Universidad Nacional de San Agustín de Arequipa

Escuela Profesional de Ciencia de la Computación

Algoritmos y Estructuras de Datos

Laboratorio Nro. 01

Presentado por:
Fiorela Villarroel Ramos

Docente :
Rolando Jesus Cardenas Talavera

1. Competencia del Curso

Comprende la importancia e impacto de los algoritmos estudiados y las nuevas propuestas.

2. Competencia del Laboratorio

- Describir, implementar y analizar algoritmos de ordenamiento.
- Interpretar el costo computacional en algoritmos de estudio.

3. Equipos y Materiales

- Un computador.
- Lenguaje de Programación (c++, python, java, c)

4. Actividad

4.1. Ejercicio 1

Genere un archivo con números aleatorios (mayor a un millón), este representará el vector a ordenar en las pruebas de los algoritmos de ordenamiento, un número puede repetirse más de una vez

```
100
1804289383 846930886 1681692777 1714636915 1957747793 424238335
   719885386 1649760492 596516649 1189641421 1025202362
   1350490027 783368690 1102520059 2044897763 1967513926
   1365180540 1540383426 304089172 1303455736 35005211 521595368
    294702567 1726956429 336465782 861021530 278722862 233665123
    2145174067 468703135 1101513929 1801979802 1315634022
   635723058 1369133069 1125898167 1059961393 2089018456
   628175011 1656478042 1131176229 1653377373 859484421
   1914544919 608413784 756898537 1734575198 1973594324
   149798315 2038664370 1129566413 184803526 412776091
   1424268980 1911759956 749241873 137806862 42999170 982906996
   135497281 511702305 2084420925 1937477084 1827336327
   572660336 1159126505 805750846 1632621729 1100661313
   1433925857 1141616124 84353895 939819582 2001100545
   1998898814 1548233367 610515434 1585990364 1374344043
   760313750 1477171087 356426808 945117276 1889947178
```

```
   1780695788 709393584 491705403 1918502651 752392754
   1474612399 2053999932 1264095060 1411549676 1843993368
   943947739 1984210012 855636226 1749698586 1469348094
   1956297539
500
1036140795 463480570 2040651434 1975960378 317097467 1892066601
   1376710097 927612902 1330573317 603570492 1687926652
   660260756 959997301 485560280 402724286 593209441 1194953865
   894429689 364228444 1947346619 221558440 270744729 1063958031
    1633108117 2114738097 2007905771 1469834481 822890675
   1610120709 791698927 631704567 498777856 1255179497 524872353
    327254586 1572276965 269455306 1703964683 352406219
   1600028624 160051528 2040332871 112805732 1120048829
   378409503 515530019 1713258270 1573363368 1409959708
   2077486715 1373226340 1631518149 200747796 289700723
   1117142618 168002245 150122846 439493451 990892921 1760243555
    1231192379 1622597488 111537764 338888228 2147469841
   438792350 1911165193 269441500 2142757034 116087764
   1869470124 155324914 8936987 1982275856 1275373743 387346491
   350322227 841148365 1960709859 1760281936 771151432
   1186452551 1244316437 971899228 1476153275 213975407
   1139901474 1626276121 653468858 2130794395 1239036029
   1884661237 1605908235 1350573793 76065818 1605894428
   1789366143 1987231011 1875335928 1784639529 2103318776
   1597322404 1939964443 2112255763 1432114613 1067854538
   352118606 1782436840 1909002904 165344818 1395235128
   532670688 1351797369 492067917 1504569917 680466996 706043324
    496987743 159259470 1359512183 480298490 1398295499
   1096689772 2086206725 601385644 1172755590 1544617505
   243268139 1012502954 1272469786 2027907669 968338082
   722308542 1820388464 933110197 6939507 740759355 1285228804
   1789376348 502278611 1450573622 1037127828 1034949299
   654887343 1529195746 392035568 1335354340 87755422 889023311
   1494613810 1447267605 1369321801 745425661 396473730
   1308044878 1346811305 1569229320 705178736 1590079444
   434248626 1977648522 1470503465 1402586708 552473416
   1143408282 188213258 559412924 1884167637 1473442062
   201305624 238962600 776532036 1238433452 1273911899
   1431419379 620145550 1665947468 619290071 707900973 407487131
    2113903881 7684930 1776808933 711845894 404158660 937370163
   2058657199 1973387981 1642548899 1501252996 260152959
   1472713773 824272813 1662739668 2025187190 1967681095
   1850952926 437116466 1704365084 1176911340 638422090
   1943327684 1953443376 1876855542 1069755936 1237379107
   349517445 588219756 1856669179 1057418418 995706887
   1823089412 1065103348 625032172 387451659 1469262009
   1562402336 298625210 1295166342 1057467587 1799878206
```

