

알고리즘설계 설계 프로젝트

n-Queens Problem

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2020년 6월 27일

학부 정보통신공학과

학년 4

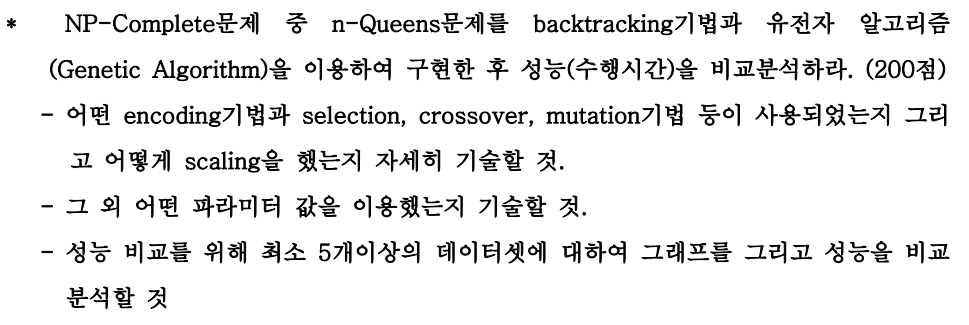
성명 이정우

학번 12171833

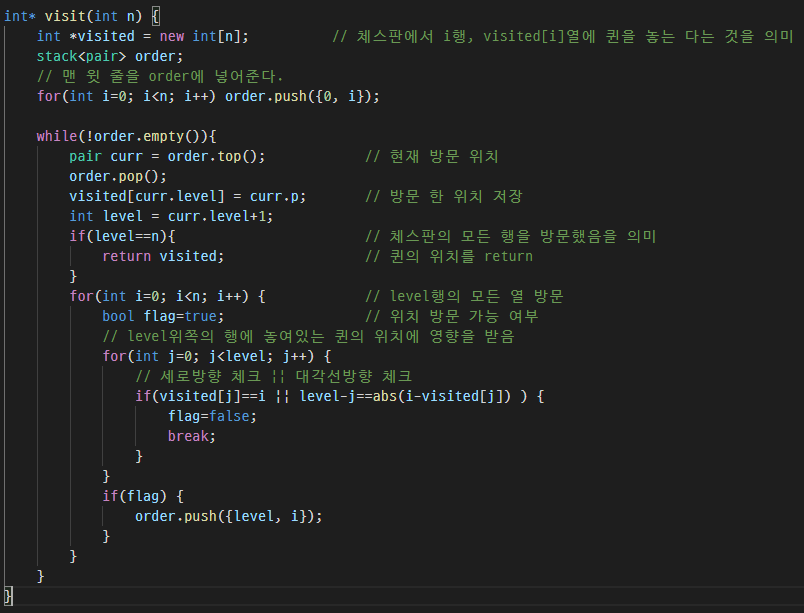


최소한 아래의 내용을 포함할 것.

1. 개요



1. 상세 설계내용

- backtracking기법

체스판에서 퀸의 위치를 표현하기 위해 visited배열을 만들어 (i, j)위치를 visited[i] = j로 표현할 수 있도록 하였다.

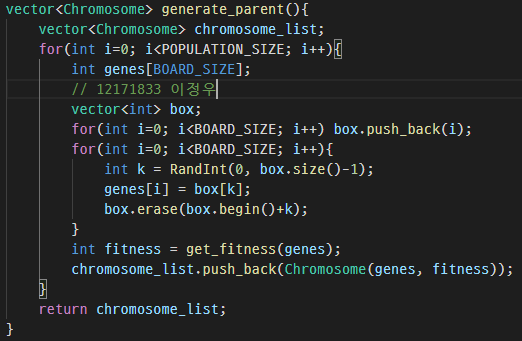
level과 p를 가지고 있는 pair로 order라는 배열을 만들어주었다.

