Algorithms
Homework 1 Report

2013-11425 이창영

1. 개요

- randomized selection algorithm은 평균적으로 O(n)의 시간이 소요되지만, 최악의 경우에는 O(n^2)의 시간이 소요된다.
- deterministic selection algorithm은 randomized와 비슷하지만 pivot을 잡을 때 medain을 구하는 알고리즘을 사용하면 최악의 경우에도 partition이 3:7로 되기 때문에 O(n)의 소요시간을 보장한다.
- 두 algorithm을 직접 구현해보고 여러 크기의 인풋에 따른 실제 수행시간을 측정하여 asymptotic time complexity에 숨어있는 상수를 계산해본다.
- 두 algorithm이 올바르게 작동하는지 확인하는 checker 프로그램을 작성한다.

2. 사용 언어

python을 사용하여 구현하였다.

3. 컴파일, 실행, 실행 결과, 구현 방법

1) rdselect program

- 컴파일, 실행 : python rdselect.py i input_filename
  (i번째로 작은 원소를 input_filename 파일에서 찾는다)
- 실행 결과 : 아래와 같은 형식으로 출력한다.
  [Randomized select result]
  3 smallest element : 16
  Program running time : 0.03s
  [Deterministic select result]
  3 smallest element : 16
  Program running time : 0.02s
- 구현 방법 : 수업시간에 배운 내용을 따라 구현

2) checker program

- 컴파일, 실행 : python checker.py i input_filename n
  (input_filename 파일에서 i 번째로 작은 원소가 n이 맞는지 확인한다.)
- 실행 결과 : 맞으면 True를 출력하고 틀리면 False를 출력한다.
- 구현 방법 :

인풋 전체를 읽어서 n보다 작은 수의 개수(x)와, 같은 수의 개수(y)를 따로 카운트한다.
x < i <= (x + y) 이면 True를 출력하고 그렇지 않으면 False를 출력한다.
이것은 전체 인풋을 한번 읽으면 되므로 O(n)의 수행시간을 가진다.

4. 결과 분석

50000개, 100000개, 500000개, 1000000개, 5000000개, 10000000개 의 인풋에 대한 수행시간을 분석하였다. randomized selection 알고리즘은 같은 인풋에 대해서도 실행할 때마다 결과가 다르기 때문에 각각 10번씩 실행한 후 평균을 구하였다.

1) 50000개

	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	0.02	0.03	0.01	0.02	0.04	0.02	0.05	0.01	0.02	0.01	0.023
deterministic(s)	0.15	0.14	0.16	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
simple rotation											0.002

deterministic / randomized = 6.52

2) 100000개

	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	0.05	0.05	0.03	0.06	0.06	0.05	0.02	0.04	0.08	0.03	0.047
deterministic(s)	0.3	0.31	0.31	0.31	0.32	0.31	0.32	0.33	0.31	0.35	0.317
simple rotation											0.004

deterministic / randomized = 6.74

2) 500000개

	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	0.11	0.15	0.07	0.25	0.27	0.26	0.1	0.12	0.11	0.22	0.166
deterministic(s)	1.66	1.67	1.63	1.61	1.7	1.61	1.61	1.61	1.63	1.63	1.636
simple rotation											0.024

deterministic / randomized = 9.86

2) 1000000개

	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	0.17	0.58	1.09	0.74	0.23	0.25	0.56	0.1	0.41	0.4	0.453
deterministic(s)	3.35	3.44	3.4	3.34	3.38	3.4	3.36	3.39	3.37	3.46	3.389
simple rotation											0.046

