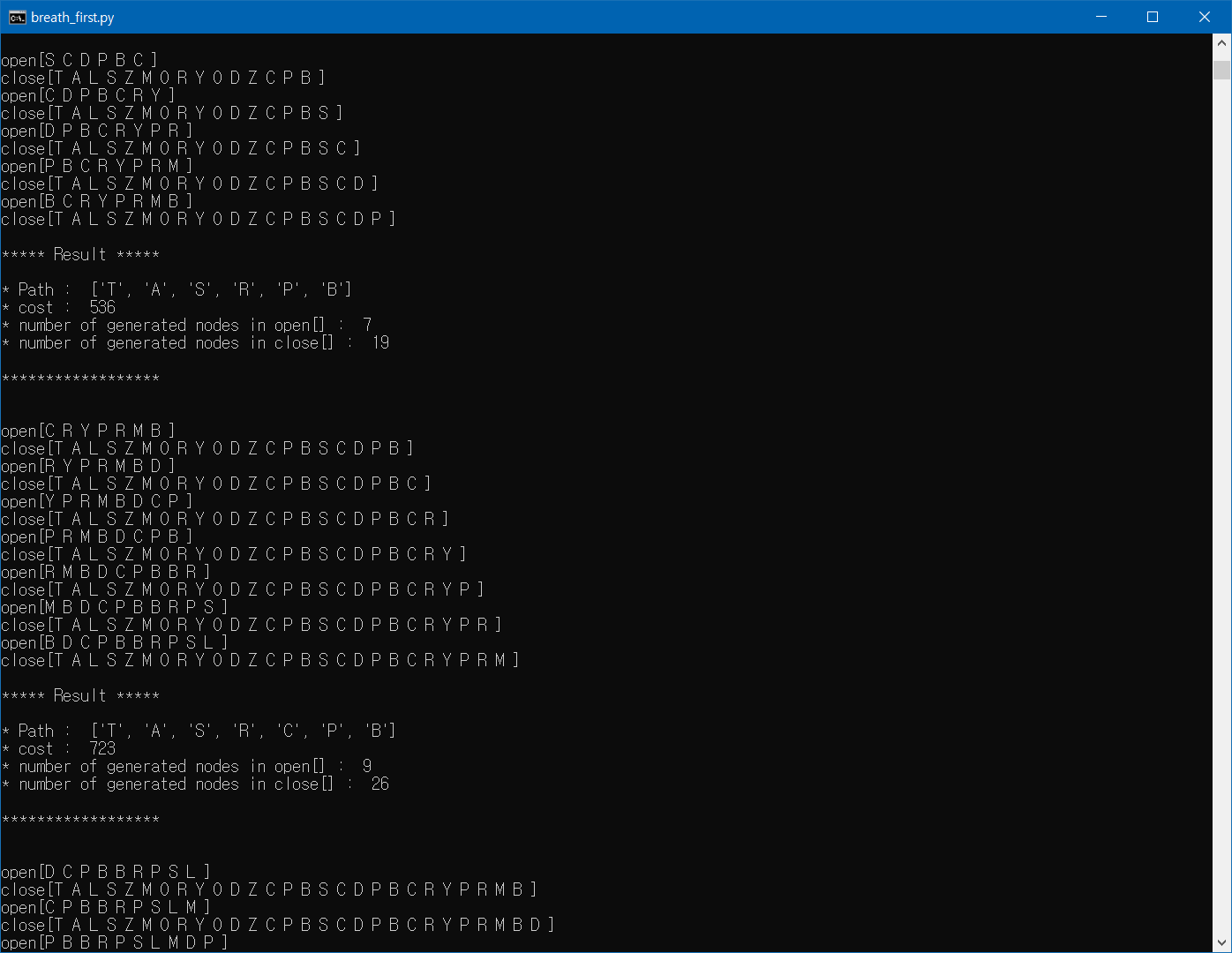


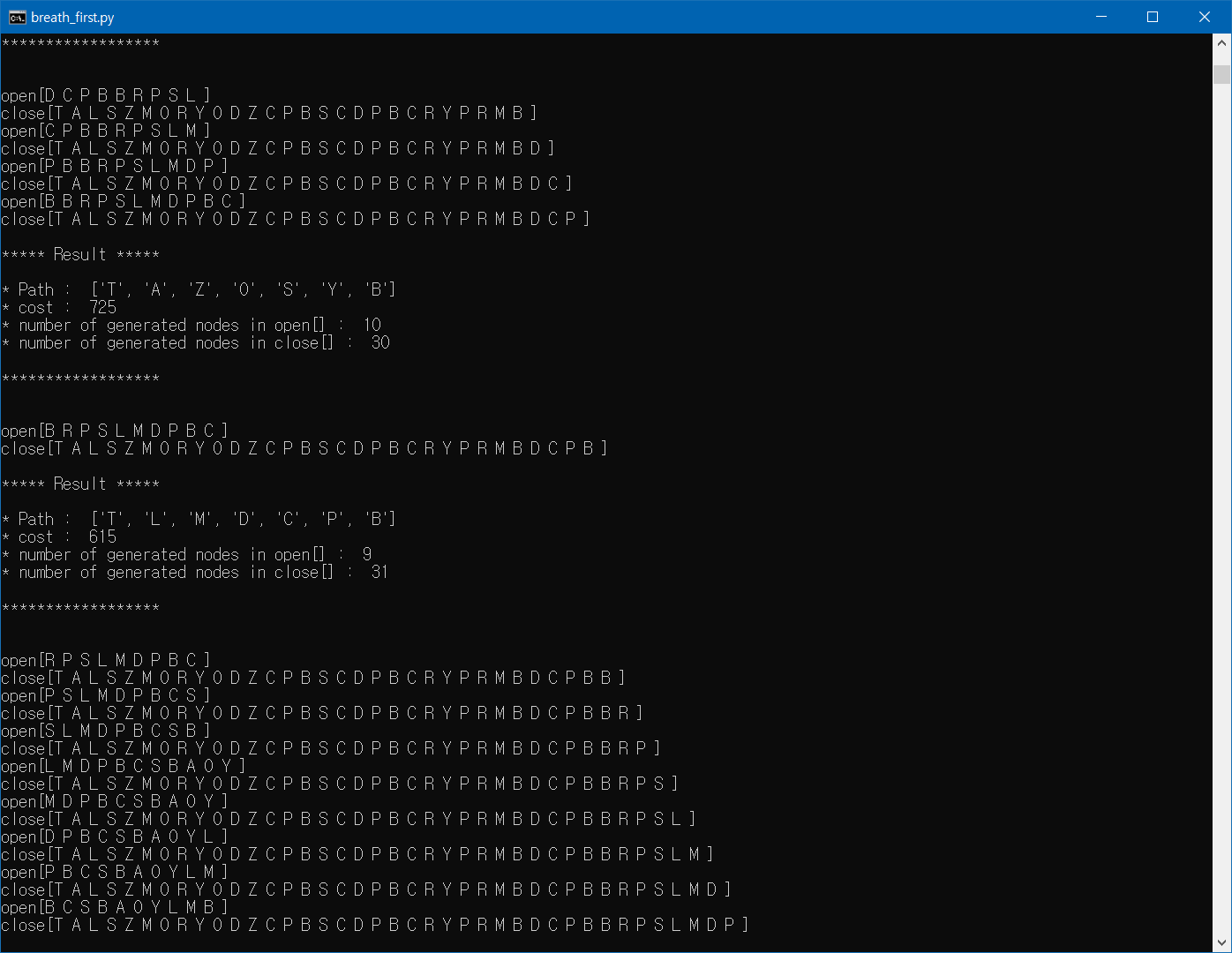
1. **Breadth-first search**

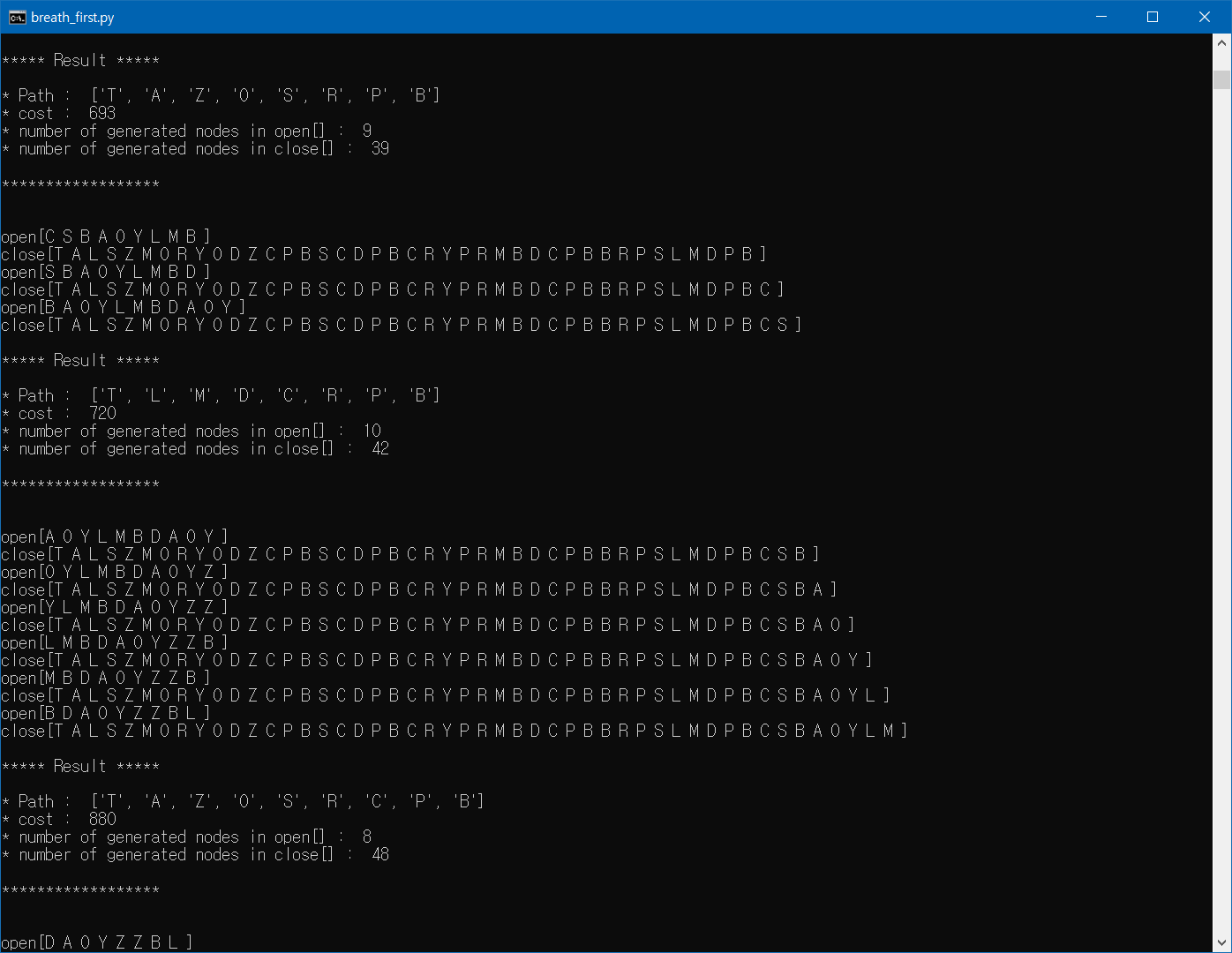