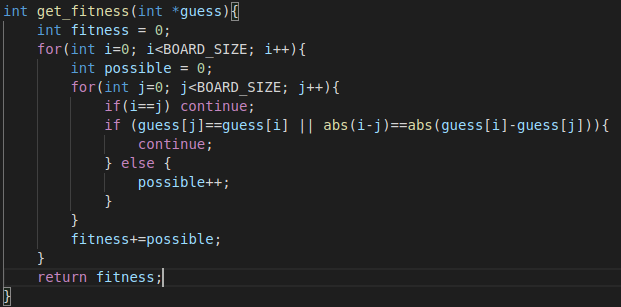
처음에는 level 0번째에 있는 모든 위치는 퀸을 놓을 수 있기 때문에 order에 넣어주었다.

두번째 행인 level 1번째의 위치부터는 위에 놓은 퀸에 의해 퀸을 놓을 수 없는 위치가 생기기 때문에 퀸을 놓을 수 있는 위치만 order에 넣어주었다.

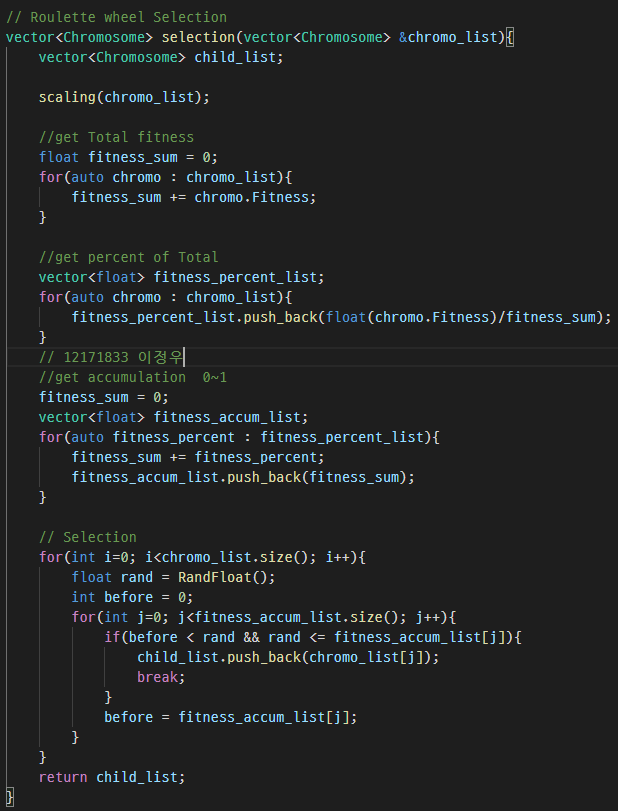
만약 level에 퀸을 놓을 수 있는 경우가 없으면 order에서 위쪽의 level의 다른 위치부터 다시 내려오기 시작한다.

- Genetic Algorithm

  
i행 genes[i]열에 퀸을 놓을 수 있는 genes배열에 랜덤 순열을 넣어주어 처음 유전자를 생성한다.

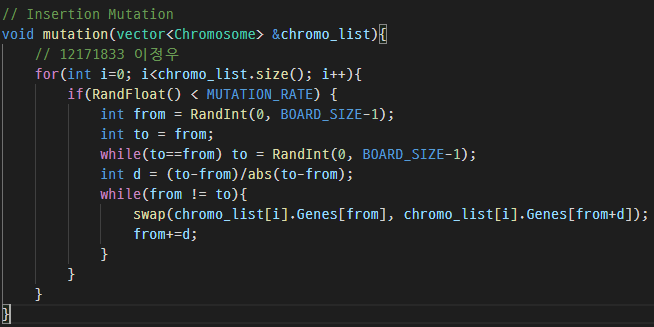
  
자신을 제외한 줄에서 퀸과 위치를 비교해 놓을 수 있다면 fitness에 +1해주었다.

만약 모든 line에 퀸을 놓았다면 fitness는 n\*(n-1)값을 가지게 된다.



Roulette Wheel selection 방식을 이용하였다.

