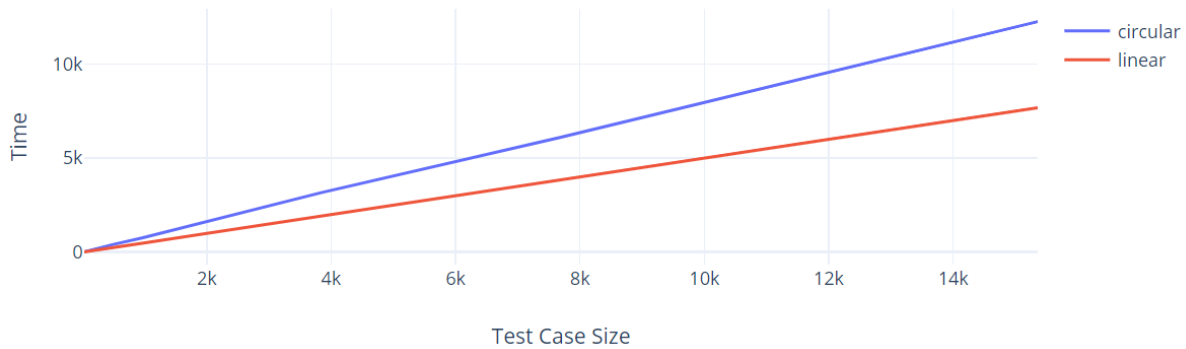


## Lab 4

Function	Big O
<pre> int*linear_queue; int linear_queue_size; int*circular_queue; int circular_queue_size; int opCode; int frontL=0, rearL=0; int frontC=0, rearC=0; </pre>	<p>O(1) [declaration]  <b>O(1)</b></p>
<pre> int isLinearQueueEmpty() {     if(frontL==rearL)     {         return 1;     }     else return 0; } </pre>	<p><b>O(1)</b></p>
<pre> int isLinearQueueFull() {     if(rearL==linear_queue_size)     {         return 1;     }     else return 0; } </pre>	<p><b>O(1)</b></p>
<pre> int linearEnqueue(int x) {     if(rearL==linear_queue_size)     {         return -1;     }     else {         linear_queue[rearL]=x;         rearL=rearL+1;         return 1;} } </pre>	<p>O(1)[comparison, return]    O(1)[index, assignment]  O(1)[arithmetic, assignment]  <b>O(1)</b></p>
<pre> int linearDequeue() {     int x;     if(frontL==rearL)     {         return -1;     }     else {         x=linear_queue[frontL];         frontL=frontL+1;         return 1;} } </pre>	<p>O(1)[Declaration]  O(1)[comparison, return]    O(1)[index, assignment]  O(1)[arithmetic, assignment]  <b>O(1)</b></p>

<pre>int isCircularQueueEmpty() {     if(frontC==rearC)     {         return 1;     }     else return 0; }</pre>	<p>O(1)[comparison, assignment]</p> <p><b>O(1)</b></p>
<pre>int isCircularQueueFull() {     if((frontC==0 &amp;&amp; rearC==circular_queue_size-1)    (rearC == frontC-1))     { return 1; }     else return 0; }</pre>	<p>O(1)[comparison, logical operator, return]</p> <p><b>O(1)</b></p>
<pre>int circularEnqueue(int x) {     if(isCircularQueueFull())     {         return -1;     }     else     { circular_queue[rearC]=x;       rearC=(rearC+1)%circular_queue_size;       return 1;     } }</pre>	<p>O(1)[function, return]</p> <p>O(1)[index, assignment] O(1)[arithmetic, assignment]</p> <p><b>O(1)</b></p>
<pre>int circularDequeue() {     int x;     if(isCircularQueueEmpty())     { return -1;     }     else     {         x=circular_queue[frontC];         if(frontC==rearC)         {             frontC=0;             rearC=0;         }         else{ frontC=(frontC+1)%circular_queue_size;         }         return x;     } }</pre>	<p>O(1)[Declaration] O(1)[function, return]</p> <p>O(1)[index, assignment] O(1)[comparison]</p> <p>O(1)[assignment]</p> <p>O(1)[arithmetic, assignment]</p> <p><b>O(1)</b></p>