1555319301 382697713 476667372 1070575321 260401255 296864819
774044599 697517721 2001229904 1950955939 1335939811
1797073940 1756915667 1065311705 719346228 846811127
1414829150 1307565984 555996658 324763920 155789224 231602422
1389867269 780821396 619054081 711645630 195740084 917679292
2006811972 1253207672 570073850 1414647625 1635905385
1046741222 337739299 1896306640 1343606042 1111783898
446340713 1197352298 915256190 1782280524 846942590 524688209
700108581 1566288819 1371499336 2114937732 726371155
1927495994 292218004 882160379 11614769 1682085273 1662981776
630668850 246247255 1858721860 1548348142 105575579
964445884 2118421993 1520223205 452867621 1017679567
1857962504 201690613 213801961 822262754 648031326 1411154259
1737518944 282828202 110613202 114723506 982936784
1676902021 1486222842 950390868 255789528 1266235189
1242608872 1137949908 1277849958 777210498 653448036
1908518808 1023457753 364686248 1309383303 1129033333
1329132133 1280321648 501772890 1781999754 150517567
212251746 1983690368 364319529 1034514500 484238046
1775473788 624549797 767066249 1886086990 739273303
1750003033 1415505363 78012497 552910253 1671294892
1344247686 1795519125 661761152 474613996 425245975
1315209188 235649157 1448703729 1679895436 1545032460
430253414 861543921 677870460 932026304 496060028 828388027
1144278050 332266748 1192707556 31308902 816504794 820697697
655858699 1583571043 559301039 1395132002 1186090428
1974806403 1473144500 1739000681 1498617647 669908538
1387036159 12895151 1144522535 1812282134 1328104339
1380171692 1113502215 860516127 777720504 1543755629
1722060049 1455590964 328298285 70636429 136495343 1472576335
402903177 1329202900 1503885238 1219407971 2416949 12260289
655495367 561717988 1407392292 1841585795 389040743 733053144
1433102829 1887658390 1402961682 672655340 1900553541
400000569 337453826 1081174232 1780172261 1450956042
1941690360 410409117 847228023 1516266761 1866000081
1175526309 1586903190 2002495425 500618996 1989806367
1184214677 2004504234 1061730690 1186631626 2016764524
1717226057 1748349614 1276673168 1411328205 2137390358
2009726312 696947386 1877565100 1265204346 1369602726
1630634994 1665204916 1707056552 564325578 1297893529
1010528946 358532290 1708302647 1857756970 1874799051
1426819080 885799631 1314218593 1281830857 1386418627
1156541312 318561886 1243439214 70788355 1505193512
1112720090 1788014412 1106059479 241909610 1051858969
1095966189 104152274 1748806355 826047641 1369356620
970925433 309198987 887077888 530498338 873524566 37487770
1541027284 1232056856 1745790417 1251300606 959372260

```
   1025125849 2137100237 126107205 159473059 1376035217
   1282648518 478034945 471990783 1353436873 1983228458
   1584710873 993967637 941804289 1826620483 2045826607
   2037770478 1930772757 1647149314 716334471 1152645729
   470591100 1025533459 2039723618 1001089438 1899058025
   2077211388 394633074 983631233 1675518157 1645933681
   1943003493 553160358
1000
1635550270 2069110699 712633417 864101839 1204275569 1190668363
   1336092622 410228794 1026413173 773319847 1404196431
   1968217462 452456682 1302539390 1858504292 235745791
   802205057 427355115 1388391521 1272796157 1452888574
   1280631491 126401947 1204462951 1210359231 521035021 40610537
    738393740 19485054 1983614030 1291554098 1655035325
   1905241081 2004187516 371653516 962033002 1047372231
   1707746139 1372261796 2073785404 333582338 628974580
   1894519218 786039021 1931513970 1605539862 1021784812
   586235379 2032894977 262692685 1859031536 1338299904
   1543324176 1985433483 395279207 606199759 358984857 435889744
    1344593499 378469911 272020127 488663950 2033505236 29777560
    345367818 257675105 991810563 1392740049 1965421244
   216588711 1319041805 151519934 845563291 1066077375 937558955
    629593614 524133589 1959343768 1215828993 409544918 74552805
    927376882 1747844822 1617876982 765326717 2143124030
