DSA Homework 1 Report

Question 1

Python code is a separate file.

Plot of the time cost as a function of input numbers:

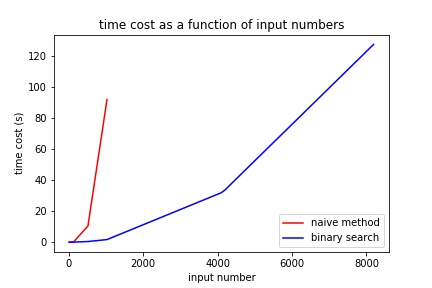


Table of data:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 4192 | 8192 |
| Time of naïve method (s) | 0.016923 | 0.022375 | 0.187438 | 10.458159 | 91.951641 | N/A | N/A | N/A |
| Time of binary method (s) | 0.017953 | 0.024934 | 0.040442 | 0.390084 | 1.654021 | 31.834062 | 33.529971 | 127.38970 |

Discussion and analysis:

Run time for naïve method with input size 4096 and bigger is too long to record.

For the naïve method, since each number in input array will be checked through three loops, big O for this method will be .

For the binary method, which takes binary search instead of the third, as well as the most inner loop check, it costs first to sort the input array and then takes to do the outer two loops and binary search. Since also bounds , the big O for this method is simply .

We can tell from the plot that time cost of the binary method grows slower than the naïve method.

Question 2

Python code is a separate file.

Plot of the time cost as a function of input numbers:

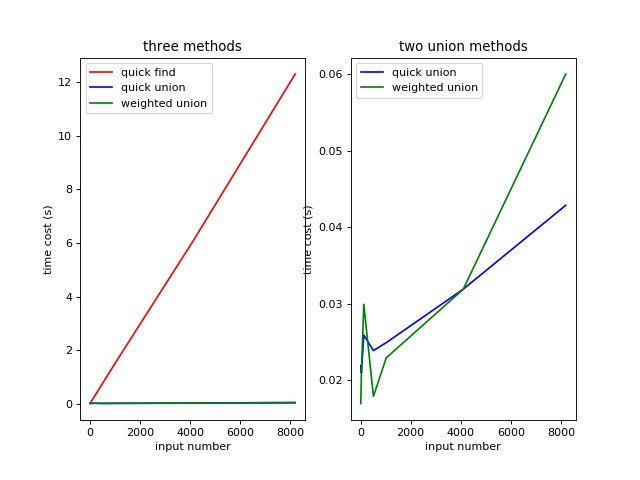


Table of data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 8192 |
| Time of quick find (s) | 0.039919 | 0.063803 | 0.209440 | 0.784708 | 1.551888 | 6.044102 | 12.291668 |
| Time of quick union (s) | 0.021939 | 0.020977 | 0.025932 | 0.023901 | 0.024933 | 0.031916 | 0.042885 |
| Time of weighted union (s) | 0.020944 | 0.017952 | 0.020945 | 0.020943 | 0.022939 | 0.033911 | 0.049897 |

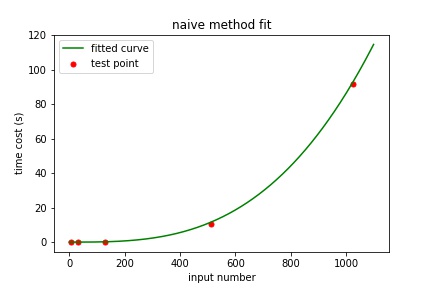
Discussion and analysis:

Run time of quick find has big O of . From the plot we can tell it is almost a straight line with the increasing of input size. From the left plot, we can tell that both union methods grow much slower than the quick find method.

For both union methods, the weighted union method is designed to perform better but as the result doesn’t have significant difference between it and the quick union method. One possible reason may be that the input size is not big enough to have tall tree in quick union method to show the difference.

Question 3

Python code is a separate file.

Plot of fitted curve: 

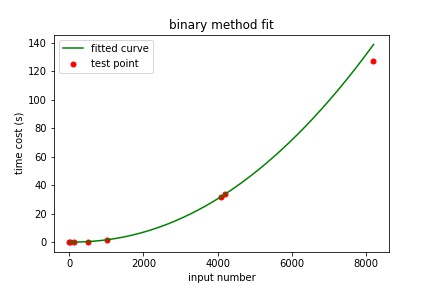
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 4192 | 8192 |
| Time of test (s) | 0.016923 | 0.022375 | 0.187438 | 10.458159 | 91.951641 | N/A | N/A | N/A |
| Time of fit (s) | 0.000044 | 0.002830 | 0.181138 | 11.592802 | 92.742416 |  |  |  |

Discussion:

In fitting result, c = 8.55179267e-08. But for the big O bunding the test result, I choose

c’ = 1.01\*c = 8.63731059e-08. Nc here is 512 when time of fit starts to become bigger than time of test.