Average running time



Ex

```

10 5967 9546 3141 3899 8419 3801 6590 3792 8168 4974 9260 2377 1355 618 1092 9135 1225 5087 5073 2951 5264 6953 914 5861 5594 4769 3554 1280 1898 7358 4898 4217 6994 4392 8
117 1675 4545 1159 1828 2713 6133 1080 5690 7488 8050 2534 2975 9275 7542 5201 8578 9158 8506 5845 5020 4100 6966 4926 5380 5216 2284 6631 9434 5541 1023 3903 7216 5568 596
2 9036 4633 7547 6468 9723 1387 4518 2157 4363 145 6151 5916 5076 5310 4422 921 6682 8523 7887 1608 255 3103 244 6886 8889 5785 4261 2792 9354 9829 4206 4742 814 1753 1211
537 9493 2081 9147 3856 8579 5298 6124 3655 6960 546 928 3642 5421 5167 1602 5677 4622 1847 2563 3512 3984 6825 2656 3338 3006 3215 4433 3821 4968 1996 710 4461 4077 6209 4
669 9008 1508 793 9015 4820 7692 9943 8463 3113 1462 6417 8790 6085 8264 7786 9597 8601 883 8605 8291 241 1820 2724 414 3141 4720 1125 3954 5150 7334 8624 4158 5194 5769 31
74 6367 9813 9469 1182 2927 932 7599 8069 7817 2216 2127 2966 817 3010 1571 9188 9604 3392 1833 18 2885 6553 7495 6839 8055 1182 1815 2214 2728 3937 1740 9095 3758 7561 277
3029 8493 4229 7451 1862 6445 9578 4828 7262 8941 2752 2722 8545 2496 4555 4915 5381 7461 2411 2220 5516 9945 4036 4082 2673 7973 5822 8121 8075 3384 8398 1105 8229 8979 8
556 92 5424 8134 1272 9038 3427 4024 1761 1972 6520 2668 6888 8253 129 5651 474 1998 5596 862 6080 4621 8835 8255 2742 6910 1639 7493 4367 6220 6472 2923 6312 1897 7410 393
7 7287 837 7961 9048 2810 834 1717 6050 9087 8198 8053 5913 196 1 6775 2629 4622 1962 884 7365 5225 8875 4858 9592 1447 7682 2516 4112 5931 6278 8049 3219 7115 6010 2267 62
77 3196 336 8679 2284 8535 6732 8197 5083 6733 4973 4064 7708 6935 1300 5073 2160 6527 6283 8105 7975 3965 6973 2087 6249 3251 136 9468 366 2498 1735 6644 5695 2072 1675 79
79 6959 8408 2528 8394 1493 3853 2459 9201 7141 3759 4274 5653 287 6909 3758 8262 7227 731 6701 3476 3982 6837 2944 701 9335 1031 3697 5030 9455 5372 9361 6414 3780 8242 48
09 1626 2095 7268 827 9236 7379 1454 4890 7666 4715 8648 2280 1942 5732 8981 1770 6066 2170 1066 6767 1506 2098 464 2888 1553 5837 8602 4320 5969 6844 5481 7595 8939 9101 8
423 4528 6480 6229 9418 4147 944 4418 2779 2887 150 8113 1009 6217 283 2076 9336 8141 526 9881 1030 8431 1990 5984 2751 4311 2828 4584 1907 8119 3685 6682 2647 166 9263 206
5 665 207 2836 3444 9446 9338 1557 456 1907 8193 8884 1244 6334 9410 7397 3716 4193 5739 9700 6945 50 2528 7881 8309 7000 1567 1343 9647 1733 606 8065 8750 814 901 2194 661
2 6591 104 7068 8499 8297 5952 9743 983 1714 3492 4700 5988 9231 4400 9205 5633 3281 7086 3943 281 5805 1638 6280 6738 2245 4345 5488 9411 1598 4035 6023 8190 491 9444 6689