\* open/close : Queue 사용

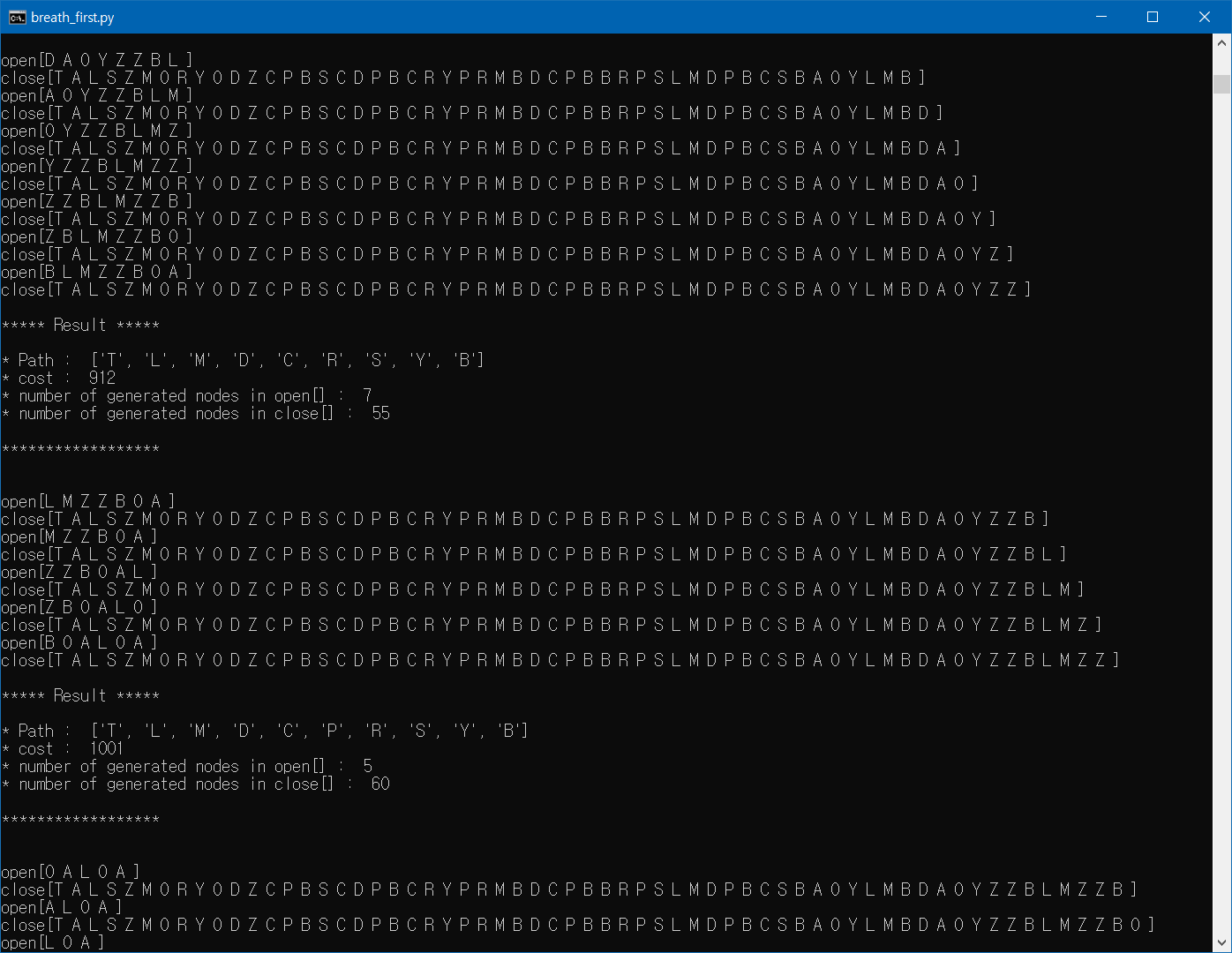


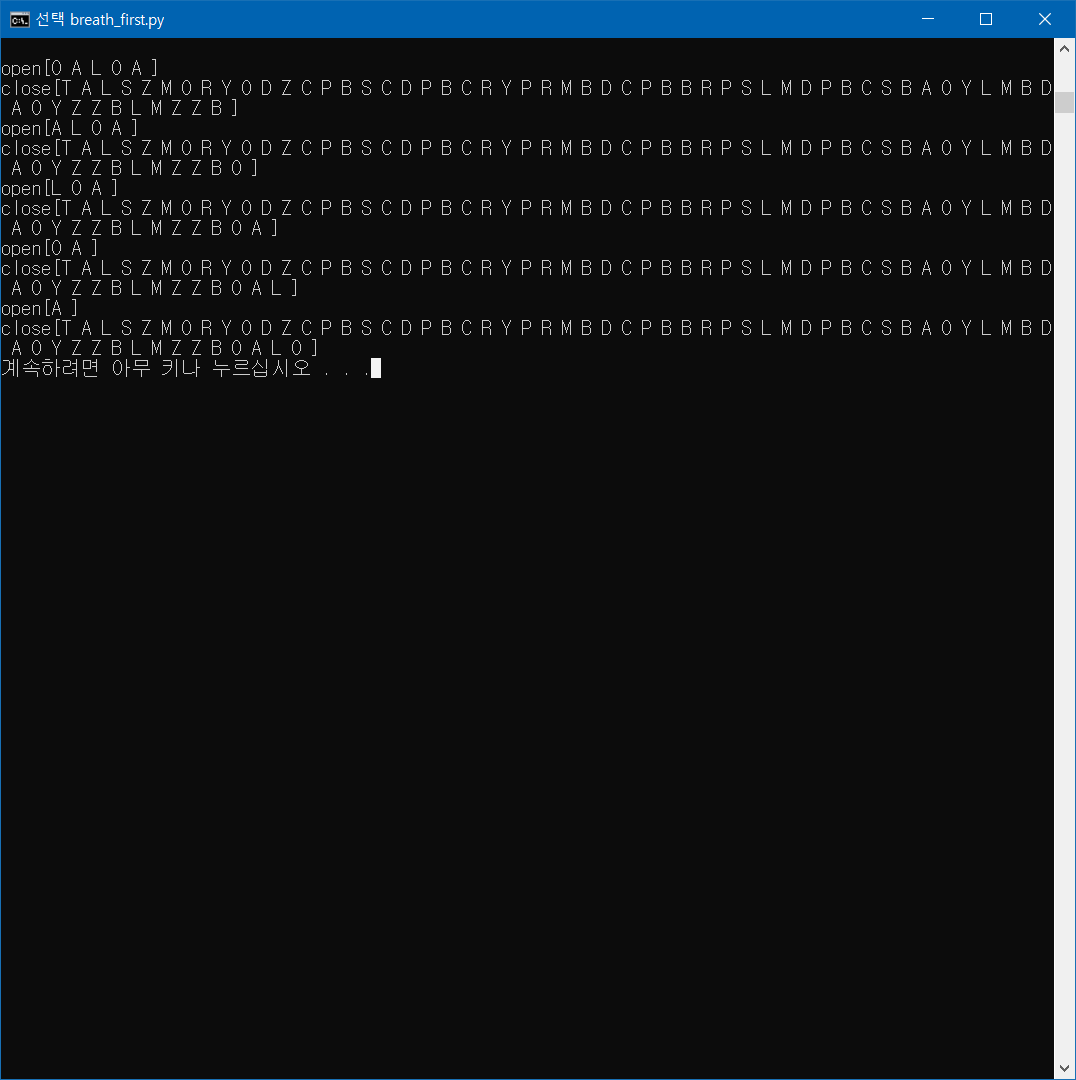






