

---

# CSE 424 HOMEWORK 1

---

SORTING OPTIMIZATION PROBLEM

MINIMUM SPANNING TREE (MST) PROBLEM

HÜSNÜ AKÇAK  
161044112

# 1 Define sorting problem as an optimization problem

In this homework, a different approach to sorting problem will be proposed to be able to define it as an optimization problem.

## 1.1 Problem Definition

An integer list will be given and output will be an ascending order sorted integer list. This sorted output list will be chosen among all permutations of given integer list.

## 1.2 Domain

$$D_{L1,L2,...,LN!} \quad (1)$$

L is a permutation of given integer list

## 1.3 Decision Variables

$$L = \{x_1, x_2, x_3, \dots, x_n\} \quad (2)$$

n = number of integers in the list

Decisions will made according to sorting of the x values

## 1.4 Constraints

$$\forall x \in L \quad x_i \geq x_{i-1} \quad (3)$$

## 1.5 Objective Function

$$MAX(\sum_{i=1}^S i) \quad (4)$$

S is last index of the sorted sub-list, the objective is to maximize the portion of sorted sub-list which is the list itself.

# 2 Design and implement brute force solutions for both problems (MST and sorting optimization problem).

## 2.1 Brute Force Solution for Sorting Optimization Problem

To test the program, 50 lists of integer is given and their length are from 1 to 50. Values of integers are between 0 and 10000.

The sorted version of lists is obtained by choosing the sorted permutation of itself, and since this is a brute force implementation, firstly all permutations are calculated by a recursive function then the sorted one is picked among them.

As it will be seen the picture below first 7 list is sorted in no-time, but later on the calculation time is increased by factor of the new size. Eventually, it is expected that the program will take too long time to complete and the stack size will be consumed and a stack overflow exception will be thrown. We also could see the memory exception in the picture below.

### 2.1.1 Complexity

per() function calculates permutations of the given list, its time complexity is  $O(n!)$

After all permutations calculated sorted one is picked among them, its complexity is  $O(n!)$

$n! + n! = O(N!)$

```

Run: bruteForceSorting
C:\Users\HÜsnÜ\PYcharmProjects\optimization\venv\Scripts\python.exe C:/Users/HÜsnÜ/PYcharmProjects/optimization/hw1/bruteForceSorting.py
list[0] is sorted in 0.0 seconds: [5732]
list[1] is sorted in 0.0 seconds: [6799, 9747]
list[2] is sorted in 0.0 seconds: [7707, 9287, 9409]
list[3] is sorted in 0.0 seconds: [3392, 5140, 7035, 8486]
list[4] is sorted in 0.0 seconds: [271, 313, 2555, 3721, 6904]
list[5] is sorted in 0.0 seconds: [798, 5642, 5784, 6295, 6370, 6448]
list[6] is sorted in 0.0 seconds: [621, 1235, 2737, 4001, 6019, 6828, 8632]
list[7] is sorted in 0.078 seconds: [1606, 3506, 4232, 4473, 5766, 6758, 9889, 9974]
list[8] is sorted in 0.687 seconds: [1315, 1693, 4584, 5501, 7024, 7185, 7523, 8991, 9860]
list[9] is sorted in 9.331 seconds: [1438, 2637, 3273, 3531, 5217, 5326, 5462, 5895, 6321, 9026]
list[10] is sorted in 95.548 seconds: [15, 485, 502, 1561, 2848, 3050, 4878, 4937, 5606, 7159, 7273]
Traceback (most recent call last):
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 126, in <module>
    sort_lists(lists)
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 79, in sort_lists
    sorted_list = (get_sorted_list(sample_lists[curr_index]))
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 90, in get_sorted_list
    per(unsorted_list, [], all_perms)
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 120, in per
    per(sub_list, curr, sol)
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 120, in per
    per(sub_list, curr, sol)
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 120, in per
    per(sub_list, curr, sol)
  [Previous line repeated 9 more times]
  File "C:\Users\HÜsnÜ\PYcharmProjects\optimization\hw1\bruteForceSorting.py", line 113, in per
    sol.append(curr.copy())
MemoryError

```

## 2.2 Brute Force Solution for MST

## 3 Design and implement a backtracking algorithm for the sort- ing optimization problem.

In backtracking method, finding permutations of the given list is used again but at this time recursive function is returned when an unsorted sequence is encountered.