5140 1748 2784 6123 3463 2628 823 5723 1859 1576 1280 3844 4857 4718 7787 1490 9724 9426 7770 6462 8023 2116 8303 7434 66 8690 9089 8256 9181 9253 1297 4321 1002 433 6796
817 3061 3972 6540 1272 1900 7820 5117 6757 2538 9256 4599 2262 5804 2369 5077 9409 837 9732 3195 904 8422 3005 5512 7603 8619 3162 8276 9612 9947 1424 429 9361 1748 3321 6
985 3648 7493 2102 6757 6384 7711 1356 4989 9097 3726 6427 8507 4563 6159 1702 1810 4501 4707 3684 8536 9070 6846 3164 0282 6703 941 6064 2506 2688 5737 9492 6338 3231 7946
3095 5967 5657 4452 965 4755 4530 7393 9614 5445 3552 1316 7265 4486 2376 7301 3022 2046 4147 2539 7680 7292 3480 96 9799 6169 5834 0291 8059 5417 7237 8307 1384 9247 2759
1249 354 3641 9742 9968 9086 9647 7636 2703 4133 6364 4 3507 8410 4151 6046 2443 7796 5878 2539 3947 8400 8373 9590 3611 3790 3179 1918 5174 2426 1029 3876 9132 4670 9070
5452 3757 9617 3089 6460 102 9453 6465 3610 4216 6968 6008 6659 4764 1887 5550 5063 6639 3924 4653 250 4066 7833 2169 5593 6611 3198 9469 5744 7869 5791 1196 7978 5409 637
4438 5511 91 7255 5473 659 4224 1482 3670 5340 9721 9220 404 6360 9496 5057 2962 3563 9242 5131 5508 5854 4682 1329 7950 8903 7120 9146 6881 8881 6136 7671 745 6227 4927 62
18 3238 9151 4052 6908 843 3773 2480 1247 6485 1977 2657 9448 1892 8251 931 7400 4105 5613 5081 8407 4516 2201 7554 7749 7435 3690 5421 8180 6269 348 4398 5859 5851 8451 27
67 6694 8576 1599 4294 1414 3576 3303 7214 5468 1554 8145 9220 2012 111 4301 419 979 2855 4325 8729 290 4367 4150 8470 6988 850 9220 2847 6701 7671 1966 9747 2600 3566 4041
4014 7142 3696 1228 8963 5251 5725 8183 3615 5836 8837 4034 3168 8044 8360 1897 8334 9079 2399 3156 6068 9601 2376 8915 6302 6400 7234 2401 9000 800 2795 3014 4294 6491 59
4 9609 8094 6319 7793 1709 2156 2982 5744 5324 1026 456 3573 5712 9535 2324 8868 1955 1925 1244 871 4579 3996 4457 6980 2996 5257 9775 2362 5903 2619 2956 5513 713 9276 965
8 2423 7784 2640 4519 9460 18 4975 9385 5730 862 1709 4598 2818 9986 2194 41 917 2543 4498 7897 5539 6107 4025 4254 8362 6644 7210 3875 7357 2838 9885 9780 6974 8877 651 27
86 8895 1978 2171 977 2841 232 1927 2011 6570 474 8404 7487 3017 9254 5385 4908 1713 5762 9162 75 2406 2725 303 6115 1915 6540 5896 8890 5418 2899 1676 665 4878 200 1643 40
71 432 3570 6082 7003 396 838 4490 3413 92 6227 8322 8157 1989 7484 8232 747 6561 4887 6863 8477 1428 9111 7367 3198 2010 5395 3863 6888 5595 1858 950 2380 1781 3393 9383 2
177 583 225 5591 675 2805 3913 8832 1146 7749 3417 1894 663 4656 5109 9140 6084 572 2859 9282 2582 8254 9498 5823 3850 1356 3134 2582 3137 2880 8317 5315 3463 8542 906 491