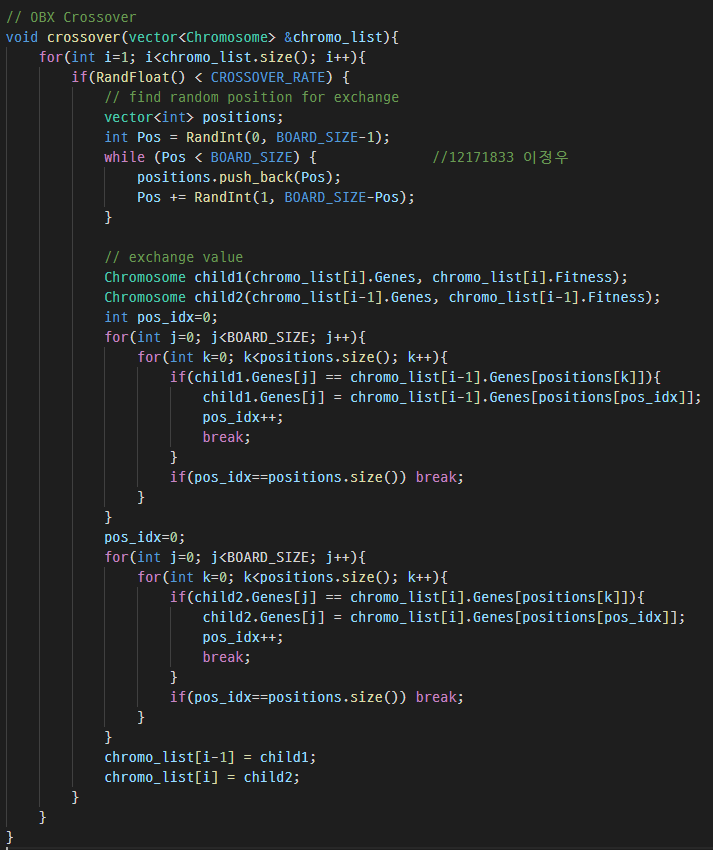
0부터 1까지 accumulation list를 만들어 rand값에 해당하는 위치에 있는 유전자를 고른다.



Insertion Muation을 이용하였다.

0~1사이의 랜덤 값이 MUTATION\_RATE보다 작다면 임의의 두 위치를 정해,

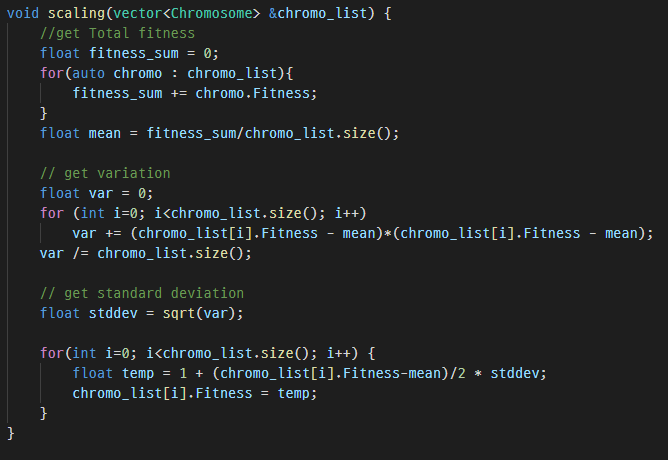
from위치의 값을 to위치로 옮기도록 하였다.



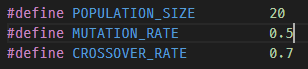
Order Based Crossover를 이용하였다.

0~1사이의 랜덤 값이 CROSSOVER\_RATE보다 작다면 random으로 위치를 정하여준다.

child1은 parent[i]의 값 중에서 parent[i-1]의 random position의 값들을 parent[i-1]의 순서대로 정렬 될 수 있도록 하였고, child2는 반대로 하여 crossover되도록 하였다.

  
Sigma Scaling을 하였다.

평균과 표준편차를 이용해 값의 범위를 -1~1로 바꿔주고, 1을 더한값을 fitness에 없데이트 해주어 0~2값을 가지도록 하였다.

