

## Algorithm #3 Homework

2019-18499 김준혁

### 실행 결과 및 실행 시간

#### 1. 공유된 testcase 10개에서의 시간 비교

##### (1) iterative backtracking

```
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : n = 4, microseconds : 6us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./2.in ./2.out
Input size : n = 4, microseconds : 8us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./3.in ./3.out
Input size : n = 5, microseconds : 13us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./4.in ./4.out
Input size : n = 6, microseconds : 13us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./5.in ./5.out
Input size : n = 7, microseconds : 56us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./6.in ./6.out
Input size : n = 7, microseconds : 104us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./7.in ./7.out
Input size : n = 8, microseconds : 165us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./8.in ./8.out
Input size : n = 8, microseconds : 396us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./9.in ./9.out
Input size : n = 8, microseconds : 1346us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 1 ./10.in ./10.out
Input size : n = 8, microseconds : 861us
```

##### (2) recursive backtracking

```
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : n = 4, microseconds : 6us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./2.in ./2.out
Input size : n = 4, microseconds : 8us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./3.in ./3.out
Input size : n = 5, microseconds : 13us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./4.in ./4.out
Input size : n = 6, microseconds : 16us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./5.in ./5.out
Input size : n = 7, microseconds : 37us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./6.in ./6.out
Input size : n = 7, microseconds : 96us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./7.in ./7.out
Input size : n = 8, microseconds : 185us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./8.in ./8.out
Input size : n = 8, microseconds : 377us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./9.in ./9.out
Input size : n = 8, microseconds : 1265us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw3_2019-18499_김준혁$ ./main 2 ./10.in ./10.out
Input size : n = 8, microseconds : 782us
```

## 2. 직접 만든 testcase 3개에서의 시간 비교 (n=11, 12, 13)

### (1) iterative backtracking

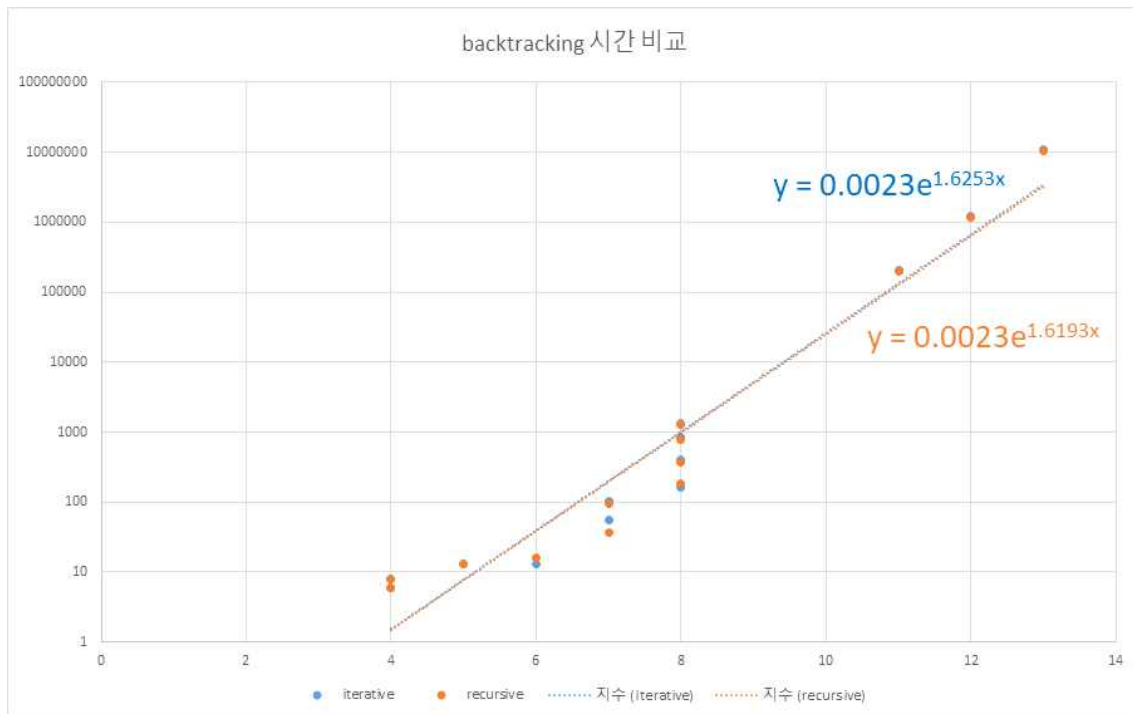
```
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : n = 11, microseconds : 210795us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 1 ./2.in ./2.out
Input size : n = 12, microseconds : 1236657us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 1 ./3.in ./3.out
Input size : n = 13, microseconds : 10768930us
```

### (2) recursive backtracking

```
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : n = 11, microseconds : 199471us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 2 ./2.in ./2.out
Input size : n = 12, microseconds : 1192444us
● dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW3_2019-18499_김준혁$ ./main 2 ./3.in ./3.out
Input size : n = 13, microseconds : 10292703us
```

## 각 알고리즘 실행 시간 비교

기본적으로 각 area(hole에 의해 나뉘어진 row)에서 해당 area에 queen을 놓지 않고 패스를 먼저하며, 이후 column을 키우면서 queen을 insert하는 방식으로 두 알고리즘을 구성했기 때문에, 말 그대로 iterative냐 recursive냐만 다른 알고리즘 구성을 가졌다. 그리하여 두 알고리즘 사이에 실행 시간 차이는 거의 없었다. 다만 recursive에서 iterative에 비해 state 조정에 관련된 코드가 거의 필요가 없어 function call에 걸리는 시간에 비해 오히려 시간이 절약되어 recursive가 최대 10% 정도 실행 시간이 짧은 형태로 나타났다.



가로축은  $n$ , 세로축은 실행 시간(us, log 간격)이다.  $n$ 의 크기에 따라 check하는 case가 크게 증가하며, 기본적으로 NP 문제이기 때문에 추세를 지수 함수로 표현해보았다. 맞는 표현인지는 잘 모르겠다. 이를 통해서도, 두 알고리즘의 시간복잡도는 비슷하지만, recursive가 조금 빠르다는 것을 확인할 수 있다.  $n = 12$ 일 때까지 10초 이내에 실행되는 것을 확인할 수 있으며, 안타깝게도  $n = 13$ 일 때 10초를 약간 넘어서 challenge의 경우 시간 초과가 되는 것을 볼 수 있다.

## Environment & How to run

```

dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ g++ --version
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

개인 구동 환경은 vscode에 WSL을 연결한 Ubuntu 18.04를 이용, g++의 버전은 캡처에 나와 있듯, 공지와 같은 컴파일러를 이용하여, 같은 명령어를 통해 컴파일, 실행(파일 위치에 맞게)하였다.