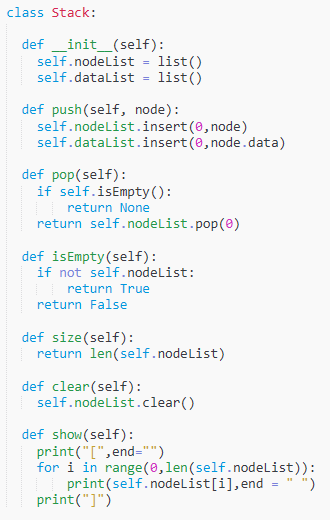


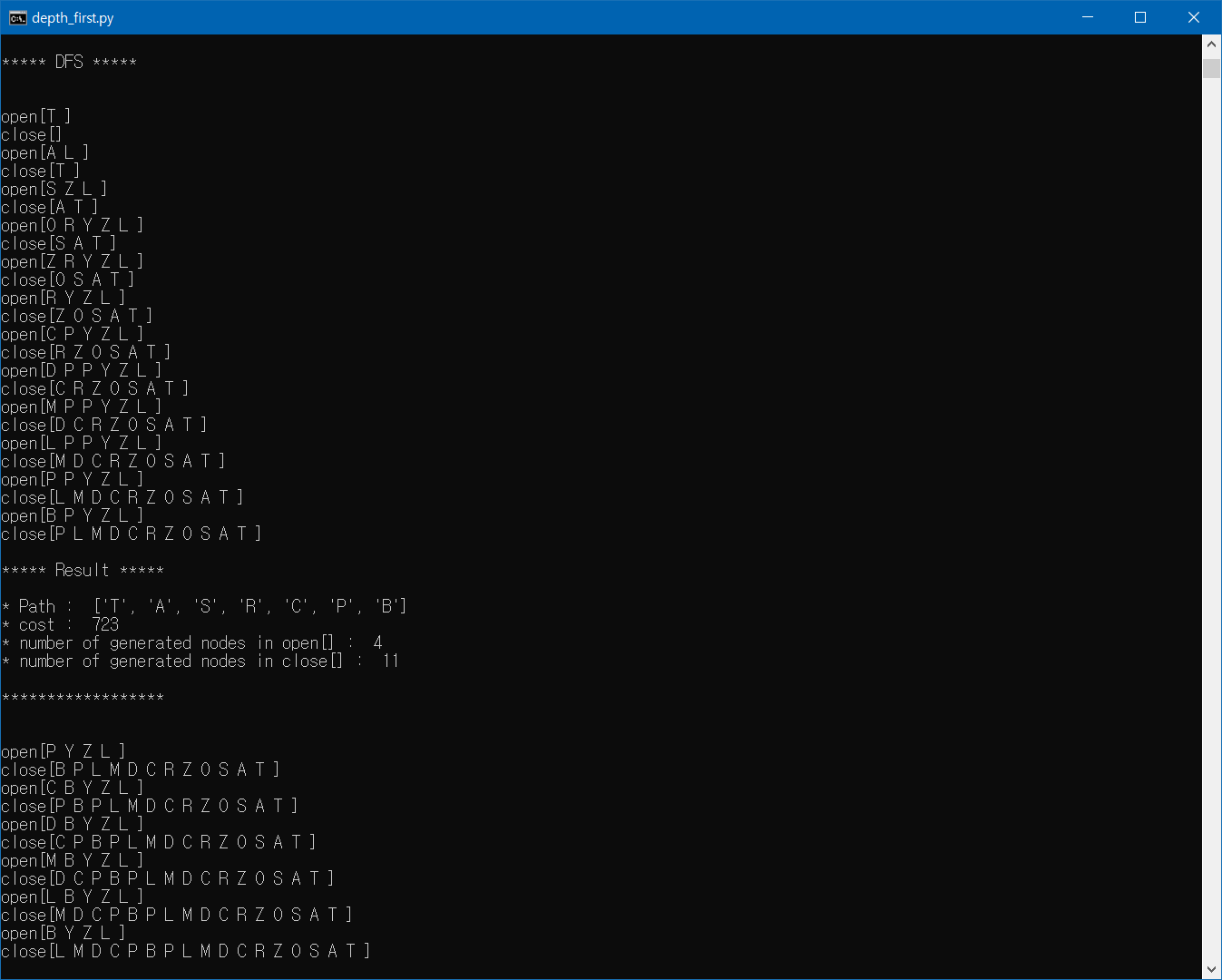


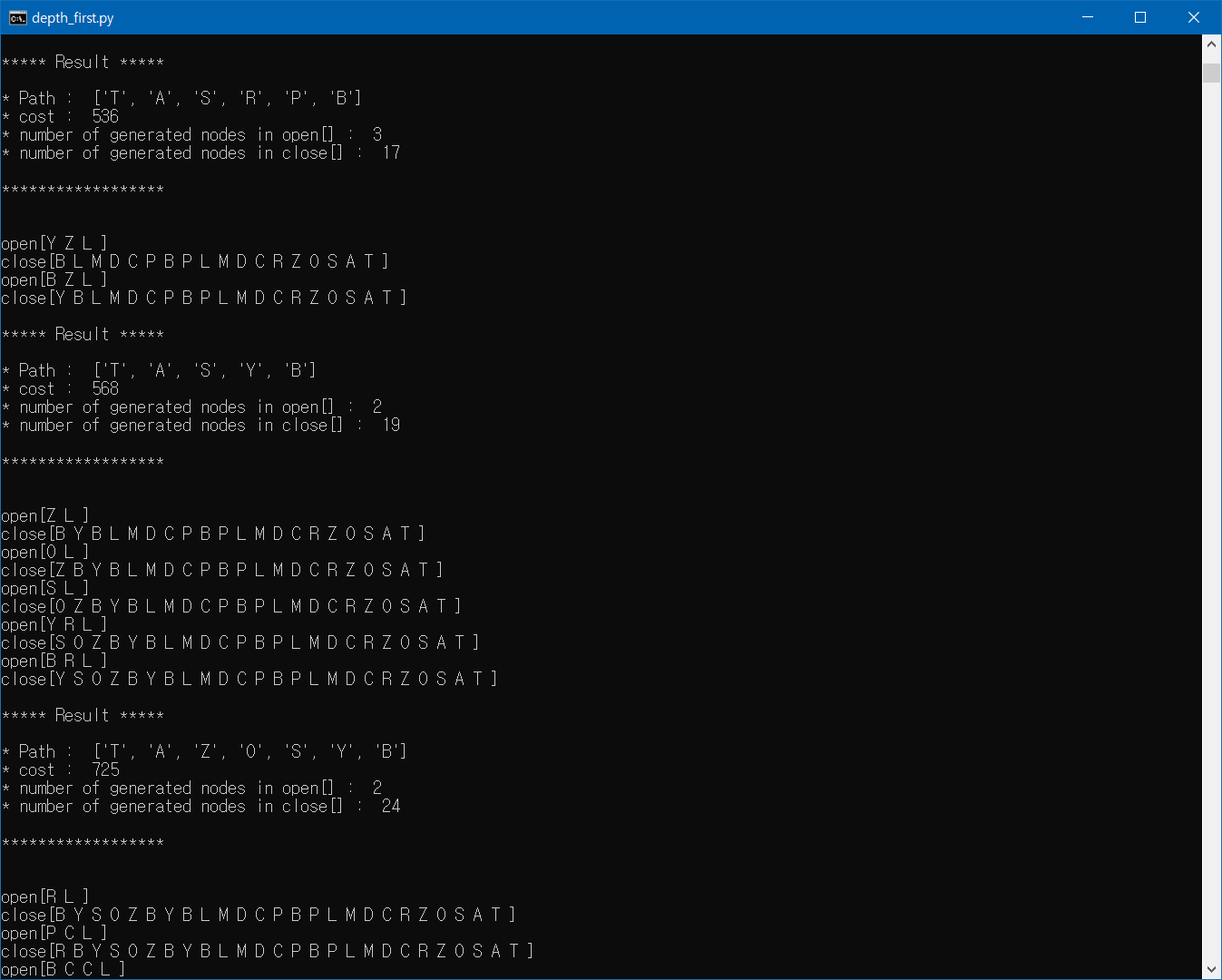


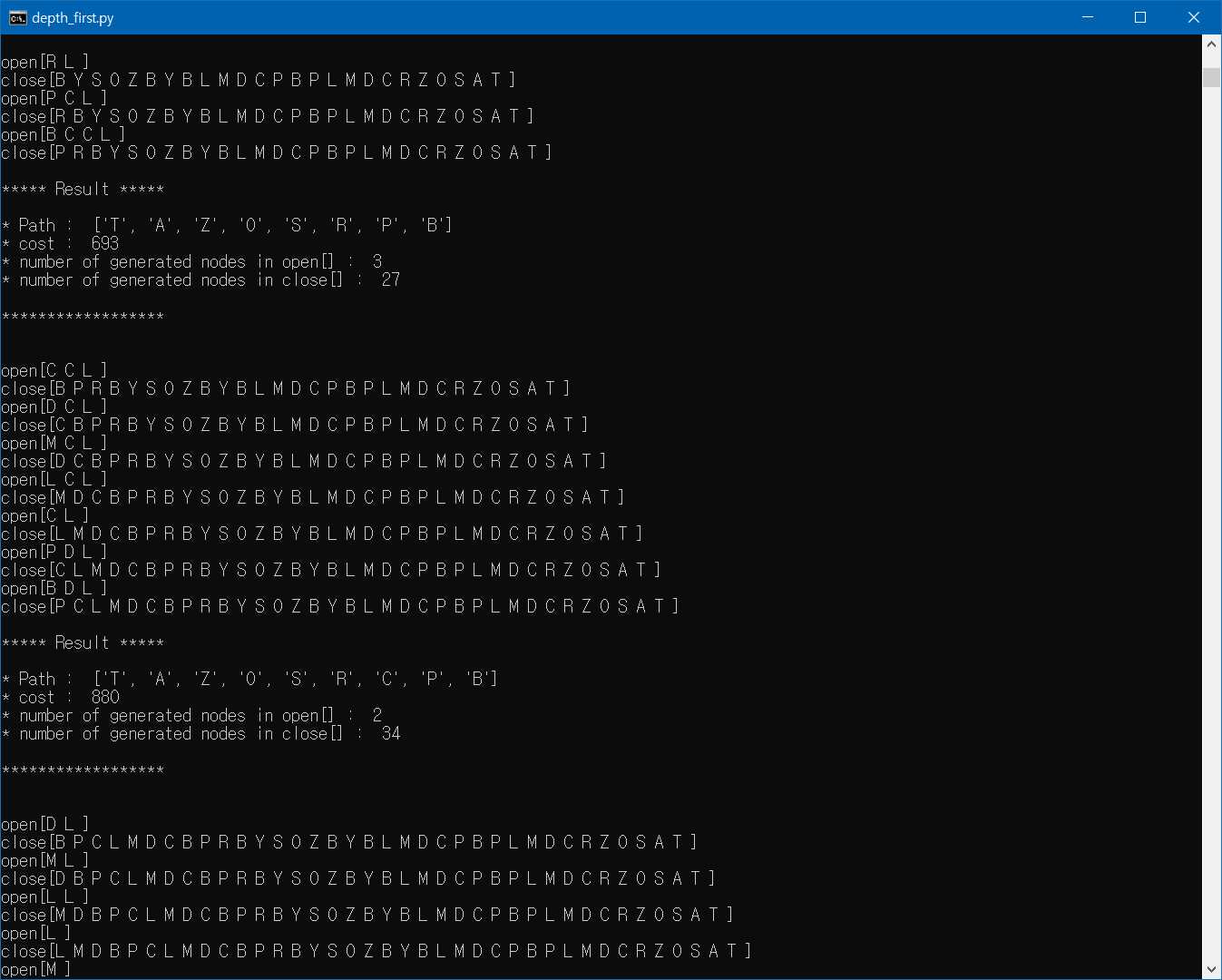