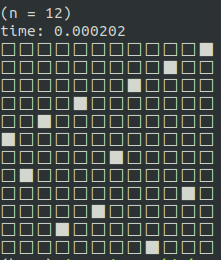
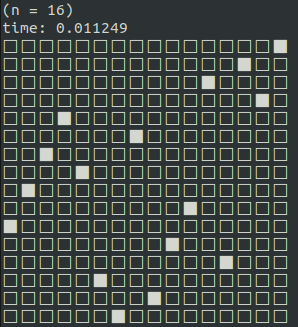
  
POPULATION\_SIZE를 100으로 설정하였다.

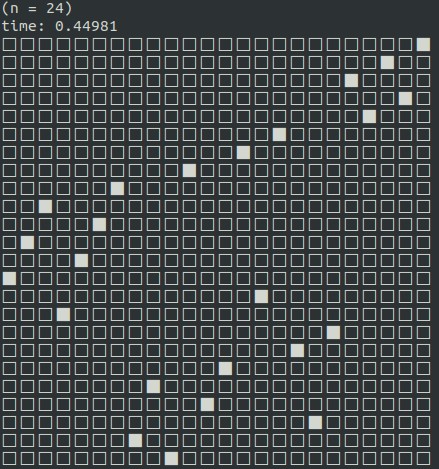
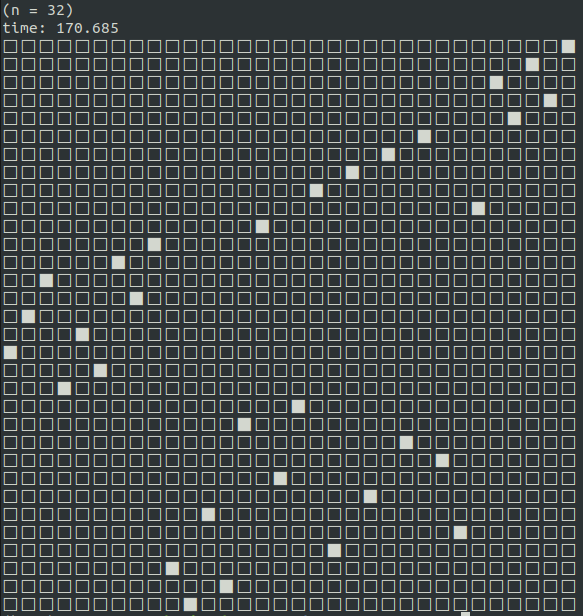
MUTATION\_RATE는 0.5로 설정하였다. fitness가 높아도 체스판을 가득 채우지 못하는 경우가 있기 때문에 MUTATION\_RATE을 높여주었다.

CROSSOVER\_RATE는 0.7로 설정하였다.

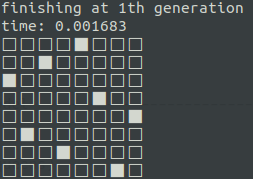
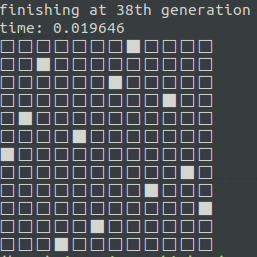
1. 실행 화면