If we have a list as [5, 2, 3], when [5, 2] is encountered the function is returned without producing [5, 2, 3]

The below picture shows that even though some solutions take relatively long execution time, all sample lists are sorted in a reasonable time.

```

Run: backtracking_sorting
list[23] is sorted in 0.022 seconds: [[5, 392, 815, 1061, 2151, 2386, 2647, 3091, 3126, 3922, 3931, 4978, 5423, 5462, 5642, 5780, 6250, 6349, 6444, 7428, 8527, 9010, 9573, 9713]]
list[24] is sorted in 0.016 seconds: [[236, 611, 750, 1849, 1970, 2902, 2969, 3073, 3274, 4243, 4810, 5235, 5544, 5698, 5814, 5902, 6569, 6682, 7748, 8208, 8548, 9228, 9865, 9916]]
list[25] is sorted in 0.0 seconds: [[53, 1264, 1868, 2466, 2522, 2538, 3421, 3734, 3805, 4134, 4163, 4355, 4747, 4828, 5209, 5428, 6422, 6995, 8012, 8142, 8703, 8895, 9017, 9156]]
list[26] is sorted in 0.062 seconds: [[792, 1105, 1467, 1581, 1589, 1625, 1677, 2078, 2221, 3279, 3430, 3872, 4061, 4189, 4458, 5026, 6201, 6648, 7585, 7769, 8448, 8563, 8636, 90]]
list[27] is sorted in 0.108 seconds: [[61, 1123, 1126, 1261, 1429, 1989, 2188, 2538, 2789, 2820, 3383, 4053, 4894, 5001, 5116, 6319, 6760, 7374, 7629, 7978, 8864, 8567, 8675, 886]]
list[28] is sorted in 0.022 seconds: [[53, 410, 622, 1185, 1742, 1992, 2013, 2945, 3224, 3388, 4073, 4572, 4877, 4926, 5221, 5272, 5279, 6594, 6970, 7007, 7419, 7615, 8206, 8612]]
list[29] is sorted in 0.091 seconds: [[582, 736, 1253, 1504, 1647, 2706, 3157, 3161, 3990, 4024, 4129, 4884, 5165, 5275, 5892, 6405, 6416, 6775, 6932, 7972, 8031, 8077, 8148, 820]]
list[30] is sorted in 0.119 seconds: [[222, 359, 371, 826, 1342, 2128, 2708, 2784, 2860, 4015, 4059, 4620, 5275, 5430, 5984, 5955, 6275, 6634, 7271, 7293, 8006, 8212, 8462, 8489]]