   76593093 1124311574 431530126 1421186593 1502781486 703550253
    1909850543 1388803074 733327814 107734713 1646478179
   1725138377 1500474762 1464415775 1941727088 672032919
   1615935710 639806732 1738110294 406011017 1269400346
   114760235 217871137 337745691 524305153 292423943 1265122573
   124666328 1910300925 2030449291 120306710 1986894018
   1007277217 551836836 1260596963 362575055 1255387090
   1022963858 1751378130 1988714904 1130698571 1250372661
   1566369633 483689685 567304789 1360613073 1155722604 35756851
    2000419805 746349250 441767868 1122336503 861109485
   659639006 1460082195 1385414639 952062949 577721120
   1510080967 714880226 460686763 1630387677 554290596
   1467963981 34740865 1814887560 1830539036 1290127955
   690367770 1434433518 1131359211 1821066342 537322532
   550245196 157272379 1104627321 1910858270 1312994984
   1140384172 1763794427 2059344234 1582152040 738647283
   772970072 94307398 51245830 10901063 1046370347 628966950
   1520982030 1761250573 1089653714 1003886059 168057522
   410134047 1038626924 1982945082 93189435 181271232 525829204
   1527622954 1312630443 199411898 2064945486 1862875640
   356684278 1022089159 1626250262 1669679262 14989683
   1242561041 1581539848 1597141723 1981208324 207026272
   1691449122 2032454154 217927335 590335821 513937457
```

```
1738909365 204102747 1603591171 595311776 372160269
2013725218 1633938701 207621703 2106914653 1815209933
733450907 1487053959 980356728 932862806 1404515797 695748720
 1289547084 279121308 174515334 811742698 294110991
1417076376 245798898 1891252715 1250801052 452825171
1435218189 1135771559 670752506 2025554010 1649709016
262178224 82173109 1105816539 857490000 454333378 972058109
343945053 661955081 931489114 11671338 1395405989 271059426
992028067 180785147 1675575223 1687776787 1470332231
1954696532 1862292122 134591281 101323875 1131884850
380390179 1992576590 235202254 833215350 1280311131
1370973813 1503967857 1158381494 873199181 1766146081
1240554603 1979015720 476152433 1694887982 803590181
820097487 209359415 1735079296 831768825 1604765404
2006138722 1823796892 1785550551 1534230297 1364090032
1108399134 1341443181 1078898506 1242990415 1442767057
63299708 1623380595 1287859999 298501962 309112297 420687483
1669475776 1813080154 1579068977 395191309 1431742587
672139932 226723382 1907895021 219544266 1030313563 580508860
 428903682 617909211 1412277685 2033669086 476564285
1088590930 1671735990 2010794583 305197314 632651476
1204754116 1384095820 1875641892 500037525 1447395528
1351538839 1787897525 1745897490 1660651136 61101360
1267889618 1326247643 1640170337 1663080928 610506582
164826621 1889804310 370917955 384370888 772634225 951426815
813274570 1390543437 216220853 699460008 1867107722
1304811783 223712350 1730418657 1610009097 856363827
787689126 846621269 584522071 1287726651 146533149 1936060910
 928140528 1892430639 1449228398 989241888 1012836610
627992393 481928577 528433890 1238498976 646755199 270754552
1609416931 1031126087 1043388777 413360099 1844400657
286448566 629580952 396377017 6072641 1934392735 620089368
1736491298 1396918184 1476453195 376696776 96055805
2060975266 1664423428 242588954 1849552528 445080308
2135019593 1151297278 1434322197 1000372555 1779289672
1916250774 1528806445 870305000 415522325 1799560997
332238283 1446648412 695466127 745598382 1143565421 981914693
 1375179334 1539942439 987987334 1162088421 12548159
576994985 411522957 1489001354 953691761 507578762 1402492972
 470631541 750167716 1104561852 915711850 737703662 108375482
 202550399 1738076217 1887665154 2118801173 1119399015
610486506 386839851 771476364 942724790 1833488263 1466942491
 1688323172 829570037 301373537 916018859 222028828
1289360871 2078107280 234576987 1866355856 342146590
1723578341 672563970 849725352 978587665 1143195511
1599893069 2083149517 2058907361 190113083 44041351 113974112
 1928189300 1931706506 85291638 900104667 394709364 472131489
```