1. **Depth-first search**

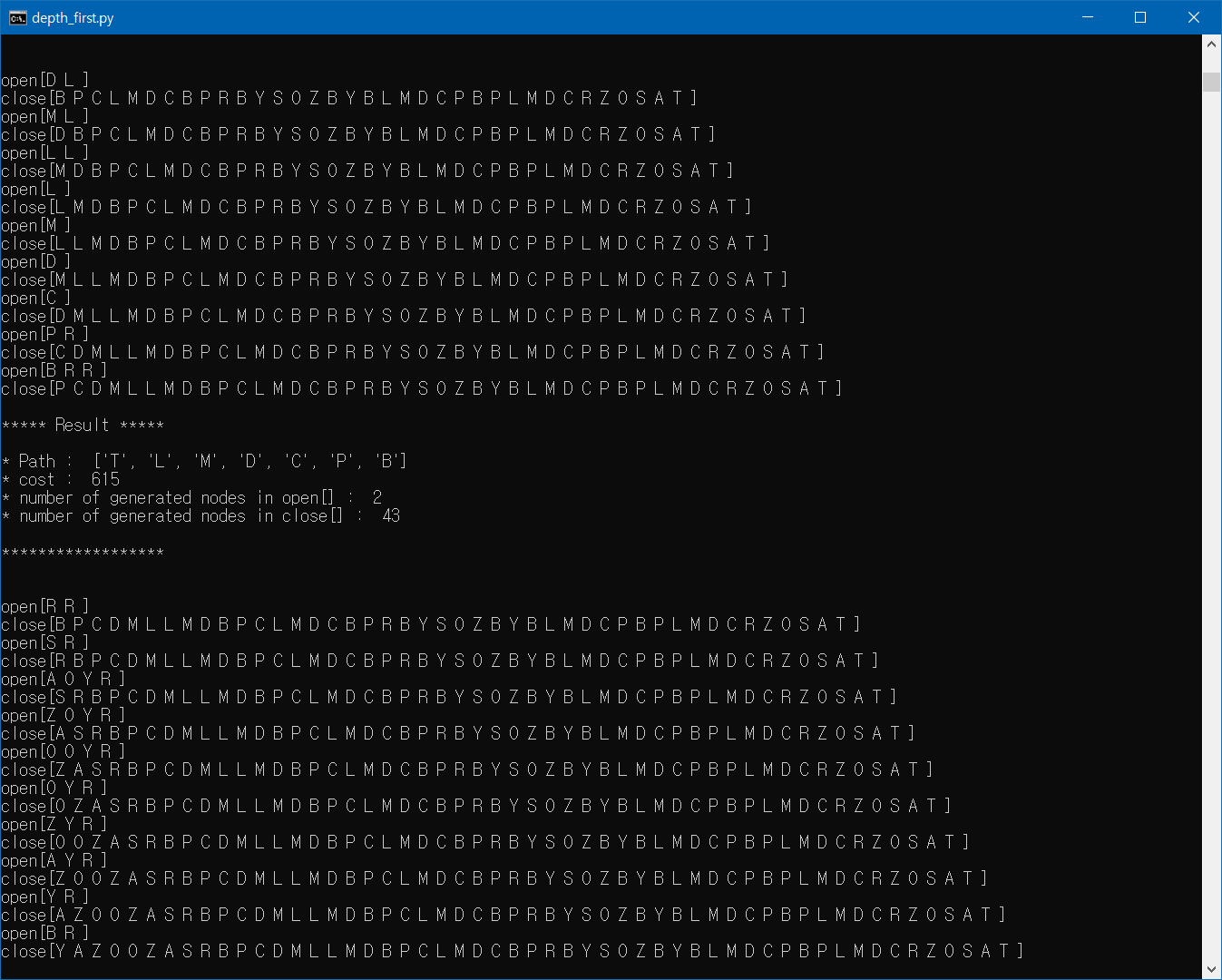
\* open/close : Stack 사용

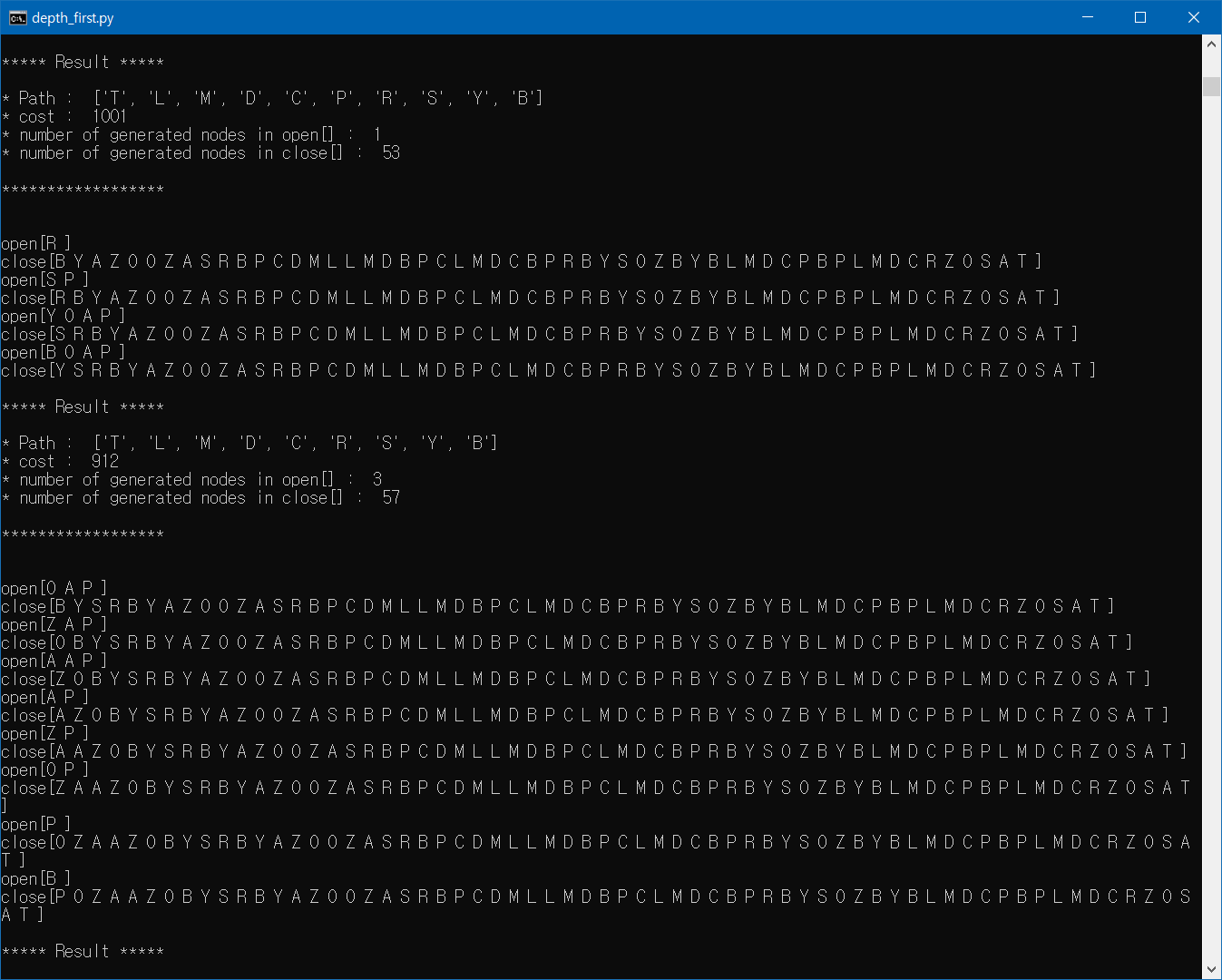








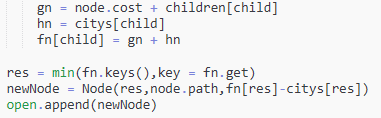






1. **Hill-Climbing**

\* open/close : list 사용



현재 자식 node중에서 f(n) 값이 가장 작은 것을 선택하고 나머지 경우는 버림.

