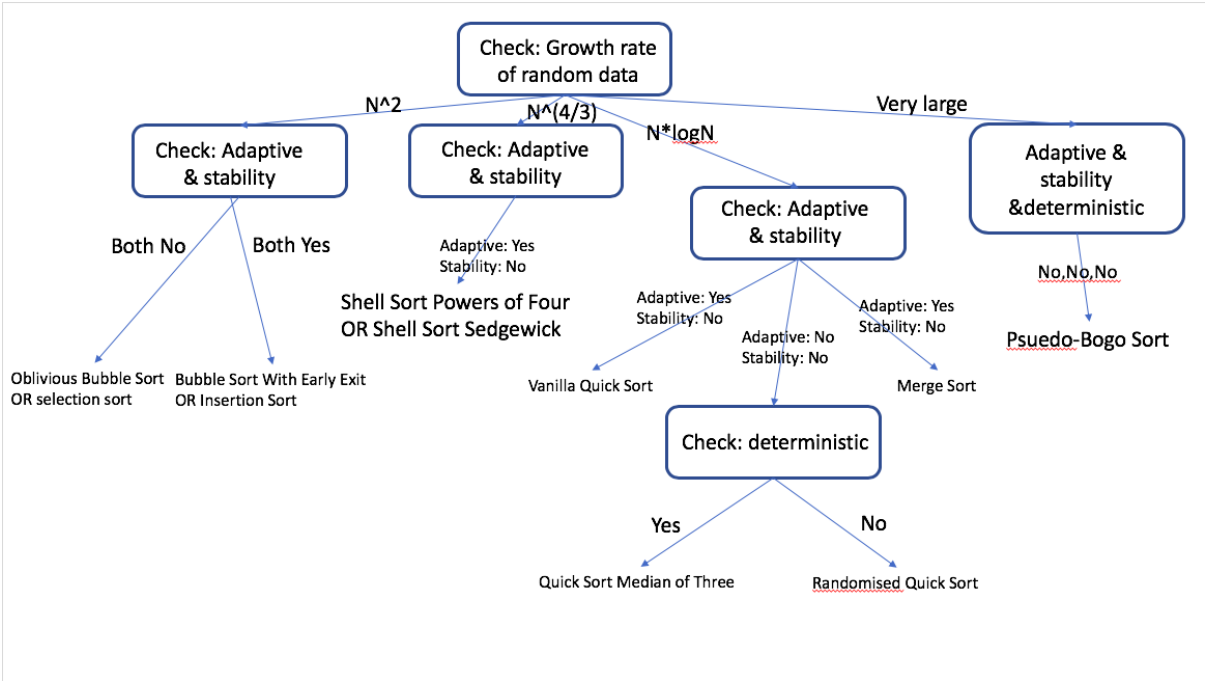
Experimental Design

Performance Analysis

I made this diagram according to my understanding of these different type of sort methods and my research.

	output stability	adaptive	Best	execution time (random		Worst	deterministic
Psuedo-Bogo Sort	No	No	short if lucky	really long		really long	No
Oblivious Bubble Sort	No(given)	No	n^2	n^2		n^2	Yes
selection sort	No	No	n^2	n^2		n^2	Yes
Bubble Sort With Early Exit	Yes(given)	Yes(for ordered)	n(for ordered)	n^2		n^2	Yes
Insertion Sort	Yes	Yes(for ordered)	n(for ordered)	n^2		n^2	Yes
Shell Sort Powers of ~~Two~~ Four	No	Yes For ordered	n log n(for special case)	larger than n^(4/3)		larger than n^(4/3)	Yes
Shell Sort ~~Sedgewick~~ (Sedgewick-like)	No	Yes	n log n(for special case)	n^(4/3)		n^(4/3)	Yes
Vanilla Quick Sort	No	Yes(ordered and reverse takes more time)	n log n	n log n		n^2(ordered and reverse)	Yes
Merge Sort	Yes	No	n log n	n log n		n log n	Yes
Quick Sort Median of Three	No	No	n log n	n log n		n log n	Yes
Randomised Quick Sort	No	No	n log n	n log n		n log n	No

Then I made this step by step process diagram shown how to distinguish different sort programs.



Growth rate: Use n of identity random number from 1-n, and n of repeated number to get average executed time for each n.

Adaptive: Get average executed time for sorted and sorted reverse data.

To distinguish Oblivious Bubble Sort and selection sort, use data

9a 10 9b 1 2 3

Oblivious Bubble Sort would get: 1 2 3 9b 9a 10

Selection sort would get: 1 2 3 9a 9b 10

Experimental Results

Performance Experiments

B	nlogn		
	random	sorted	reverse
100000	0.05	0.05	0.04
200000	0.1	0.1	0.1
300000	0.15	0.13	0.15
500000	0.23	0.22	0.19
700000	0.3	0.3	0.3
900000	0.41	0.36	0.38

A	n^2		
5000	0.04	0.06	0.06
10000	0.19	0.18	0.19
20000	0.58	0.58	0.56
40000	2.3	2.2	2.1
80000	9.3	9.0	8.7
160000	37	36	36
100000	14	13.5	13.1
160000	35.4	35.3	33.2

For Program A, we observed that the execute time growth rate is n^2. Then test on adaptive and stability. Both are No. This narrow program A down to Oblivious Bubble Sort and selection sort.

Then use data set 9a 10 9b 1 2 3 to test program A. Get result 1 2 3 9a 9b 10. From this we know that program A is selective sort.

For Program B, we observed that the execute time growth rate is n*log(n). Then we tested deterministic and get this result:

B					
3 a	1 c	1 a	1 b	1 a	1 c
3 b	1 a	1 c	1 a	1 c	1 a
3 c	1 b	1 b	1 c	1 b	1 b
10 a	2 a	2 a	2 a	2 a	2 a

1 a	3 b	3 c	3 c	3 c	3 b
1 b	3 a	3 a	3 a	3 b	3 a
1 c	3 c	3 b	3 b	3 a	3 c
2 a	10 a	10 a	10 a	10 a	10 a

Which shows program B is not deterministic. So B was been narrowed down to Randomised Quick Sort.

Conclusions

On the basis of our experiments and our analysis above, we believe that

- ProgramA implements the selective sorting algorithm
- ProgramB implements the Randomised Quick Sort sorting algorithm