

## Experiment-1 (with A)

int Count\_inversion\_byp\_Brute\_force (arr)

// Brute force is to iterate over every element  
for every possibility  
respectively to find all the solution

and consider the result and give count inversion.

// Input is an array of Number

// Of Output = No of inverted value.

int arr[10] = { 10, 2, 1, 8, 6, 7, 5, 3, 9, 4 };

int invertedCount = 0;

for (i: 0 → len. arr - 1) do

for (j: i+1 → len. arr - 1) do

if (arr[i] > arr[j]) do

(invertedCount)++;

cout << "Number of inverted pairs is " << invertedCount;

Time complexity of this = O(n^2)

Space complexity = O(1)

For i for (i: 0 → len. arr - 1) will take O(n)

for (j: i+1 → len. arr - 1) will take O(n)

if (arr[i] > arr[j]) will take O(1)

so As they are nested then we multiply

$$O(n) \cdot O(n) = O(n^2)$$

Time complexity of this = O(n^2)

Space complexity = O(1)

Implementation of this

using bubble sort

Optimal solution.

int merge\_and\_count(arr, left, right)

// Find inversion :- Find the inversion by  
// divide & conquer approach to a optimal solution using divide  
// & conquer method

// Input : A array arr, left index  
and the right index

// Output : The no of count inversion.

ob (l - r) <= 1 inversion = 0

ob inversion = 0

ob  $\sum_{i=0}^{mid} arr[i] < \sum_{i=mid+1}^{right}$

if (left < right)

int mid = left + (right - left) / 2

inversion += merge\_and\_count(arr, left, mid)

inversion += merge\_and\_count(arr, mid + 1, right)

(a) cross inversion += merge\_and\_count(arr, left, mid, right)

(b) left < mid < mid + 1

(c) mid < right & arr[mid] < arr[mid + 1]

int merge\_and\_count(arr, left, mid, right)

inversion = 0

# input // Cross inversion : Merge and count inversion

// input : a array, a left index which  
start, the mid index and, weight

// output . . . inversion

inversion = 0

left\_subarray = [left : mid + 1]

right\_subarray = [mid + 1 : right]

i = 0, j = 0, k = left.

while ( i < left\_subarray.size + & j < right\_subarray.size )

if ( left\_subarray[i] < right\_subarray[j] )

arr[k++] = left\_subarray[i++];

else do

arr[k++] = right\_subarray[j++]

inversions += left\_subarray.size - i

while i → left\_subarray.size do

arr[k++] = left\_subarray[i++];

while j → right\_subarray.size do

arr[k++] = right\_subarray[j++]

return inversion.

vector<int> countStudentInversion(arr)

if zero inversion = 0

one inversion = 0

two inversion, three inversion = 0 ]

// count no of inversion :- count the zero, one & two

1s = three inversion of for  
(original & each case) NOT

// input we take a array as a input

// output :- we get the zero, one two  
three inversion count

(example given) NM Inversion 5 times

if inversion == 0

if exactly zero inversion + = 1 tri condition 11

else if one inversion == 1

One inversion + = 1

else if inversion == 2

else if two inversion + = 1 tri condition 11

else if inversion == 3

else if three inversion + = 1 tri condition 11

else or else from 6

other inversion + = 1

if zero inversion, one inversion, two inversion  
three inversion, other inversion ]

((binary search + (infix + 1 more) search))

Time complexity of n log n + 3 more iterations

The complexity of the code is  $O(n)$  as we

(O, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 580, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 610, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 630, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 660, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 690, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 710, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 730, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 760, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 790, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 810, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 830, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 860, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 890, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 910, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 930, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 960, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 990, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1010, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1030, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1060, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1090, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1110, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1130, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1160, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1190, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1210, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1230, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1260, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1290, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1310, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1330, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1360, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1390, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1410, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1430, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1460, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1490, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1510, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1530, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1560, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1590, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1610, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1630, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1660, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1690, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1710, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1730, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1760, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1790, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1810, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1830, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1860, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1

$$\therefore \text{Total time (Completion)} = T(n)$$

$$T(n) = 2 T(n/2) + O(n)$$

Using Master theorem.