1671581032 1337434154 158136104 991039875 878273679
987706141 1292413412 1794292538 1209734969 434290636
1724916170 1444311956 153162844 2067062760 1020406649
825726814 769304465 1998994314 1968922326 221713886
1934660183 1880346039 411826969 1978701535 1994320152
192532621 1762924393 2079611790 1092637289 10150109 404259631
616734673 1347584264 562395735 1607774548 78374295
1550101877 752704313 1872666833 612353198 1186994949
1450099355 2056665155 1340157793 1369678468 929588156
18400960 2138982933 781098823 1987323286 213213171 568275358
1720185677 625040140 399493245 1567022181 817572761 14933990
1499150323 1910210050 25084100 1903409954 379461075
1372668364 318322042 1987235624 1451042659 1868423919
592456289 1176225844 333293469 1779451238 478841551 242474976
972125383 1848520019 1172063133 990526343 1840019304
1953161956 830365981 2053232475 373953666 403068011 530788967
773446912 1970090192 1348361729 788380902 1321756868
1111088131 813465002 1077683174 1490549207 38649718
1396005216 1330301183 1489692377 1116945487 1922757472
518434573 1450238957 1554725062 997276125 1692713933
379366797 698312496 717293418 1369893141 390848153 522971726
52775474 296596980 896925393 455843485 827385948 1670372305
278450030 28264029 311269559 1600206898 1139352160 1124734562
530406424 482417719 1163384280 1926411641 1812718902
505593010 895873480 1587992726 1024027583 198628789 995234140
2021303708 1891342723 1374600938 572132557 461152493
597010431 962980710 984124220 649785905 1259577690 1881049613
1105629391 2086963638 1403938270 1384079421 2115227667
1715207829 836802671 1107096180 692458743 1367209095
1589513899 1855843024 1146137088 1254749154 213952386
2042010569 695258232 1237979969 93155710 1690492373
1111800030 1984498433 917609663 1683932587 298167279
1514620094 499429649 1282291499 16922351 1759007339
1015857464 1122551742 1698487330 272312086 359147515
1666231349 1987519915 1195950186 625843881 532495011
415675634 67874133 240854387 1561812722 1322623287 454806773
1456339643 2017881519 1692786742 1549495354 1560890244
657103124 1386510139 331016259 193552063 1684677418
1845636353 692981712 819485269 1862558705 304505404
1835342733 837626799 2002992734 2107654819 1196774315
1521740435 1947691087 245240853 100669 332702450 660916487
67974802 573556837 75245562 1390598089 1028363610 1531585205
1260995960 573666704 933596911 674402557 1230769829 172623403
1005418816 1424321892 1857300821 703571522 2117303605
529302443 418646579 274325361 217161528 1256273378 129834447
177332700 305564045 1651574882 2125023787 550804899
1651675551 310242589 1211721386 1719650353 883799426