list[31] is sorted in 0.083 seconds: [[27, 112, 242, 843, 981, 1001, 1510, 2027, 2516, 2980, 3252, 3850, 4149, 4200, 4274, 4730, 4898, 5242, 5304, 5413, 5672, 6129, 6292, 6665, 6]]
list[32] is sorted in 0.129 seconds: [[769, 943, 1043, 1284, 1961, 2583, 2668, 2974, 3432, 3454, 3614, 3914, 4906, 4949, 5020, 5329, 5526, 5761, 5832, 6281, 6664, 6784, 7413, 767]]
list[33] is sorted in 0.174 seconds: [[513, 558, 1529, 1866, 1869, 2486, 2639, 2751, 3332, 3531, 3673, 3835, 5414, 5782, 5929, 6003, 6343, 6469, 6489, 6922, 7138, 7163, 7413, 744]]
list[34] is sorted in 0.212 seconds: [[390, 1064, 1280, 1391, 1431, 1779, 2265, 2562, 2795, 2813, 3064, 3534, 3658, 4780, 4939, 5148, 5241, 5401, 5410, 5419, 5753, 5990, 6005, 63]]
list[35] is sorted in 0.059 seconds: [[218, 456, 917, 1268, 1475, 2650, 3291, 3678, 4031, 4115, 4118, 4197, 4315, 4509, 4639, 5023, 5542, 5904, 5928, 6321, 7345, 7472, 7620, 7704]]
list[36] is sorted in 0.164 seconds: [[464, 936, 1117, 1207, 1377, 1609, 1681, 2186, 2204, 2618, 2859, 2940, 3282, 3324, 3326, 3462, 3520, 3539, 3608, 3815, 4656, 5177, 5435, 549]]
list[37] is sorted in 1.87 seconds: [[83, 252, 644, 714, 914, 1365, 1420, 1430, 1520, 1591, 1813, 1855, 1867, 2165, 3131, 3387, 3917, 4282, 4295, 4873, 4903, 5527, 5572, 5897, 59]]
list[38] is sorted in 0.202 seconds: [[883, 925, 1388, 1605, 1635, 1720, 1928, 1969, 2001, 2503, 2753, 2944, 3100, 3180, 3215, 3571, 4230, 5256, 5309, 5481, 5527, 5595, 5603, 571]]
list[39] is sorted in 1.264 seconds: [[295, 831, 971, 1094, 1398, 1479, 1857, 2044, 2243, 2469, 2542, 2632, 2713, 2859, 3404, 3816, 4065, 4090, 4259, 4535, 4598, 4780, 4840, 5057]]