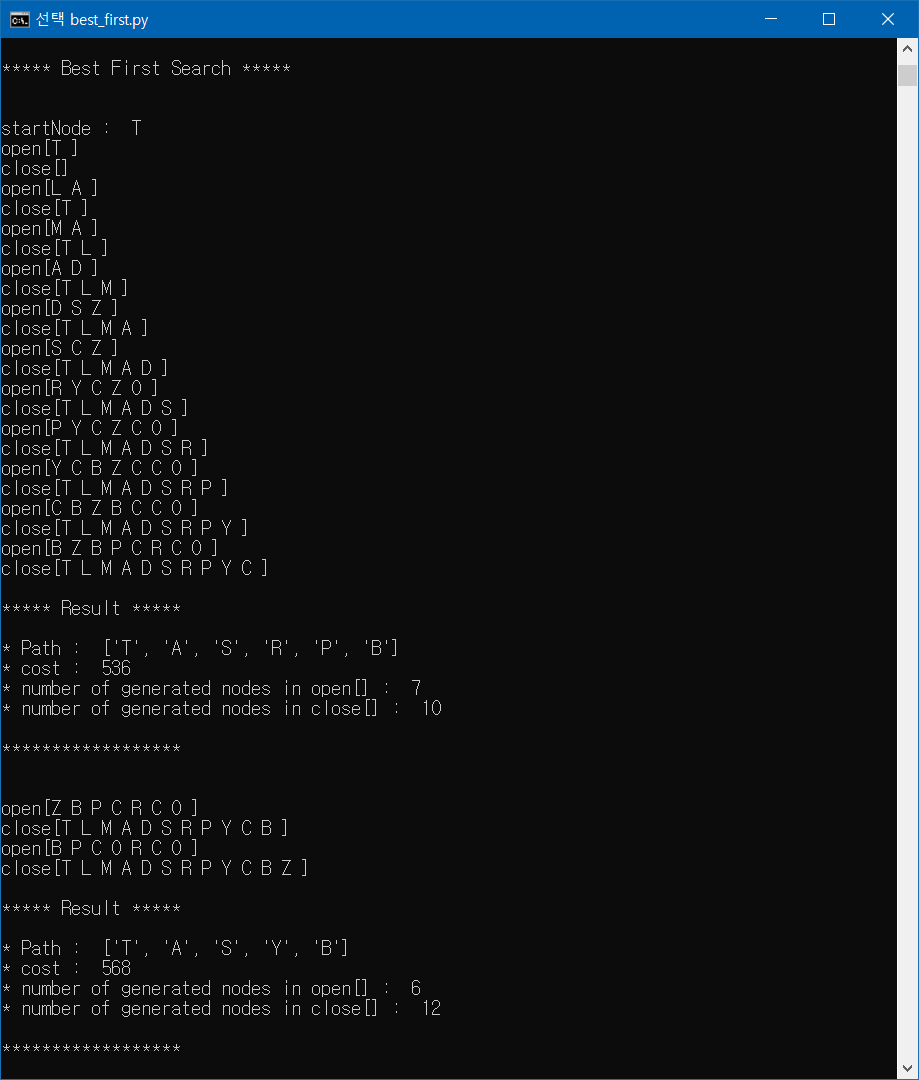


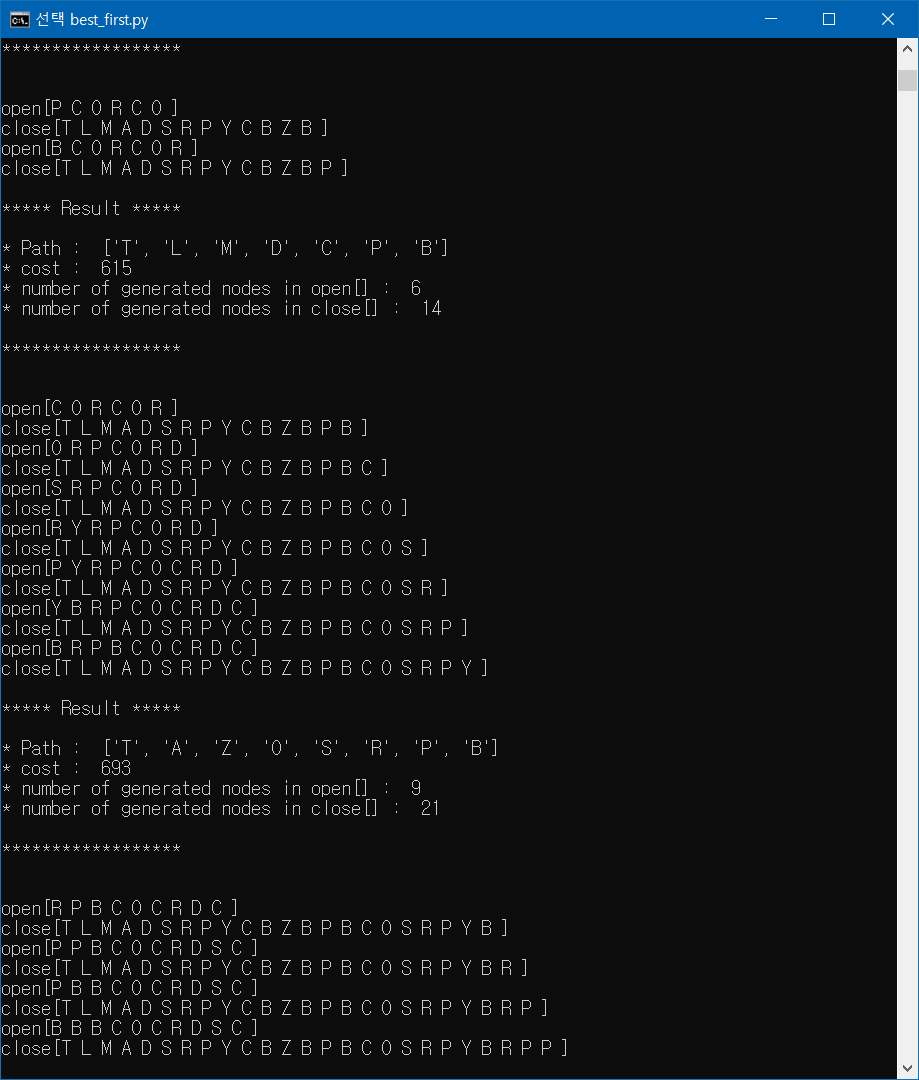
1. **Best-first search**

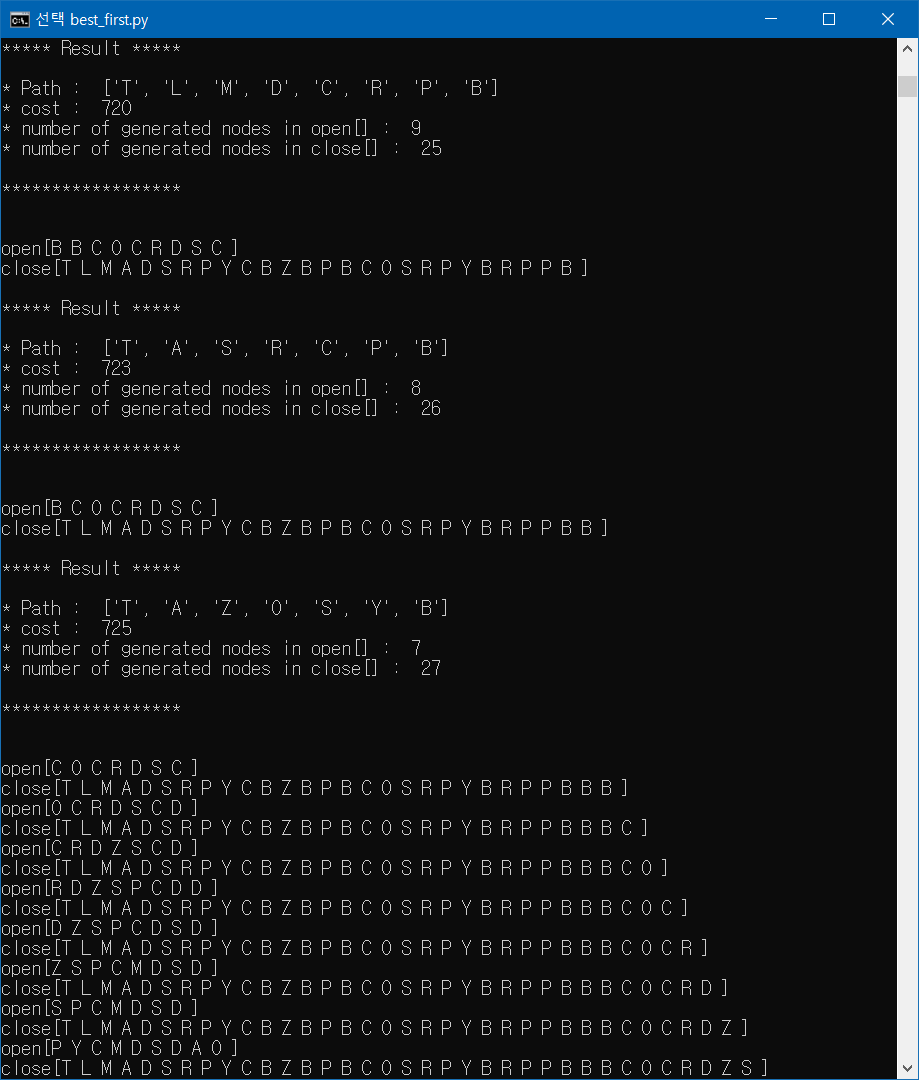
\* open : priority Queue 사용

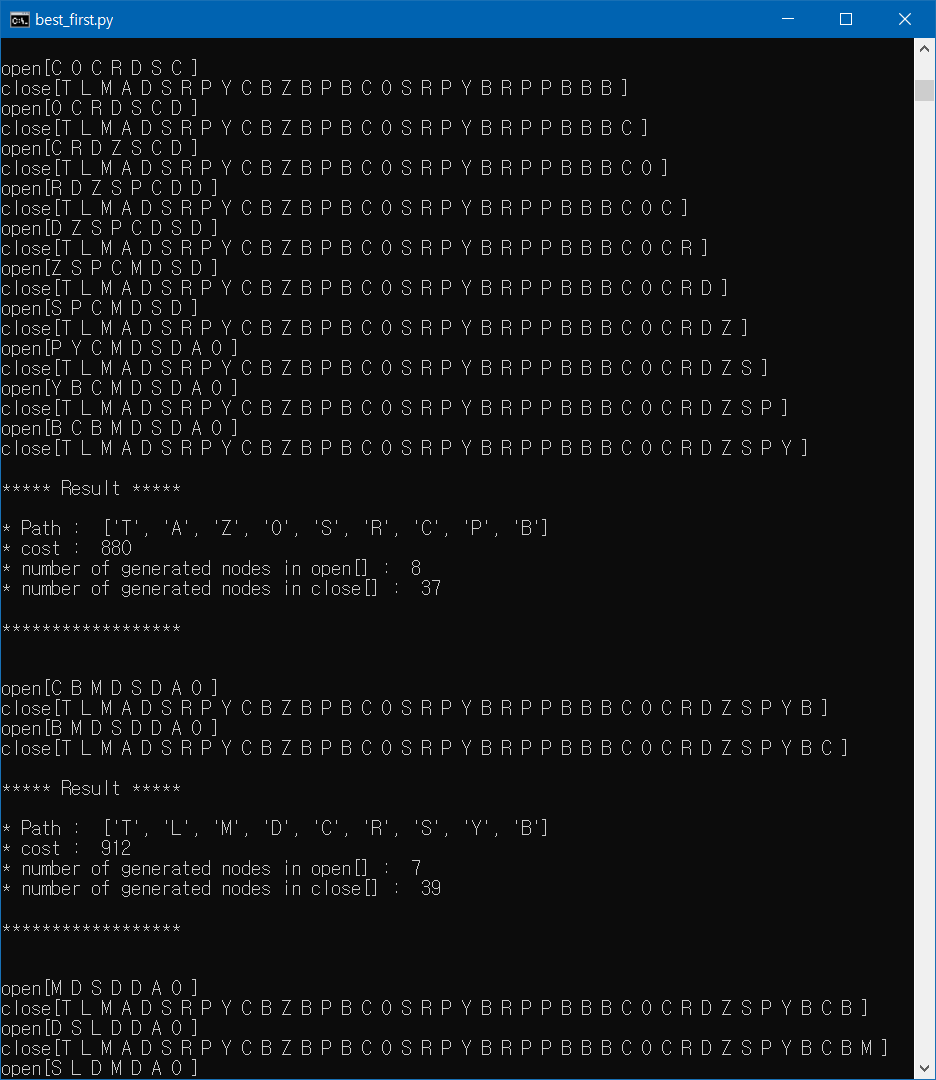
\* close : list 사용

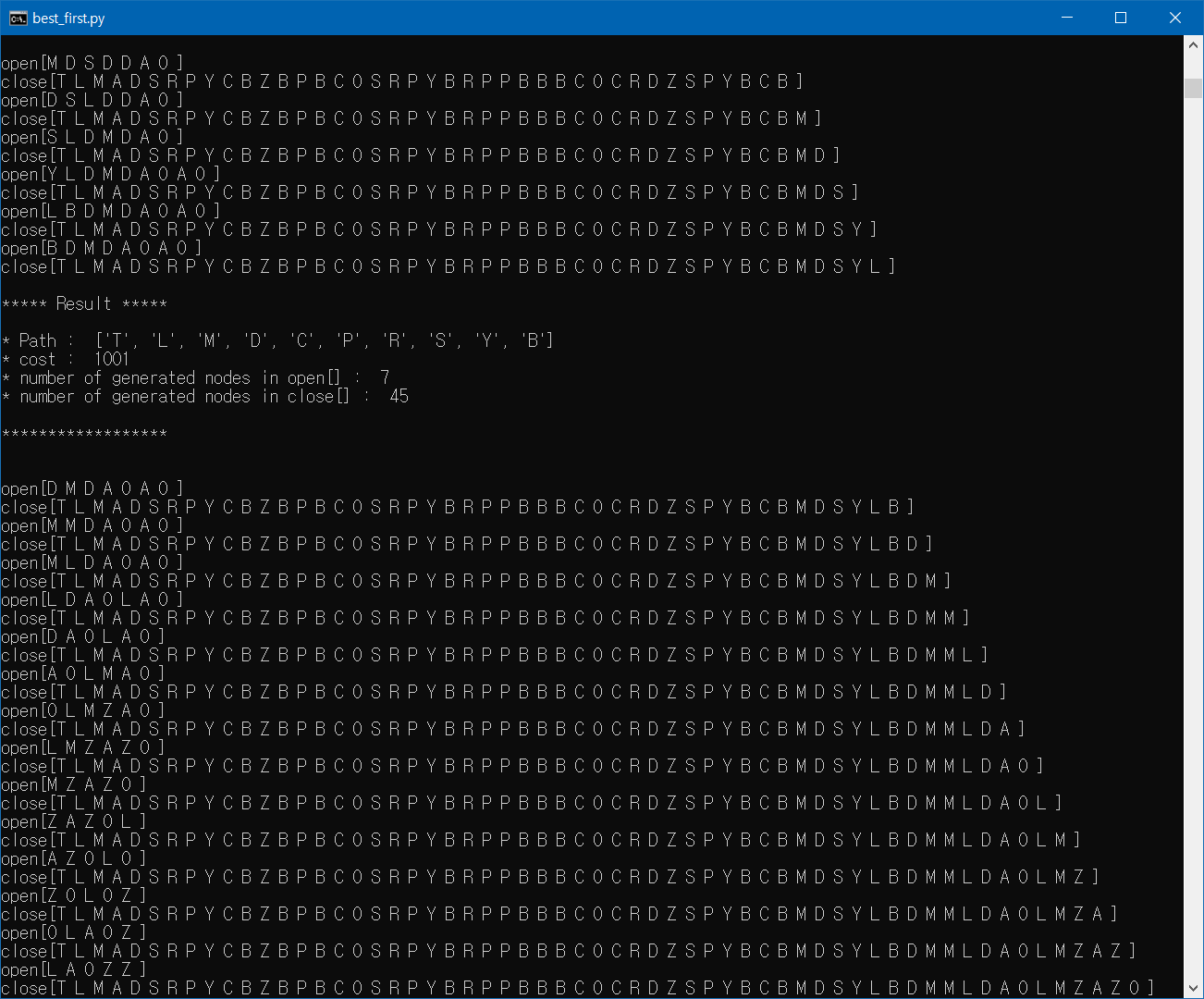


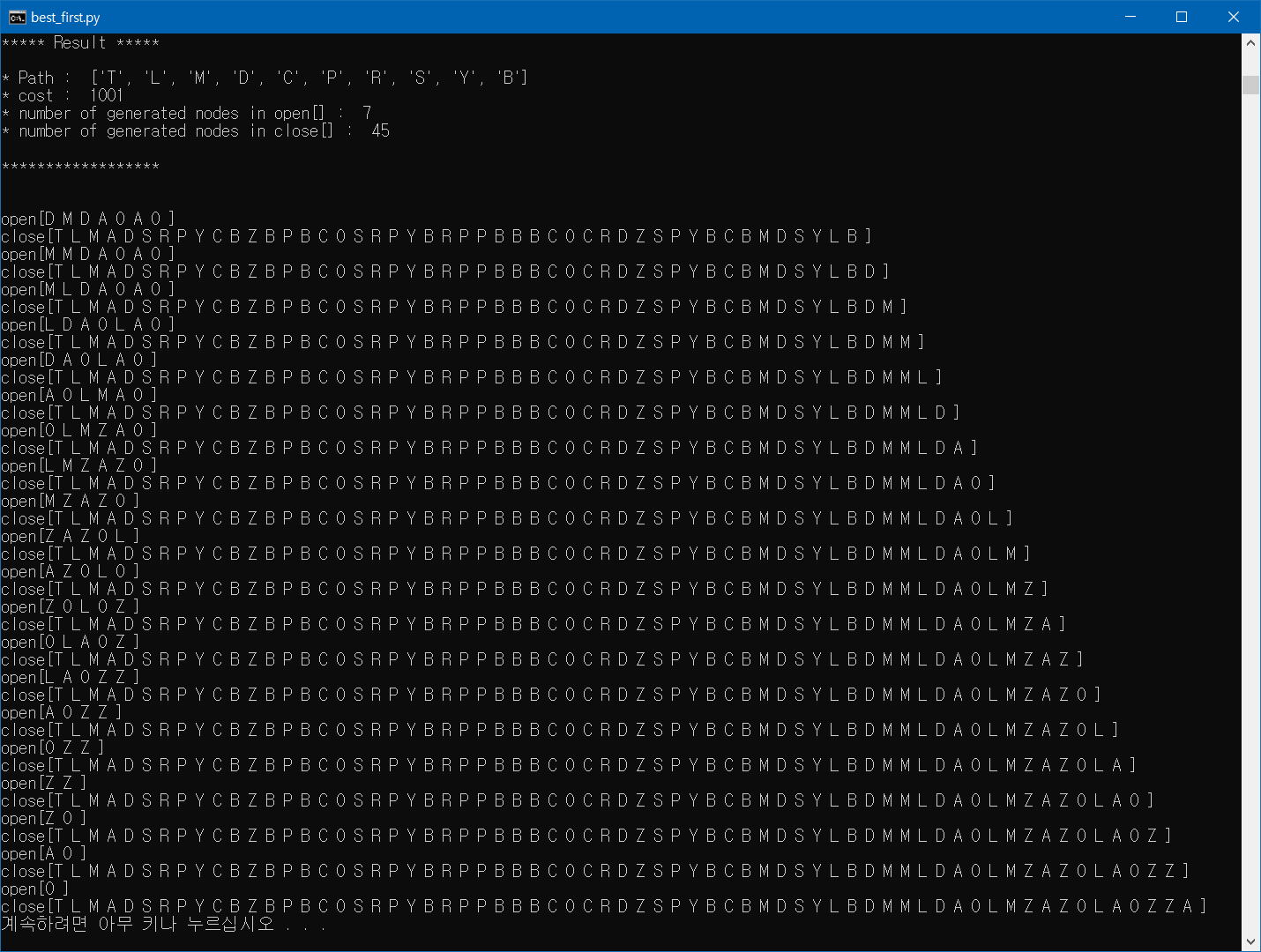






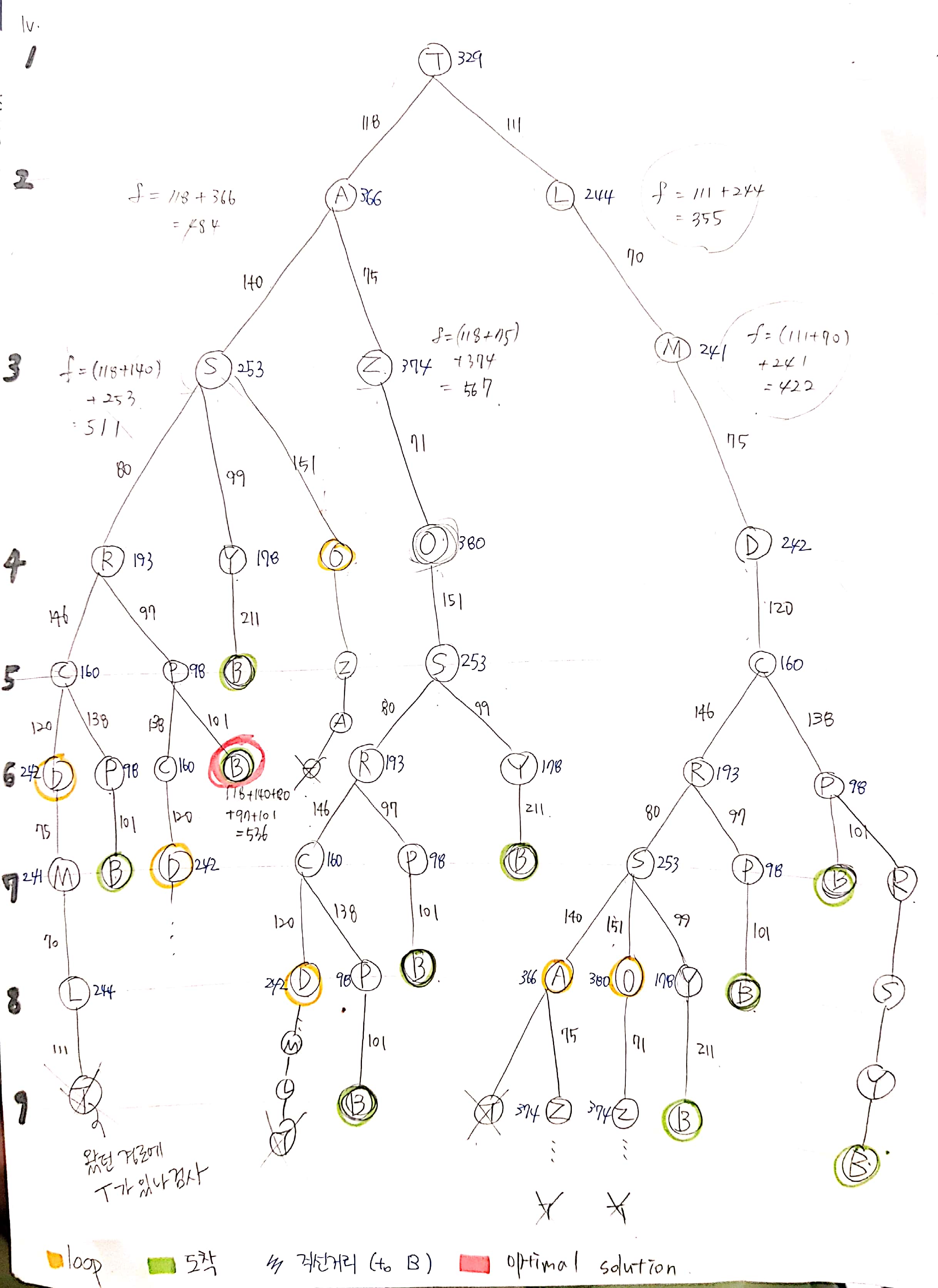






## 부록

[손풀이]



[참고]

1. [파이썬 - 기본을 갈고 닦자!](https://wikidocs.net/book/1553) <https://wikidocs.net/book/1553>
2. Python 3.7.3 documentation <https://docs.python.org/3/index.html>
3. 한권으로 배우는 파이썬 기초&알고리즘 사고법 -[아리스티데스 보우라스](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=&auth_no=223398), [루키아 아이나로지두](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=&auth_no=223399) 저/[길준민](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=&auth_no=208462), [임종범](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=&auth_no=223400), [송의성](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=&auth_no=223401) 역 외 1명/[제이펍](javascript:void(0);)
4. 인공지능 수업 강의자료 3장, 4장
5. 개발환경 구성 <https://webnautes.tistory.com/454>
6. openpyxl 사용법 <https://dejavuqa.tistory.com/25>