1286966948 962764794 1912163036 671068506 76277107 338346092
1604665417 750679664 1569115921 1777288820 1756098480
845954166 1487105994 312186354 815774123 2016408437 730832933
1090099484 86086317 1987106312 1219933931 263419017
145186709 724025165 240959156 695991608 228217069 551201745
1907712995 1947867422 1435001171 1047196295 763148569
1199680559 1718264801 839425676 1538026652 1175446571
1590105340 959658925 805251743 1198720172 1805613091
144874089 1510906527 473903566 13798878 94255812 1564003050
99885196 2081362124 636453333 363304213 79065186 1360478499
604263370 775056794 1588695568 1155465115 535286141
1389079342 442982639 1582482437 4744263 1642663198 1153263590
844169939 1033206202 181226513 286791631 1992865128
986478257 1485511804 1650994571 1131352346 848934683
2124898138 1145151225 943190495 1541417540 1245036421
877068972 30387226 1608340634 956134158 1390865725 65120356
1731190952 832077645 1220585472 118993446 73673339 1663568111
1701475883 78417603 1158747661 707255825 922587542 44470216
888482339 1209379174 2037335344 1874960596 547407330
1540846267 858829294 1396342013 1518260757 2003980519
192048860 912194650 1101533292 1069117832 942581876 562390279
2025251990 185963953 627510635 1608959295 1018041598
1848096107 1727952741 1091714937 1364180570 1281944976
1170132540 375444584 1989200801 2092720083 419914800
730199492 1154615609 309766496 457676440 1702022939
1850612763 1316505735 950881304 1221389873 1173002606
1142930164 2133584523 127052251 64564349 928682751 689442530
2089816339 1114646704 1316953165 1551291986 2132688302
1017565625 1131761079 1076919591 234262547 266222407 99568484
609707131 107939561 44804919 1029621931 838139053 1199420528
1339388427 1295815494 753959819 1042517543 464837581
1704841123 116423768 1637840187 700287639 102524643
1764892438 764851988 1031207394 306851320 707184680
2145854098 1623804486 110993018 2131058752 493886463
1242754098 1060494695 728149010 1508976505 1160063179
1337856142 1616916066 1204868098 219994425 307571472
256804978 1559382853 1603386966 1010764797 454416748
2068224547 568122272 570840516 1558581086 1268409912
673365159 1175989877 2033261900 1704572553 1482841197
592962932 1702943003 959162035 703955951 1686518107
1453048498 1946710049 599529154 33713861 1308202906
1759592334 1371570003 777635325 816976784 1591564428
1085206797 1073781763 1003463633 541110115 2084546560
1457880381 461851014 505185185 2028720897 2020432100
1773595097 554602408 1048938329 1659373349 111691313
384295879 104852634 1814634316 1343457914 808808585
1353668775 649022765 608034986 1953197930 682736626

```
   1916237892 1565306616 2054306629 546389569 234799752
   1498387409 1631596366 1308581515 354367395 25222833
   1245644428 1812247776 487073847 1750829613 1693485026
   360022300 1376941062 100603786 1408960629 888830763 212295100
    1793256508 993683397 2026929416 989230775 1802491982
   1233114544 1638253540 263043320 1038828826 173506518 31797565
    456651794 80329499 578187134 691451546 1578716908 62299853
   2000033062 1933084303 87522686 1098193842 1597848432
   574596534 701539807 1143849810
5000
. . .
. . .
. . .
```

