HW1 - Analysis of Two Efficient Selection Algorithms

2016-17097 Ryu Junyul

1. Abstract

This analysis compares the input size – time relationship of the randomized / deterministic selection. Experiment results show that both algorithms comply with O(n) time complexity, and generally the randomized selection is faster due to the ratio of the constants hidden in the asymptotic time complexities of the two algorithms. However, it is observed that deterministic selection shows a lesser constant ratio when the input distribution is skewed to the right because the efficient pivot selection leads to reduced partitioning overhead.

2. Environment Settings

The root directory of programs is created as a Python virtual environment – Python 3.6.8. The main program – main.py does not require any external libraries, thus no libraries are required for the testing of the program. External libraries – 'pyplot' module of 'matplotlib' to plot the experiment graph, 'numpy' to generate inputs of various distributions – are used in experiment programs maker.py and plotter.py.

** Important: External libraries are **not** used for implementing the selection algorithms or the checker. pyplot is used only for drawing graphs, and numpy is used for generating inputs.

3. How to Run

- Execute command `python3 main.py` from the program root directory
- Enter directory of the input file to the prompt
- Enter whether to run checkers to the prompt

4. Programs

There are three python programs inside the root directory – main.py, maker.py, plotter.py.

- main.py: Main program for running the selection algorithms and the checker. Results of the checker program will be saved to result.txt, and appear on the prompt if chosen to run the checkers. Output files will be saved to the same directory as the input file. For the output files(random.txt, deter.txt), the first line is the value of i-th element, and the second line is the seconds taken for the execution of the algorithm.

- plotter.py: Plots the results of the main.py. For convenience, it plots the results stored in the directories test10**2 ~ test10**8, and saves the graph as an image file.

- maker.py: Creates input files from prompts. Input files can be created with three modes of distribution – random, normal, and skewed. Random distribution is the baseline for the input files. Skewed distribution is introduced for creating imbalanced distributions. Hyper parameters(standard deviation, skewness) are set manually. Figure 1 depicts the example of inputs made by maker.py with input size 10**5.

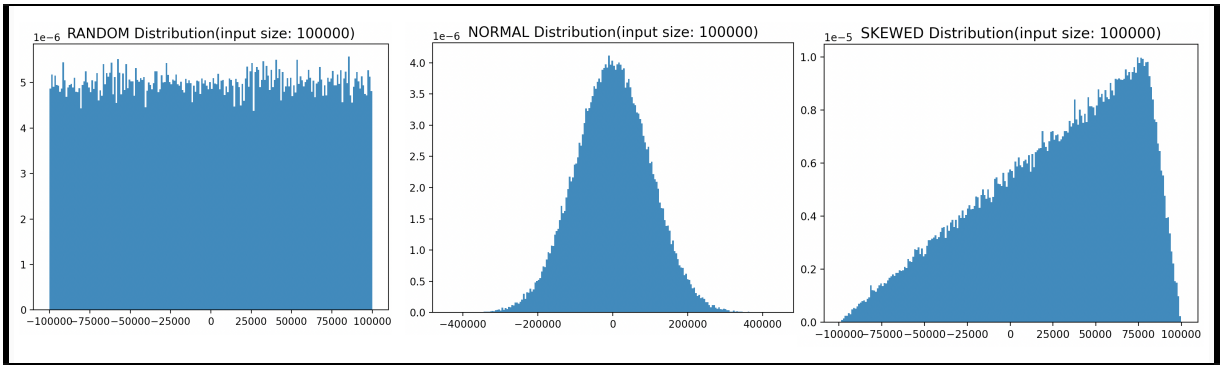

Figure 1. Example inputs made by maker.py

5. Experiment

Experiment is conducted as a comparison of time between the two selection algorithms – randomized select, deterministic select. The input size ranges from 10**2, 10**3, …, 10**8. Also, three distribution types – random, normal, skewed – are created by the maker.py. This separation of distribution types is due to monitor any difference in the performance of the algorithm affected by the distribution. Thus, for each distribution type, the experiment will compare the input size – time relation of the two algorithms within the distribution type.