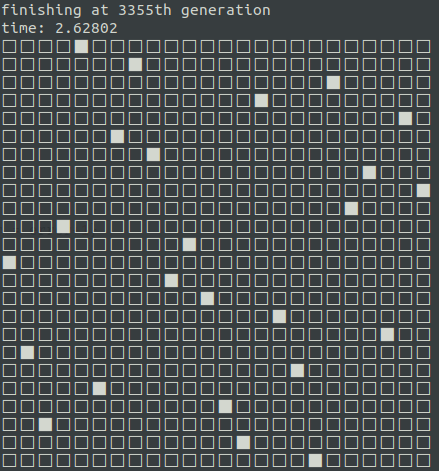
- Backtracking

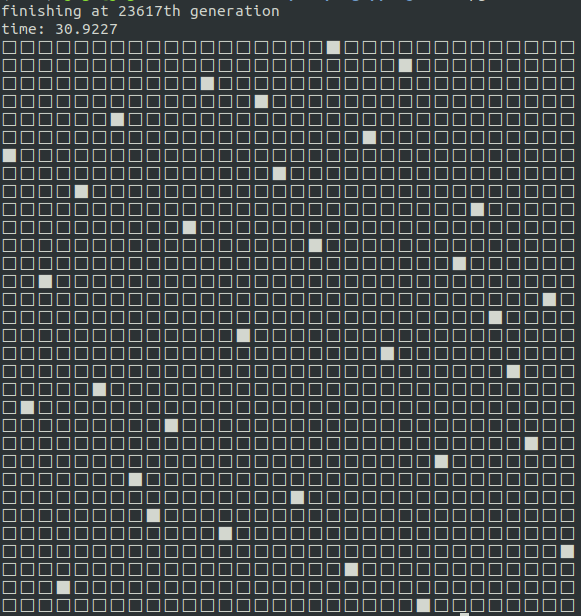
  

- Genetic Algorithm



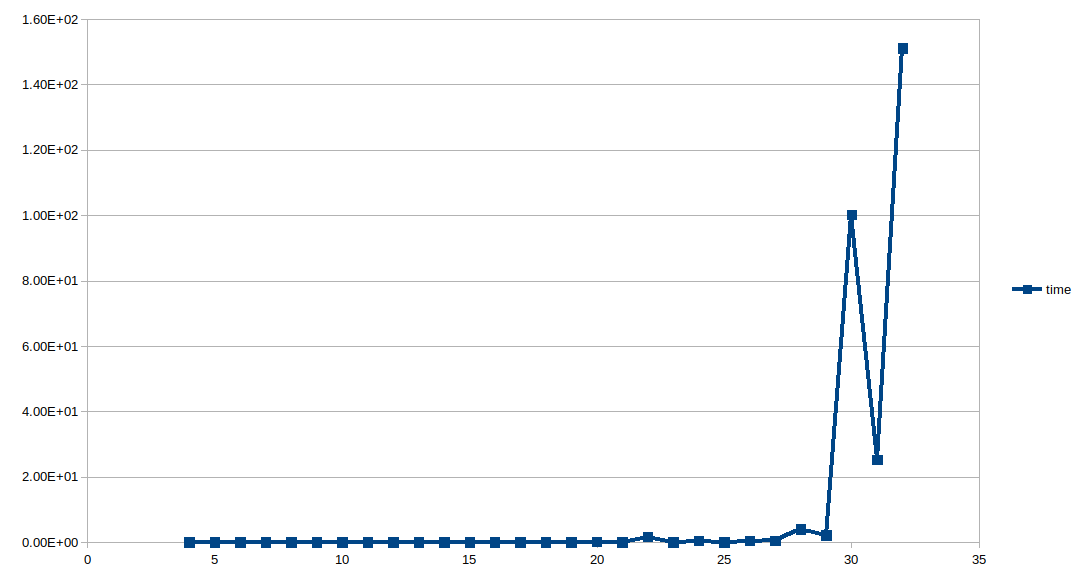


분석 및 결론

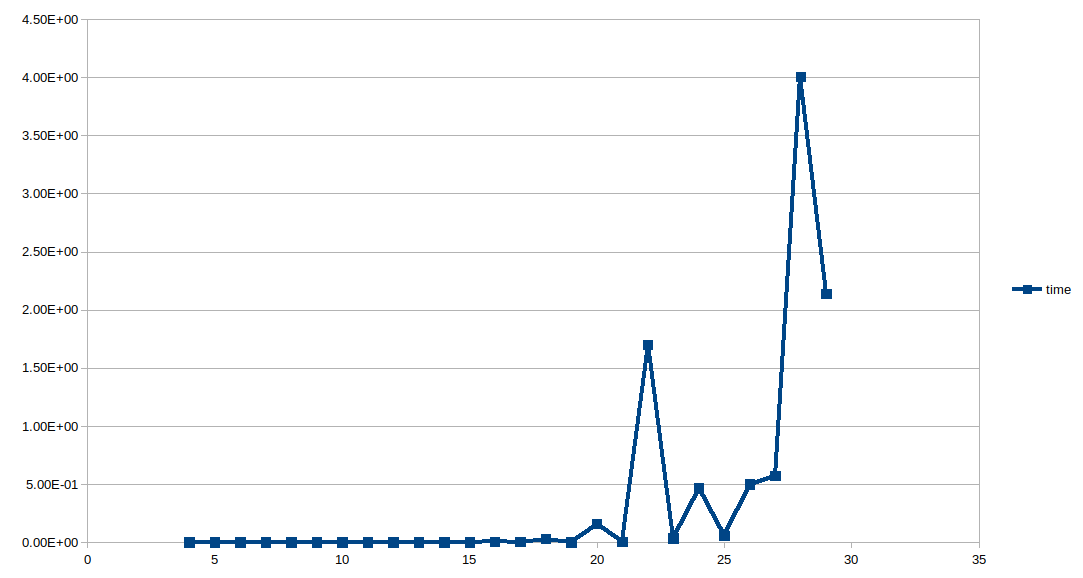
- Backtracking

n값에 따른 시간변화를 보기 위해 그래프를 그려보았다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| time | 2x10-5 | 5x10-6 | 1.5x10-5 | 7x10-6 | 5.8x10-5 | 2.5x10-5 | 6.2x10-5 | 3.5x10-5 | 0.000181 |
| n | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| time | 8.4x10-5 | 0.001551 | 0.001227 | 0.008208 | 0.003924 | 0.031004 | 0.001905 | 0.159225 | 0.007653 |
| n | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| time | 1.70234 | 0.035182 | 0.467612 | 0.059104 | 0.500981 | 0.572547 | 4.00631 | 2.14095 | 100.215 |
| n | 31 | 32 |  |  |  |  |  |  |  |
| time | 25.3659 | 151.201 |  |  |  |  |  |  |  |



(n : 4~32)



(n : 4~29)

- Genetic Algorithm

N=8

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복 | 1 | 2 | 3 | 4 | 5 |
| time | 0.001683 | 0.001667 | 0.001877 | 0.0039 | 0.003178 |

Average time : 0.002461s

N=12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복 | 1 | 2 | 3 | 4 | 5 |