list[40] is sorted in 0.607 seconds: [[38, 727, 765, 1147, 1310, 1426, 1834, 1846, 1892, 2332, 2557, 2700, 2722, 2849, 2871, 2875, 3060, 3179, 3191, 3474, 4011, 4093, 4365, 4378]]
list[41] is sorted in 0.596 seconds: [[233, 1030, 1620, 1710, 2097, 2465, 3285, 3294, 3383, 3763, 3898, 4689, 4793, 5028, 5062, 5168, 5207, 5224, 5805, 5950, 6053, 6636, 6772, 68]]
list[42] is sorted in 0.435 seconds: [[95, 178, 292, 472, 595, 694, 927, 1273, 1423, 1701, 2240, 2255, 2763, 3388, 3485, 3595, 3947, 3989, 4070, 4500, 4589, 4821, 4842, 4851, 486]]
list[43] is sorted in 0.587 seconds: [[166, 197, 374, 544, 712, 1660, 1839, 2191, 2204, 3108, 3254, 3497, 3913, 4122, 4633, 4643, 4725, 4961, 5375, 6061, 6180, 6266, 6374, 6468]]
list[44] is sorted in 3.535 seconds: [[82, 736, 973, 1909, 2258, 2304, 2349, 2484, 3045, 3101, 3135, 3298, 3478, 3572, 3621, 3729, 4072, 4142, 4399, 4513, 5130, 5326, 5403, 5621]]
list[45] is sorted in 0.639 seconds: [[61, 95, 115, 502, 525, 722, 1012, 1249, 1344, 1451, 1712, 2090, 2142, 2241, 2433, 2488, 2904, 3024, 3156, 3167, 3218, 3232, 3242, 3450, 345]]
list[46] is sorted in 0.634 seconds: [[106, 153, 288, 302, 539, 862, 1136, 1676, 1936, 1988, 2115, 2194, 2416, 2480, 2797, 3398, 3604, 3666, 3832, 3920, 3992, 4421, 4835, 4996, 5]]
list[47] is sorted in 8.334 seconds: [[213, 315, 871, 970, 994, 1101, 1273, 1406, 1760, 1797, 2452, 2642, 3036, 3117, 3327, 3738, 3750, 4128, 4134, 4430, 4645, 4961, 5414, 5681]]
list[48] is sorted in 0.98 seconds: [[167, 316, 358, 814, 931, 1600, 1945, 2172, 2320, 2557, 2674, 2808, 2831, 2867, 2961, 3341, 3738, 3740, 4088, 4115, 4147, 4288, 4453, 4527, 4]]
Process finished with exit code 0

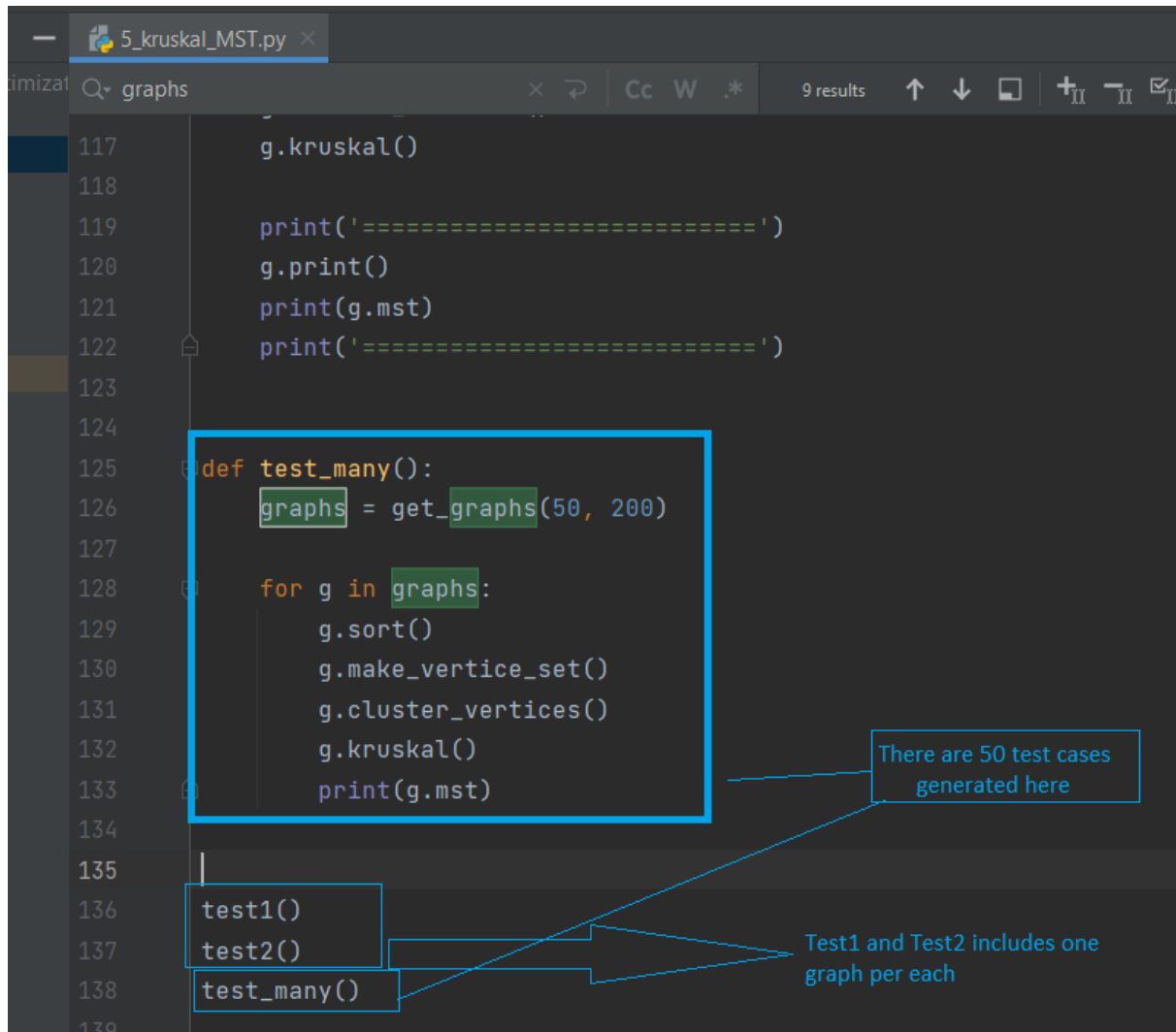
```