A. Random Distribution

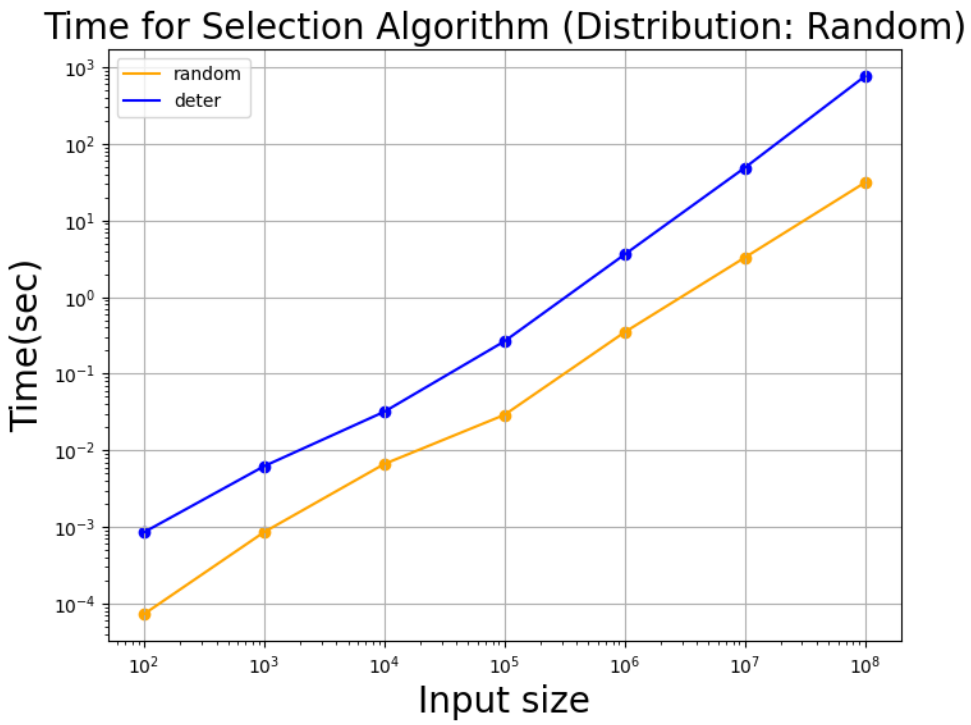

Figure 2. Experiment Result (Input distribution: Random)

B. Normal Distribution

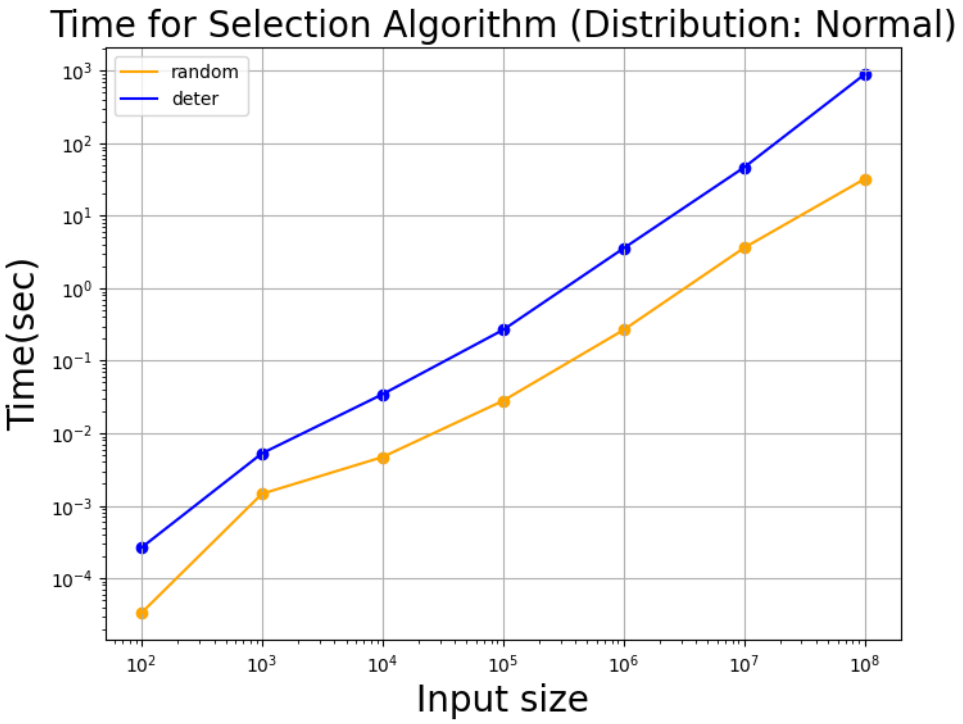

Figure 3. Experiment Result (Input distribution: Normal)

