Average Case Analysis

*(Fill in the table cells with execution times)*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | InpType1 | | | InpType2 | | | InpType3 | | | InpType4 | | |
|  | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 |
| Ver1 | 0.00019998550415039061 | 0.001403474807739258 |  | 0.00019960403442382814 | 0.0012059688568115234 |  | 0.0003998279571533203 | 0.0014075279235839845 |  | 0.0 | 0.0012000083923339843 |  |
| Ver2 | 0.00020084381103515624 | 0.0021939754486083986 |  | 0.00020003318786621094 | 0.0017935276031494141 |  | 0.0002045154571533203 | 0.0014057636260986328 |  | 0.0 | 0.00020022392272949218 |  |
| Ver3 | 0.00020122528076171875 | 0.0018000602722167969 |  | 0.00019965171813964843 | 0.0015912532806396484 |  | 0.0001953601837158203 | 0.001592540740966797 |  | 0.00019626617431640626 | 0.0017951011657714843 |  |
| Ver4 | 0.0001987934112548828 | 0.0036026477813720704 |  | 0.00020008087158203126 | 0.001599740982055664 |  | 0.0 | 0.0011999130249023438 |  | 0.0009903907775878906 | 0.00019965171813964843 |  |

Comments:

It is generally agreed upon in the computer science community that versions 2 and 4 are the best approaches.

Version 2, which uses a random pivot, is generally considered the best overall because it has a good average case performance. It has a time complexity of O(n\*log(n)) on average, which makes it a fast and efficient sorting algorithm for large data sets.

Version 4, which uses the "median of three" rule to choose the pivot, is also a good choice because it has a good worst-case performance. It has a time complexity of O(n\*log(n)) in the worst case, which is similar to version 2. However, it is slightly slower than version 2 in the average case due to the extra overhead of selecting the pivot using the "median of three" rule.

Version 1, which always chooses the first element of the list as the pivot, is not a good choice because it has a poor worst-case performance. It has a time complexity of O(n^2) in the worst case, which makes it much slower than the other versions for large data sets.

Version 3, which first randomly permutes the list and then uses the classical deterministic algorithm, is also not a good choice because it has a poor average case performance. It has a time complexity of O(n\*log(n)) on average, which is similar to the other versions. However, it has a higher constant factor due to the overhead of the initial random permutation. This makes it slower than the other versions in practice.

Worst Case Analysis

*(Fill in the table cells with execution times)*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | InpType1 | | | InpType2 | | | InpType3 | | | InpType4 | | |
|  | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 | *n*=100 | *n*=1000 | *n*=10000 |
| Ver1 | 0.0 | 0.015001296997070312 |  | 0.0 | 0.016039371490478516 |  | 0.0 | 0.009966611862182617 |  | 0.0 | 0.002996683120727539 |  |
| Ver2 | 0.0 | 0.0020399093627929688 |  | 0.0 | 0.0020368099212646484 |  | 0.0 | 0.0009725093841552734 |  | 0.0 | 0.0 |  |
| Ver3 | 0.0 | 0.0020003318786621094 |  | 0.0 | 0.0010001659393310547 |  | 0.0 | 0.00103759765625 |  | 0.0 | 0.0020003318786621094 |  |
| Ver4 | 0.0 | 0.001986980438232422 |  | 0.0 | 0.0010004043579101562 |  | 0.0 | 0.0009624958038330078 |  | 0.0009903907775878906 | 0.0 |  |

Comments:

In the worst case, the following time complexities apply to the different versions of quicksort that we listed:

* Ver1: O(n^2)
* Ver2: O(n^2)
* Ver3: O(n^2)
* Ver4: O(n\*log(n))

As we can see, only Ver4 has a time complexity of O(n\*log(n)) in the worst case. This means that it is the most efficient among the four versions in the worst case.

It is important to note that the worst-case time complexity of quicksort is highly dependent on the pivot selection strategy. Choosing the pivot poorly can lead to a worst-case time complexity of O(n^2), which makes the algorithm much slower for large data sets. On the other hand, choosing the pivot wisely can lead to a much better worst-case time complexity of O(n\*log(n)).

Therefore, it is generally a good idea to use a pivot selection strategy that has a good worst-case performance, such as the "median of three" rule (Ver4). This helps ensure that the algorithm remains efficient even in the worst case.