4 Design and implement a branch and bound solution for the MST problem.

5 Implement a well-known greedy MST algorithm.

Kruskal's MST algorithm is implemented. This greedy algorithm picks edges from smallest to bigger and in each iteration it makes sure the selected smallest edge does not lead a cycle.

To test the implemented algorithm 52 graph is used and 2 of them is also demonstrated on paper.

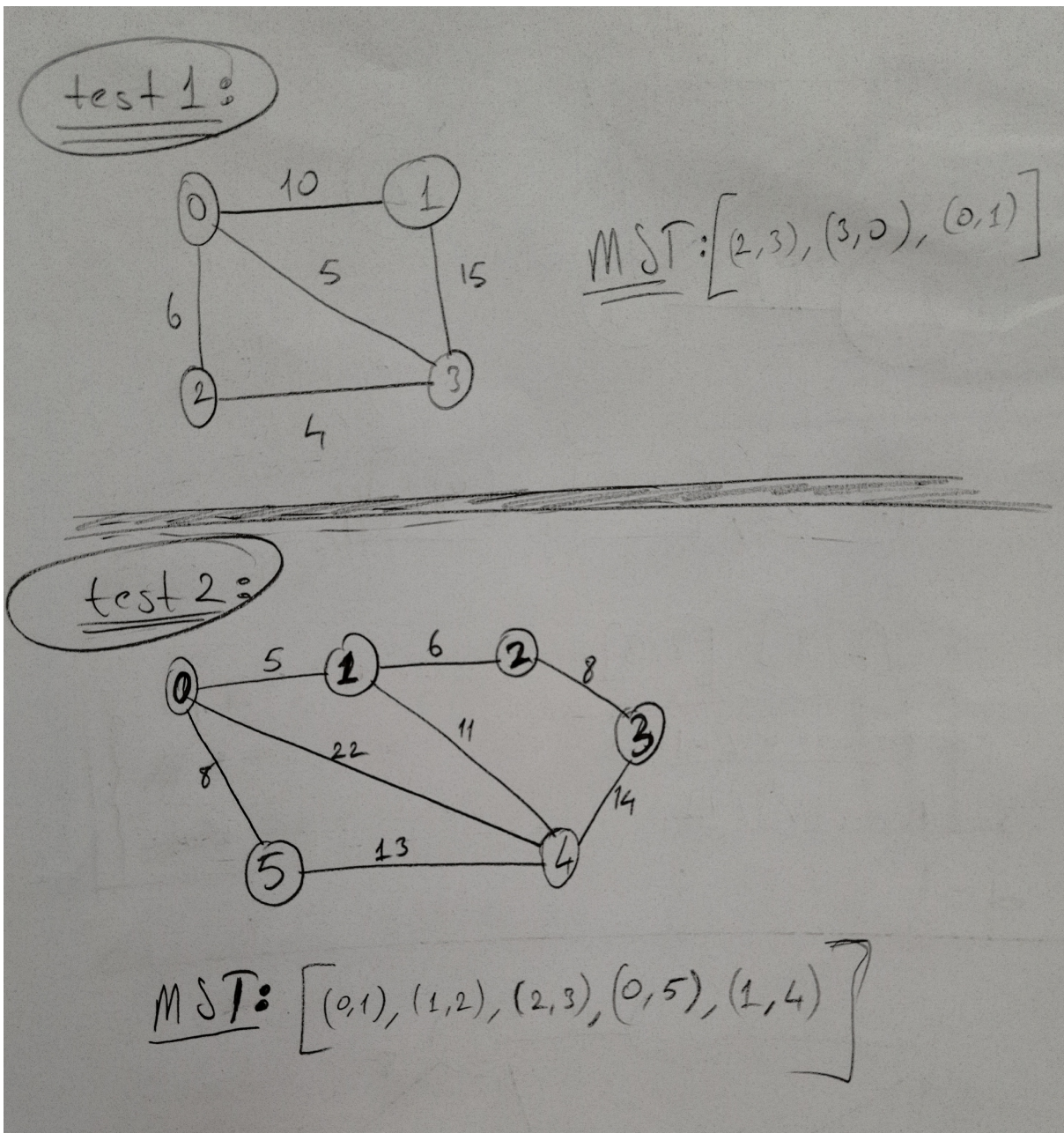


```
117 g.kruskal()
118
119 print('=====')
120 g.print()
121 print(g.mst)
122 print('=====')
123
124
125 def test_many():
126     graphs = get_graphs(50, 200)
127
128     for g in graphs:
129         g.sort()
130         g.make_vertice_set()
131         g.cluster_vertices()
132         g.kruskal()
133         print(g.mst)
134
135
136 test1()
137 test2()
138 test_many()
139
```