deterministic / randomized = 7.48

2) 5000000개

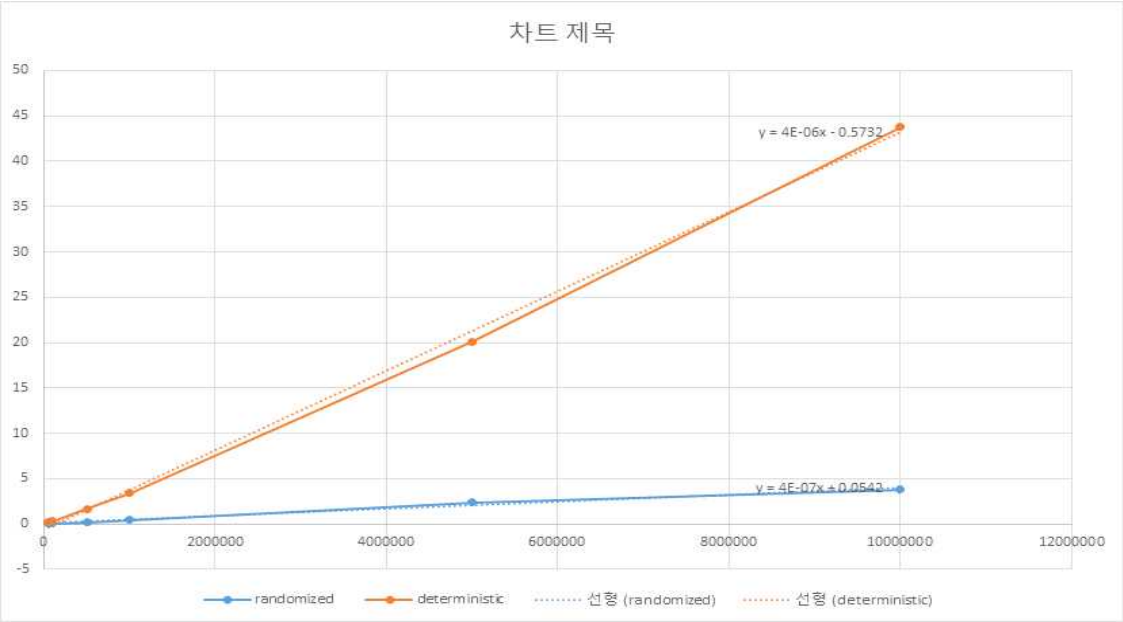
	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	3.2	1.99	1.56	1.31	2.4	5.95	1.61	1.83	2.06	1.63	2.354
deterministic(s)	20.27	20.2	19.85	19.89	20.53	19.81	20.11	20.04	20.05	20.21	20.096
simple rotation											0.287

deterministic / randomized = 8.54

2) 10000000개

	1	2	3	4	5	6	7	8	9	10	평균
randomized(s)	5.32	5.46	3.41	5.05	1.37	7.52	3.52	2.18	1.7	2.54	3.807
deterministic(s)	44.08	43.56	43.71	43.82	44.02	43.68	43.93	43.76	43.88	43.57	43.801
simple rotation											0.476

11.51



그래프를 그려보니 두 selection algorithm 모두 linear한 소요시간을 가진다는 것을 알 수 있다.O(n)

또한 randomized와 deterministic의 상수 비율은 대략 1:8.44 정도 된다.

또한, 각 인풋의 크기에 대해 단순히 반복문으로 한번 순환하였을 때 걸리는 시간을 측정하여 비교하면 n에 대해 어느 정도의 상수 비율을 가지는지 알 수 있다.

	50000개	100000개	500000개	1000000개	5000000개	10000000개	평균
randomized / simple rotate	11.5	11.75	6.92	9.85	8.2	8	9.37
deterministic / simple rotate	75	79.25	68.17	73.68	70.02	76.04	73.69

따라서 구현한 randomized selection algorithm, deterministic selection algorithm은 단순 순환에 비해 각각 9.37, 73.69 정도의 상수 비율을 가지고 있음을 알 수 있다.

5. 결론

- 이론적으로 randomized selection algorithm은 평균 O(n)의 수행시간을 보장하지만, 최악의 경우에는 O(n^2)의 수행시간을 가진다. 하지만 deterministic selection algorithm은 최악의 경우에도 O(n)의 수행시간을 보장한다.

- 하지만 실제로 구현하여 실행해 보았을 때 randomized selection algorithm이 훨씬 더 빠르게 작동하는 것을 알 수 있다.
- 그 이유는 deterministic selection algorithm이 매우 큰 overhead를 가지기 때문이고, randomized selection algorithm이 계속해서 worst한 방향으로 진행될 확률은 매우 낮기 때문이다.
- 이를 통해 많은 경우에서 random을 이용한 algorithm이 굉장히 좋은 알고리즘이 될 수 있음을 알 수 있다.