4.2. Ejercicio 2

Implemente los siguientes algoritmos:

- Bubble sort
- Heap sort
- Insertion sort
- Selection sort
- Shell sort
- Merge sort
- Quick sort

4.2.1. Bubble Sort

```cpp
#include <bits/stdc++.h>

using namespace std;

#define f(i, a, b) for (int i = a; i < b; i++)

void swap(int *a, int *b)
{
  int aux = *a;
  *a = *b;
```

```
11    *b = aux;
12  }
13
14  void bubbleSort(int A[], int tam)
15  {
16    f(i, 0, tam - 1)
17        f(j, 0, tam - i - 1)
18          if (A[j] > A[j + 1])
19              swap(&A[j], &A[j + 1]);
20  }
```

4.2.2. Heap Sort

Este algoritmo hace uso de una estructura de datos llamada **heap** , para administrar la información y se representa como un **arreglo**.

Para el desarrollo de este algoritmo se usa el **max-heap** el cual tiene la propiedad de que el nodo padre es mayor que los nodos hijos , además las operaciones básicas de los **heaps** se ejecutan en un tiempo proporcional a la altura del árbol y toma un tiempo de *O(lg n)*.

Operaciones Básicas

- **Max-Heapify** Se ejecuta en un tiempo de *O(lg n)* el cual se encarga de mantener la propiedad del Max-Heap.
- **Buil-Max-Heap** Se ejecuta en un tiempo lineal.
- **HeapSort** Se ejecuta en un tiempo de *O(n lg n)*

```
1     #include <bits/stdc++.h>
2   #include <time.h>
3   #include <iostream>
4   #include <fstream>
5
6   using namespace std;
7
8   #define f(i, a, b) for (int i = a; i < b; i++)
9   #define INF std::numeric_limits<int>::max();
10
11  bool check(int A[], int tam)
12  {
13    f(i, 1, tam) if (A[i] < A[i - 1]) return 0;
14
```

```
15    return 1;
16  }
17
18  void swap(int *a, int *b)
19  {
20    int aux = *a;
21    *a = *b;
22    *b = aux;
23  }
24  int LEFT(int i)
25  {
26    return (i << 1);
27  }
28
29  int RIGHT(int i)
30  {
31    return (i << 1) + 1;
32  }
33
34  void MaxHeapify(int A[], int tam, int i)
35  {
36    int l = LEFT(i);
37    int r = RIGHT(i);
38
39    int largest;
40    if (l < tam && (A[l] > A[i]))
41      largest = l;
42    else
43      largest = i;
44
45    if (r < tam && (A[r] > A[largest]))
46      largest = r;
47
48    if (largest != i)
49    {
50      swap(&A[i], &A[largest]);
51      MaxHeapify(A, tam, largest);
52    }
53  }
54
55  void BuildMaxHeap(int A[], int tam)
56  {
57    for (int i = tam / 2 - 1; i >= 0; i--)
58    {
59      MaxHeapify(A, tam, i);
60    }
61  }
```

```
62
63 void heapSort(int A[], int tam)
64 {
65   BuildMaxHeap(A, tam);
66   for (int i = tam - 1; i >= 0; i--)
67   {
68     swap(&A[0], &A[i]);
69     MaxHeapify(A, i, 0);
70   }
71 }
```

Insertion sort

```
1  #include <bits/stdc++.h>
2  #include <time.h>
3  #include <iostream>
4  #include <fstream>
5
6  using namespace std;
7
8  #define f(i, a, b) for (int i = a; i < b; i++)
9  #define INF std::numeric_limits<int>::max();
10
11 bool check(int A[], int tam)
12 {
13   f(i, 1, tam) if (A[i] < A[i - 1]) return 0;
14
15   return 1;
16 }
17
18 void swap(int *a, int *b)
19 {
20   int aux = *a;
21   *a = *b;
22   *b = aux;
23 }
24
25 void insertionSort(int A[], int n)
26 {
27   for (int i = 1; i < n; i++)
28   {
29     int key = A[i];
30     int j = i - 1;
31     while (j >= 0 && A[j] > key)
```

```
32      {
33        A[j + 1] = A[j];
34        j--;
35      }
36      A[j + 1] = key;
37    }
38  }
39
```

4.2.3. Selection sort

```
1
```

4.3. Shell sort

```
1  #include <bits/stdc++.h>
2  #include <time.h>
3  #include <iostream>
4  #include <fstream>
5
6  using namespace std;
7
8  #define f(i, a, b) for (int i = a; i < b; i++)
9  #define INF std::numeric_limits<int>::max();
10
11 bool check(int A[], int tam)
12 {
13   f(i, 1, tam) if (A[i] < A[i - 1]) return 0;
14
15   return 1;
16 }
17
18 void swap(int *a, int *b)
19 {
20   int aux = *a;
21   *a = *b;
22   *b = aux;
23 }
24
25 void shellSort(int A[], int tam)
```

```
{
  for (int gap = tam / 2; gap > 0; gap /= 2)
  {
    for (int i = 0, m; i < tam - gap; i++)
    {
      m = i;
      while (m >= 0 && A[m + gap] < A[m])
      {
        swap(&A[m], &A[m + gap]);
        m = m - gap;
      }
    }
  }
}
```

4.3.1. Merge sort

```
#include <bits/stdc++.h>
#include <time.h>
#include <iostream>
#include <fstream>

using namespace std;

#define f(i, a, b) for (int i = a; i < b; i++)
#define f_2(i, a, b) for (int i = a; i < b; i = i * 2)
#define MIN(a, b) ((a < b) ? a : b)
#define INF std::numeric_limits<int>::max();

bool check(int A[], int tam)
{
  f(i, 1, tam) if (A[i] < A[i - 1]) return 0;

  return 1;
}

void swap(int *a, int *b)
{
  int aux = *a;
  *a = *b;
  *b = aux;
}

void merge(int A[], int p, int q, int r)
```

```
{
  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1 + 1], R[n2 + 1];

  f(i, 0, n1) L[i] = A[p + i];
  f(i, 0, n2) R[i] = A[q + i + 1];

  L[n1] = INF;
  R[n2] = INF;
  int i = 0, j = 0;
  f(k, p, r + 1)
  {
    if (L[i] <= R[j])
    {
      A[k] = L[i];
      i++;
    }
    else
    {
      A[k] = R[j];
      j++;
    }
  }
}

void mergeSort(int A[], int p, int r)
{
  if (p < r)
  {
    int q = (p + r) / 2;
    mergeSort(A, p, q); // 0 2 ->
    mergeSort(A, q + 1, r);
    merge(A, p, q, r);
  }
}
```