There are 50 test cases generated here

Test1 and Test2 includes one graph per each

Test1() and Test2() are easy to verify with the following handwriting demonstration, the other 50 graphs are auto generated and put here to provide alternative observation.



Handwriting demonstration for Test1() and Test2() functions

```

Run: 5_kruskal_MST x
C:\Users\HÜsnÜ\PycharmProjects\optimization\venv\Scripts\python.exe C:/Users/HÜsnÜ/PycharmProjects/optimization/hw1/5_kruskal_MST.py
=====
2 3 4
0 3 5
0 2 6
0 1 10
1 3 15
[[2, 3], [0, 3], [0, 1]]
=====
0 1 5
1 2 6
0 5 8
2 3 8
4 1 11
5 4 13
3 4 14
0 4 22
[[0, 1], [1, 2], [0, 5], [2, 3], [4, 1]]
=====
Process finished with exit code 0

```

**Graphs**

**Minimum Spanning Trees**

Terminal output for Test1() and Test2()

## 6 Implement a greedy algorithm for the sorting optimization problem.

In this part, best available option is picked in each iteration of finding permutation of the given list. As a result when we complete one permutation of the list we have reached the sorted list, so only one permutation is calculated.

Picking best available option is to pop minimum value from input list and put that newly-built result list, so at the end a sorted list is obtained.

```

Run: 6_greedy_sorting_optimization
List[27] is sorted in 0.0 seconds: [61, 1123, 1126, 1261, 1429, 1989, 2188, 2538, 2789, 2820, 3383, 4053, 4894, 5001, 5116, 6319, 6760, 7374, 7629, 7978, 8064, 8567, 8675, 8868,
List[28] is sorted in 0.0 seconds: [53, 410, 622, 1185, 1742, 1992, 2013, 2945, 3224, 3388, 4073, 4572, 4877, 4926, 5221, 5272, 5279, 6594, 6970, 7007, 7419, 7615, 8206, 8612, 88
List[29] is sorted in 0.0 seconds: [582, 736, 1253, 1504, 1647, 2706, 3157, 3161, 3990, 4024, 4129, 4884, 5165, 5275, 5892, 6405, 6416, 6775, 6932, 7972, 8031, 8077, 8148, 8209,
List[30] is sorted in 0.0 seconds: [222, 359, 371, 826, 1342, 2128, 2708, 2784, 2860, 4015, 4059, 4620, 5275, 5430, 5904, 5955, 6275, 6634, 7271, 7293, 8006, 8212, 8462, 8489, 85
List[31] is sorted in 0.0 seconds: [27, 112, 242, 843, 981, 1001, 1510, 2027, 2516, 2980, 3252, 3850, 4149, 4200, 4274, 4730, 4898, 5242, 5304, 5413, 5672, 6129, 6292, 6665, 6882
List[32] is sorted in 0.0 seconds: [769, 943, 1043, 1284, 1961, 2583, 2668, 2974, 3432, 3454, 3614, 3914, 4906, 4949, 5020, 5329, 5526, 5761, 5832, 6281, 6664, 6784, 7413, 7679,
List[33] is sorted in 0.0 seconds: [513, 558, 1529, 1866, 1869, 2486, 2639, 2751, 3332, 3531, 3673, 3835, 5414, 5782, 5929, 6003, 6343, 6469, 6489, 6922, 7138, 7163, 7413, 7444,
List[34] is sorted in 0.0 seconds: [390, 1064, 1288, 1391, 1431, 1779, 2265, 2562, 2795, 2813, 3064, 3534, 3658, 4780, 4939, 5148, 5241, 5401, 5410, 5419, 5753, 5990, 6005, 6375,
