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 | 0.0002092123031616211 | 0.0025971531867980957 | 0.03441375494003296 | 0.0002442598342895508 | 0.003490626811981201 | 0.04797196388244629 | 0.0003387928009033203 | 0.004868507385253906 | 0.08332222700119019 | 0.001133739948272705 | 0.10147100687026978 | 10.37787413597107 |
| Ver2 | 0.00026422739028930664 | 0.0031175613403320312 | 0.0390472412109375 | 0.00025594234466552734 | 0.0029212236404418945 | 0.03896981477737427 | 0.00028514862060546875 | 0.003124535083770752 | 0.04035449028015137 | 0.0012550950050354004 | 0.10412824153900146 | 10.642061054706573 |
| Ver3 | 0.00020313262939453125 | 0.002553999423980713 | 0.032885730266571045 | 0.0002461671829223633 | 0.003491640090942383 | 0.04876101016998291 | 0.00034117698669433594 | 0.004954040050506592 | 0.08659875392913818 | 0.0011336207389831543 | 0.10393881797790527 | 10.515772044658661 |
| Ver4 | 0.0002066493034362793 | 0.0023996829986572266 | 0.031912147998809814 | 0.00021415948867797852 | 0.0023928284645080566 | 0.03214383125305176 | 0.00025266408920288086 | 0.0028904080390930176 | 0.037403643131256104 | 0.001199483871459961 | 0.10546886920928955 | 10.448498725891113 |

Comments:

* *In all input types except InpType4, the best performing version is Version 4, which chooses pivots according to the “median of three” rule.*
* *Since the input array is random (not sorted) in the average case, Version 3’s shuffle before Quicksort does not affect the execution time that much. Both Version 3 and 1 choose the pivot as the first element in a random list, their execution times are very close to each other in all input types.*
* *Moreover, these 2 versions are slower than Version 2 and 4 when the input types are 1, 2, and 3.*
* *Version 2, which chooses pivots randomly, performs better than the versions that choose the pivot as the first element but performs worse than median of three version (Ver4).*
* *All the versions perform poorly when InpType is 4, which means that the input is made of 1’s.*
* *When the duplicate elements in the list increase as we switch from input type 1 to 3, it can be seen that Ver1 and Ver3 are severely affected by this unlike Ver 2 and 4.*

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 | 0.0002932548522949219 | 0.02188897132873535 | 2.2599520683288574 | 0.00019311904907226562 | 0.012312889099121094 | 1.3410592079162598 | 0.00016379356384277344 | 0.0063550472259521484 | 0.562985897064209 | 0.0009021759033203125 | 0.07655811309814453 | 8.500389814376831 |
| Ver2 | 0.00020694732666015625 | 0.0022220611572265625 | 0.027543067932128906 | 0.00019311904907226562 | 0.0022978782653808594 | 0.03263592720031738 | 0.00022602081298828125 | 0.0027320384979248047 | 0.0303647518157959 | 0.000993967056274414 | 0.07688212394714355 | 8.567365884780884 |
| Ver3 | 0.0002880096435546875 | 0.02131032943725586 | 2.2266480922698975 | 0.00019407272338867188 | 0.011982202529907227 | 1.2757487297058105 | 0.0001647472381591797 | 0.006097078323364258 | 0.5564050674438477 | 0.0009031295776367188 | 0.07985997200012207 | 8.734565019607544 |
| Ver4 | 0.00015497207641601562 | 0.0018341541290283203 | 0.022340774536132812 | 0.00015997886657714844 | 0.0021820068359375 | 0.028973817825317383 | 0.0002028942108154297 | 0.0021398067474365234 | 0.02817511558532715 | 0.0009508132934570312 | 0.08533096313476562 | 9.161188125610352 |

Comments:

*(Write your detailed comments about the worst case running times)*

* *Ver1 and Ver3 are affected by the increasing duplicate elements as input type switches from 1 to 3.*
* *Ver4 (median of three) performs the best in input types 1, 2, and 3.*
* *All the algorithms perform poorly when the input type is 4. (Input made of 1’s.)*
* *Ver4 is not affected by the worst execution time and the average execution time because the input style does not change.* *That's why the two execution times are very close to each other*
* *Increasing the n to 10000 causes a serious increase in execution time.*