Fitted curve of binary method:



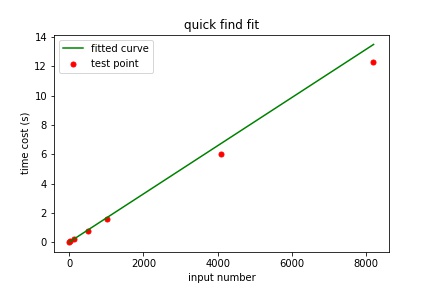
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 4192 | 8192 |
| Time of test (s) | 0.017953 | 0.024934 | 0.040442 | 0.390084 | 1.654021 | 31.834062 | 33.529971 | 127.389705 |
| Time of fit (s) | 0.000041 | 0.000874 | 0.018556 | 0.376818 | 1.671196 | 32.036444 | 33.648810 | 138.788515 |

Discussion:

In fitting result, c = 1.47263334e-07. But for the big O bunding the test result, I choose

c’ = 1.08\*c = 1.47263334e-07. Nc here is 1024 when time of fit starts to become bigger than time of test.

Fitted curve for quick find in Q2:



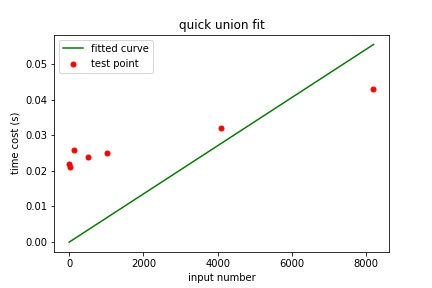
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 8192 |
| Time of quick find (s) | 0.039919 | 0.063803 | 0.209440 | 0.784708 | 1.551888 | 6.044102 | 12.291668 |
| Fitted time (s) | 0.014809 | 0.054300 | 0.212265 | 0.844123 | 1.686601 | 6.741468 | 13.481291 |

Discussion:

In fitting result, c = 0.00149588. But for the big O bunding the test result, I choose

c’ = 1.1\*c = 0.00164546. Nc here is 128 when time of fit starts to become bigger than time of test.

Fitted curve for quick union in Q2:



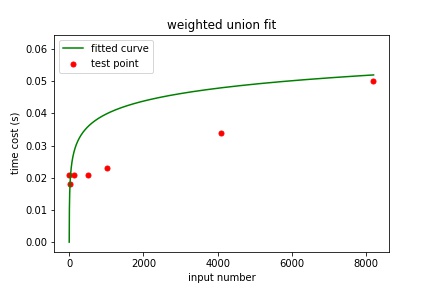
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 8192 |
| Time of quick find (s) | 0.021939 | 0.020977 | 0.025932 | 0.023901 | 0.024933 | 0.031916 | 0.042885 |
| Fitted time (s) | 0.000061 | 0.000223 | 0.000873 | 0.003470 | 0.006933 | 0.027711 | 0.055416 |

Discussion:

In fitting result, c = 6.14893045e-06. But for the big O bunding the test result, I choose

c’ = 1.1\*c = 6.7638235e-06. Nc here is 8192 when time of fit starts to become bigger than time of test.

Fitted curve for weighted union in Q2:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input size | 8 | 32 | 128 | 512 | 1024 | 4096 | 8192 |
| Time of quick find (s) | 0.020944 | 0.017952 | 0.020945 | 0.020943 | 0.022939 | 0.033911 | 0.049897 |
| Fitted time (s) | 0.012660 | 0.020147 | 0.028002 | 0.035956 | 0.039945 | 0.047928 | 0.051921 |

Discussion:

In fitting result, c = 0.00307223. But for the big O bunding the test result, I choose

c’ = 1.1\*c = 0.0039939. Nc here is 32 when time of fit starts to become bigger than time of test.

Question 4:

Python code is a separate file.

In this question, the program should only find the maximum and minimum value of the array. This task can be done within one loop of all input. Since the task will be done within one loop, the running time should grow in order . In reality, 8k input size may not big enough for python to clearly show the growth order, so the plot of runtime as a function od input scale may not be able to the linear pattern.

Plot of runtime as a function of input scale for question4:

