

Algorithm #1 Homework

2019-18499 김준혁

실행 결과 및 실행 시간

(1) size : 1

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 1, microseconds : 25us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 1, microseconds : 25us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(2) size : 10

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 10, microseconds : 29us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 10, microseconds : 25us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(3) size : 100

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 100, microseconds : 28us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 100, microseconds : 29us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(4) size : 1000

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 1000, microseconds : 37us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 1000, microseconds : 56us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(5) size : 10000

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 10000, microseconds : 114us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 10000, microseconds : 266us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/Hw1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(6) size : 100000

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 100000, microseconds : 871us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 100000, microseconds : 2498us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(7) size : 1000000

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 1000000, microseconds : 8492us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 1000000, microseconds : 26068us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(8) size : 10000000

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 10000000, microseconds : 98814us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 10000000, microseconds : 251366us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

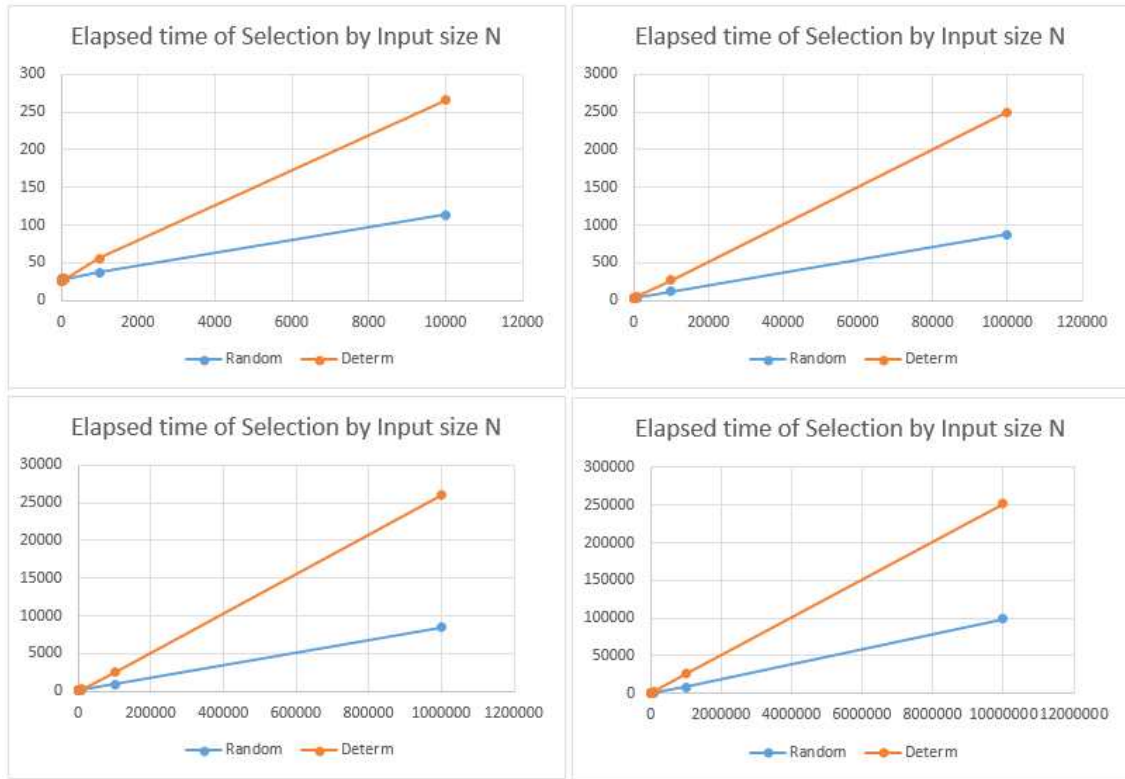
(9) size : 5521212 (randomly generated)

```
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 1 ./1.in ./1.out
Input size : 5521212, microseconds : 41562us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./main 2 ./1.in ./1.out
Input size : 5521212, microseconds : 138563us
dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁$ ./checker ./1.in ./1.out
1
```

(사진에 Randomized Selection의 checker 결과가 미포함되어있지만, 직접 out 파일을 체크하여 올바른 답을 내었음을 확인하였다.)

각 알고리즘 시간복잡도 계수 구하기

$1 \leq n \leq 100$ 일 때의 case를 통해 $O(n)$ 의 시간복잡도를 갖는 두 알고리즘의 그것을 $a_i n + b_i$ 로 표현할 때, $b_i = 25(\mu s)$ 로 생각할 수 있으며, size를 10배씩 늘려가며 확인한 결과를 통해서 직관적으로 $a_1 \simeq 0.01(\mu s)$, $a_2 \simeq 0.025(\mu s)$ 정도로 유추할 수 있지만, 그래프를 통해 조금 더 정확히 파악해보았다.



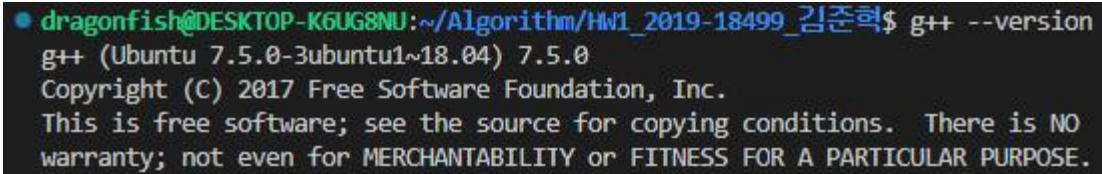
측적을 약간씩 달리하여 확인하여, $a_1 \simeq 0.01(\mu s)$, $a_2 \simeq 0.025(\mu s)$ 정도로 결과를 낼 수 있다고 본다. 그리고 size를 무작위로 정하여 이 식에 대입했을 때(상단 9번 결과), 예상 실행 시간은 $T_{random,pred} = 55237\mu s$, $T_{determ,pred} = 138055\mu s$, 실제 실행 시간은 $T_{random,real} = 41562\mu s$, $T_{determ,real} = 138563\mu s$ 으로, 각각 오차 24.757%, 0.368% 정도로 나타났다. Randomized Select의 특성상 오차의 편차가 있다는 것을 감안하며, Deterministic Select의 오차를 보았을 때, 충분한 정확도를 가질 수 있는 값이라 생각한다.

그러므로 Deterministic Select의 Worst Case가 $O(n)$ 이라 하더라도, Randomized Select와 n 의 계수가 2.5배 차이가 나, 일반적인 상황에서 Randomized Select가 훨씬 효율적이라고 판단했다.

checker 프로그램

checker 프로그램의 조건으로, 일반적으로 $O(n)$ 의 시간으로 동작해야 하므로, 알고리즘 수업을 통해 들은 Selection algorithm 중에, pivot을 함수가 parameter로 받은 배열/벡터의 가장 오른쪽 값으로 하는, 가장 일반적인 Selection algorithm을 이용하여 checker.cc를 구성하였다. 이를 통해 X.in의 값을 입력받아 Selection 계산을 한 후에, X.out과의 값을 비교하여 정답인지를 출력하도록 하였다.

Environment & How to run

A terminal window with a black background and green text. The prompt is 'dragonfish@DESKTOP-K6UG8NU:~/Algorithm/HW1_2019-18499_김준혁\$'. The command 'g++ --version' has been executed, resulting in the output: 'g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0', 'Copyright (C) 2017 Free Software Foundation, Inc.', and 'This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.'

```
● 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.