| time | 0.019646 | 0.051337 | 0.012966 | 0.010204 | 0.009842 |

Average time : 0.020799s

N=16

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복 | 1 | 2 | 3 | 4 | 5 |
| time | 0.015823 | 0.859028 | 2.72809 | 0.029647 | 0.007996 |

Average time : 0.728117s

N=24

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복 | 1 | 2 | 3 | 4 | 5 |
| time | 2.62802 | 0.061587 | 0.466185 | 3.2763 | 0.94716 |

Average time : 1.47585s

N=32

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복 | 1 | 2 | 3 | 4 | 5 |
| time | 30.927 | 0.381801 | 20.3381 | 12.4692 | 79.1754 |

Average time : 28.6583s

Backtracking을 이용하였을 때 n이 30보다 작을 때는 실행시간이 5초이하로 나왔지만 30보다 커질 때 시간이 급격하게 오래 걸리는 것을 볼 수 있었다.

또한 해를 찾으면 바로 return해주기 때문에 n이커지면 실행시간이 대체적으로는 증가하지만 무조건 오래 걸리는 것은 아니란 것을 볼 수 있다.

Genetic 알고리즘을 이용했을 때 랜덤 값에 의해 시간이 빨라지거나 느려지기 때문에 시간에 안정적이지는 않지만 N이 커지면 평균적으로 봤을 때 실행시간이 길어지는 것을 볼 수 있다.

또한 N이 30보다 작을 때 까지는 Backtracking기법이 더 빠른 시간안에 문제를 해결했지만 그 이후로는 Genetic알고리즘을 사용한 것이 훨씬 더 빠른 것을 볼 수 있다.

처음 genetic알고리즘을 구현하였을 때 population size를 20으로하여 실행해보았는데 N이 32일 경우에 10번중에 2번정도만 5분안에 수행이 되어 population size를 100으로 늘려서 실행을 하였다.