List[35] is sorted in 0.0 seconds: [218, 456, 917, 1268, 1475, 2650, 3291, 3678, 4031, 4115, 4118, 4197, 4315, 4509, 4639, 5023, 5542, 5904, 5928, 6321, 7345, 7472, 7620, 7704, 7
List[36] is sorted in 0.0 seconds: [464, 936, 1117, 1287, 1377, 1609, 1681, 2186, 2204, 2618, 2859, 2940, 3282, 3324, 3326, 3462, 3520, 3539, 3608, 3815, 4656, 5177, 5435, 5498,
List[37] is sorted in 0.0 seconds: [83, 252, 644, 714, 914, 1365, 1420, 1430, 1520, 1591, 1813, 1855, 1867, 2165, 3131, 3387, 3917, 4282, 4295, 4873, 4903, 5527, 5572, 5897, 5926
List[38] is sorted in 0.0 seconds: [883, 925, 1388, 1605, 1635, 1720, 1928, 1969, 2301, 2503, 2753, 2964, 3100, 3180, 3215, 3571, 4230, 5256, 5309, 5481, 5527, 5595, 5603, 5711,
List[39] is sorted in 0.0 seconds: [295, 831, 971, 1094, 1398, 1479, 1857, 2044, 2243, 2469, 2542, 2632, 2718, 2859, 3404, 3816, 4065, 4090, 4259, 4535, 4598, 4780, 4840, 5057, 5
List[40] is sorted in 0.0 seconds: [38, 727, 765, 1147, 1310, 1426, 1834, 1846, 1892, 2332, 2557, 2700, 2722, 2849, 2871, 2875, 3060, 3179, 3191, 3474, 4011, 4093, 4365, 4378, 44
List[41] is sorted in 0.0 seconds: [233, 1030, 1620, 1710, 2897, 2465, 3285, 3294, 3383, 3763, 3898, 4689, 4793, 5028, 5062, 5168, 5207, 5224, 5805, 5950, 6053, 6636, 6772, 6889,
List[42] is sorted in 0.0 seconds: [95, 178, 292, 472, 595, 694, 927, 1273, 1423, 1701, 2240, 2255, 2763, 3388, 3485, 3595, 3947, 3989, 4070, 4500, 4589, 4821, 4842, 4851, 4867,
List[43] is sorted in 0.0 seconds: [166, 197, 374, 544, 712, 1668, 1839, 2191, 2204, 3108, 3254, 3497, 3913, 4122, 4633, 4643, 4725, 4961, 5375, 6061, 6180, 6266, 6374, 6468, 662
List[44] is sorted in 0.0 seconds: [82, 736, 973, 1909, 2258, 2304, 2349, 2484, 3045, 3101, 3135, 3298, 3478, 3572, 3621, 3729, 4072, 4142, 4399, 4513, 5130, 5326, 5403, 5621, 58
List[45] is sorted in 0.0 seconds: [61, 95, 115, 502, 525, 722, 1012, 1249, 1344, 1451, 1712, 2898, 2142, 2241, 2433, 2488, 2904, 3024, 3156, 3167, 3218, 3232, 3242, 3450, 3450,
List[46] is sorted in 0.0 seconds: [106, 153, 288, 302, 539, 862, 1136, 1676, 1936, 1988, 2115, 2194, 2416, 2480, 2797, 3398, 3604, 3666, 3832, 3920, 3992, 4421, 4835, 4996, 5348
List[47] is sorted in 0.0 seconds: [213, 315, 871, 970, 994, 1101, 1273, 1406, 1760, 1797, 2452, 2642, 3036, 3117, 3327, 3738, 3750, 4128, 4134, 4430, 4645, 4961, 5414, 5681, 587
List[48] is sorted in 0.0 seconds: [167, 316, 358, 814, 931, 1600, 1945, 2172, 2320, 2557, 2674, 2808, 2831, 2867, 2961, 3341, 3738, 3740, 4088, 4115, 4147, 4288, 4453, 4527, 466
Process finished with exit code 0

```