4.3.2. Quick sort

```
#include <bits/stdc++.h>
#include <time.h>
#include <iostream>
#include <fstream>
```

```cpp

using namespace std;

#define f(i, a, b) for (int i = a; i < b; i++)
#define INF std::numeric_limits<int>::max();

bool check(int A[], int tam)
{
  f(i, 1, tam) if (A[i] < A[i - 1]) return 0;

  return 1;
}

void swap(int *a, int *b)
{
  int aux = *a;
  *a = *b;
  *b = aux;
}

int Partition(int A[], int p, int r)
{
  int x = A[r];
  int i = p - 1;
  for (int j = p; j < r; j++)
  {
    if (A[j] <= x)
    {
      i++;
      swap(&A[i], &A[j]);
    }
  }

  swap(&A[i + 1], &A[r]);
  return i + 1;
}

void quickSort(int A[], int p, int r)
{
  if (p < r)
  {
    int q = Partition(A, p, r);
    quickSort(A, p, q - 1);
    quickSort(A, q + 1, r);
  }
}
```

51

5. Ejercicio 3

Analizar la complejidad computacional de cada uno.

Algoritmo	Peor de los casos
Bubble Sort	$\mathcal{O}(n^2)$
Heap Sort	$\mathcal{O}(nlgn)$
Insertion Sort	$\mathcal{O}(n^2)$
Selection Sort	$\mathcal{O}(n^2)$
Shell Sort	$\mathcal{O}(n^2)$
Merge Sort	$\mathcal{O}(nlogn)$
Quick Sort	$\mathcal{O}(n^2))$

Cuadro 1: Complejidad de los algoritmo de ordenamiento

6. Ejercicio 4

Evaluar y comparar sus algoritmos usando el archivo generado (variando el tama no del vector) y construir una(s) gráfica(s) comparativa(s).

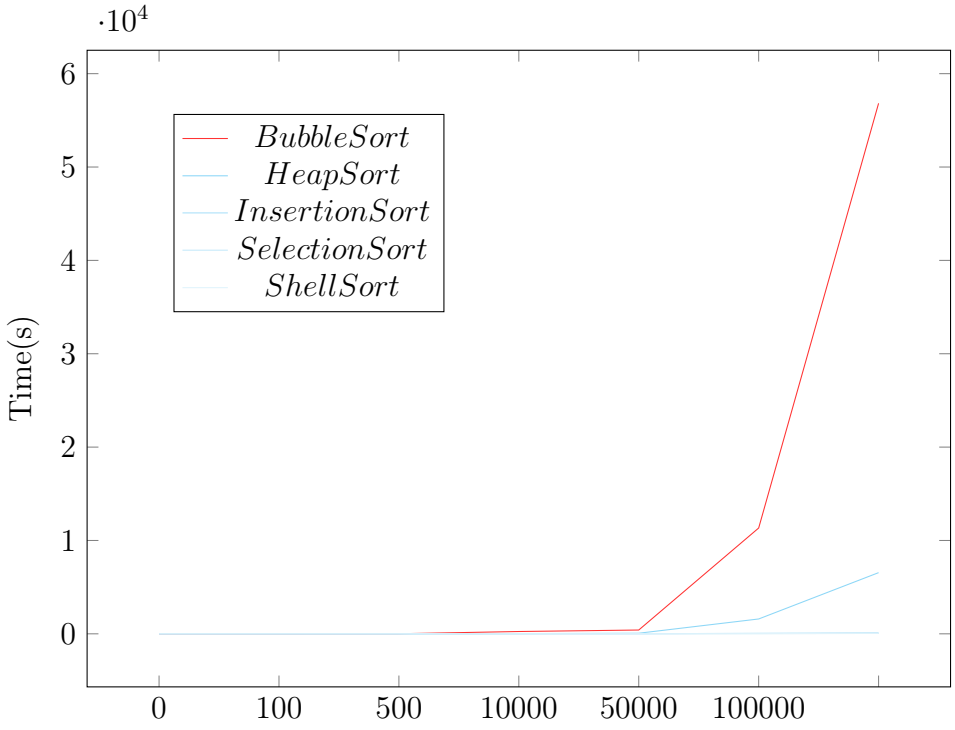

Figura 1: Array con elementos en orden ascendente