C. Skewed Distribution

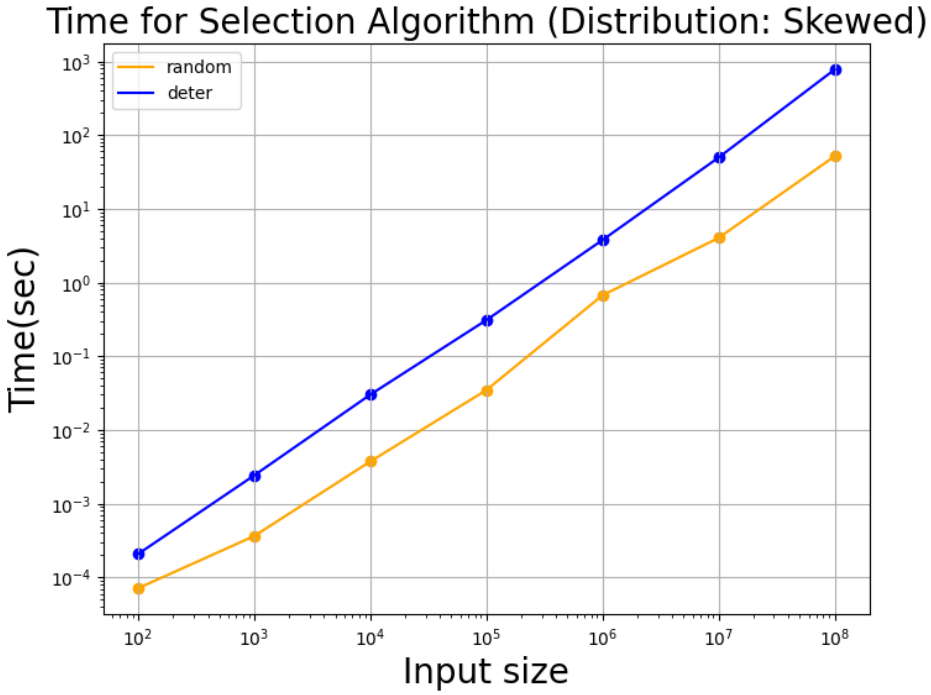

Figure 4. Experiment Result (Input distribution: Skewed)

6. Results

From the experimental results, both algorithms comply with O(n) asymptotic time complexity under all three input distributions. Deterministic selection shows slower performance for all experiment input sizes under all three input distributions. Despite the theoretical fact that the deterministic selection can be performed in $\Omega(n)$, it is slower in practical cases due to the larger constant k hidden as O(kn). The average ratio of the constant between the two algorithms is computed as Figure 5. Also, it is observed that the ratio is significantly lower when the input distribution is skewed right. From this, it can be deduced that the overhead of partitioning is significantly larger when the input distribution is skewed to the right, because the partition algorithm uses the rightmost element as the pivot. However, the deterministic selection algorithm leverages this condition because it chooses the most efficient pivot element which is the mean of the means. Therefore, deterministic selection shows lower constant ratio results when the input distribution is skewed to the right, although it is still much slower than the randomized selection.

Random	Normal	Skewed	Average
11.7848	11.7760	8.4979	10.6862

Figure 5. Constant Ratio Result

7. Example Running

Two examples are provided in /test1 and /test2. Outputs are saved in the same directory.

- /test1 (input size: 10**5)

```
(venv) ryu@yujun-yeol-ui-MacBookPro hw1 % python3 main.py
Directory of input file: ./test1
Run checkers? [Y/N]: Y
[Random Select]  SUCCESS
[Deterministic Select]  SUCCESS
DONE -- Output files saved to  ./test1/
```

[Element] 58952
[Time(sec)] random: 0.042556047439575195 deter: 0.2665741443634033

- /test2 (input size: 10**6)

```
(venv) ryu@yujun-yeol-ui-MacBookPro hw1 % python3 main.py
Directory of input file: ./test2
Run checkers? [Y/N]: Y
[Random Select]  SUCCESS
[Deterministic Select]  SUCCESS
DONE -- Output files saved to  ./test2/
```

[Element] 63346
[Time(sec)] random: 0.3421976566314697 deter: 3.554591655731201