$$a=2, b=2 \Rightarrow d=1$$

$$a=6^d \text{ as } 2=2^1$$

$$\therefore T(n) = O(n \log n)$$

## Experiment - 2

string  
int manualMult (num1, num2)

// Multiply int: Multiply integers by creating a  
array of size 10, 50  
1 = 1000 digits

// input : We will take two string  
num1 and num2

// output : a string which is product  
of num1 and num2

if (num1 == 0 || num2 == 0)

return "0";

reverse (num1.begin(), num1.end())

reverse (num2.begin(), num2.end())

vector<int> result (num1.size() + num2.size(), 0)

for (int i = 0; i < num1.size(); i++)

```

    for i = 0 → num1.size(); i++ do
        for k j : 0 → num2.size() do
            product int mul = (num1[i] - '0') * (num2[j] - '0')
            sum from result [i+j] += mul
            if result [i+j] >= 10
                result [i+j+1] += result [i+j] / 10

```

$(i + 1) \text{th digit of result} [i+j] = 10$

$((0^1 - 0) \times 0) + ((0^2 - 0) \times 1) = 0$

```

while (result · bck == 0) do
    result · pop · bck

```

$(0^2 - 0) \times 0 + ((0^1 - 0) \times 1) > (0^1 - 0) \times 1$

loop + ~~rest of str~~

$(0^1 - 0) \times 1 > (0^1 - 0) \times 1$

~~for i = 0 → 0~~

for result · size - 1 → 0

result str ~~<~~ push = result [i] + '0'

return result str ~~<~~ n = str

The time complexity  $n = 1 \times$

(if add digits of num1 = 0)

The complexity will above digits of  
 $n = 1 \times$  (no. of digits of num2)  $n_2$  that are  
 $\Theta(n_1 \times n_2)$

$(1 \times 1 \times 1) + (1 \times 1 \times 1) = 19$

$(0 \times 0 \times 0) + (0 \times 0 \times 0) = 0$

$(0 \times 1 \times 1) + (0 \times 0 \times 1) = 0$

; the last mixture

at string::Karatsuba(string num1, num2)

// multiplication of two digit :- multiplying num1 and  
num2 by Karatsuba.

// input : num = two strings num1 and num2

// output : string which is product

of num1 and num2

if (num1.length() == 1 & num2.length() == 1)

result = (num1[0] - '0') \* (num2[0] - '0')

return to\_string(result);

while (num1.length() < num2.length()) num1 += '0';  
+ num1

while (num2.length() < num1.length()) num2 += '0';

num2 = '0' + num2;

n ← 1 → size / 2

(0) + (1 + 2 + ... + n - 1) = num2.length / 2

half = n / 2

x1 = num1.substr(0, half)

x0 = num2.substr(half)

y1 = num1.substr(half, half)

y0 = num2.substr(half, half)

P1 = Karatsuba(x1, y1)

P2 = Karatsuba(x0, y0)

P3 = Karatsuba((x1, x0), (y1, y0))

return result;

~~part 1~~ Test Case ~~#~~ 1 - Part 1 (1)  
Experiment - 7 = 5 min  
~~using binomial~~

- 1) arr = [1, 2, 3, 4, 5]  
ans = 0
- 2) arr = [5, 4, 3, 2, 1]  
ans = 10
- 3) arr = [2, 3, 8, 6, 1]  
ans = 5
- 4) arr = []  
ans = 0
- 5) arr = [6, 9]  
ans = 0

Experiment - 2

- 1) num 1 = "12"  
num 2 = "34"  
ans = "408"
- 2) num 1 = "12345678901234567890"  
num 2 = "98765432109876543210"  
ans = "1218326311370217952232463  
80111263526900"
- 3) num 1 = "0"  
num 2 = "123456"  
ans = 0

4) Test num 1 = " 123 " for  
num 2 = " 123 " ans

ans forward input.

5) num 1 = " 12abc " o = 0  
num 2 = " 123 " l = 3

ans forward input.

6) num 1 = " - 1 " l = 2  
num 2 = " 12345 " l = 5

ans = invalid input.

Test = num (2)

l = forward

o = end (0)

(1)

" 123 " = num (1)