7699 1171 5675 8846 5272 5444 7092 5935 101 8553 5075 6185 9125 7934 1820 8059 2541 1318 3882 2743 2674 3369 5325 2164 6249 3642 7479 6064 8536 4737 6555 6236 2260 8583 143
4 7532 4027 4878 9820 4128 3431 1247 6666 2556 5534 8486 615 8075 6156 850 7170 5182 4219 2495 7346 6820 6137 4825 9236 1025 5914 5792 7261 8174 727 5047 5707 4754 9925 187
9 5235 3356 3126 1901 2264 5012 6739 9232 9439 9247 82 6609 4429 693 9104 1776 3825 1593 2953 3061 2619 5220 5205 9880 3394 5932 1280 5453 7039 1205 7332 2274 914 6811 4175
3178 1823 914 2410 7615 151 8044 4224 542 9497 3329 9070 9674 4022 8376 2736 3893 3506 4093 126 6990 226 1406 8796 7205 2611 6128 9539 3525 9201 66 3056 7467 7332 5466 508
2 3845 4311 9306 4707 160 8907 3850 9835 262 8586 8923 4165 2182 3216 4201 5524 0794 2039 4200 7059 4651 6801 2950 4528 2444 3016 7584 9911 6700 9403 1345 545 66 7004 5333
226 5991 5543 6413 2605 4129 5336 3113 2663 4905 3746 8187 4699 5786 8660 8111 6789 2013 1061 7669 4457 430 1606 721 3482 1009 2066 4028 1075 0070 5713 7653 1414 1256 4067
871 1737 3575 3484 752 660 7231 5291 5360 3017 4151 3471 6158 6164 884 3827 6974 7666 5433 7695 1149 6442 9761 1529 3869 5184 7242 7875 2950 4850 1942 3321 2939 4049 6806 3
691 4710 389 8982 6422 3406 9486 6245 5916 2002 3481 9743 8976 1148 5177 3023 8649 7971 2785 178 8193 4321 3772 6068 7271 8622 4362 592 1561 8411 3750 1604 3121 4139 6938 5
895 3897 2776 2140 9813 1131 5622 5909 107 3122 7438 3131 8123 1761 2268 4653 9954 6589 8425 2374 212 3399 6736 7156 4960 5148 7259 2916 4621 7750 9854 6869 1648 8983 9009
7813 114 983 74 6573 4105 7512 6056 2228 5626 8324 6881 5580 1265 1658 4307 7829 5597 1043 4986 17 2543 8597 9285 3517 6347 5492 386 7995 4475 5747 2161 4589 6731 8587 1162
7188 6100 7219 9417 1726 1895 2650 3658 9513 4309 4317 [2019A7PS0020U]linuxbpcd1 Week_04]$

```

Function	Big O
<pre> void load(int array[],int size) {     int index=1;     int length=size;     enqueue(array[1]);     while(size&gt;1)     {         if(2*index &gt; length    2*index+1&gt; length)         {             break;         }         if(array[2*index]!=-1)         {             enqueue(array[2*index]);         }     } } </pre>	<p>O(1)[declaration, assignment]  O(1)[declaration, assignment]  O(1)  O(n-1)  O[1*(n-1)](arithmetic, comparison, or)  O[1*(n-1)](arithmetic, index, comparison)  O(1)</p>

<pre>         }         if(array[(2*index) + 1] != -1)         {             enqueue(array[(2*index)+1]);         }         index++;         size--;     } }</pre>	<p><math>O(1 \cdot (n-1))</math> (arithmetic, index, comparison)</p> <p><math>O(1)</math></p> <p><math>O(1)</math> [increment]</p> <p><math>O(1)</math> [decrement]</p> <p><b><math>O(n)</math></b></p>
--	---