《交通地理信息系统》

实 验 报 告

学 号 2020112921

姓 名 刘欣豪

年 级 2020级

专 业 交通工程

院 系 交通运输与物流学院

二0二二年十月

# 实验一 最短路算法分析报告

1. 实验目的

|  |
| --- |
| 1. 掌握最短路LC、LS算法原理。 2. 掌握算法的程序实现方法。 3. 掌握程序的调试方法及测试方法。 4. 掌握算法时间复杂度分析方法 |

1. 实验任务

|  |
| --- |
| 1、读取Sioux Falls (sf)、ChicagoSketch (cs)、ChicagoRegional(cr)网络  2、依据GLC的代码，编写LC 和LS算法函数的编写；  3、针对sf、cs和cr网络，分别挑选20个O-D对计算最短路并打印输出；  4、在cs和cr网络上，对比GLC/LC/LS三个算法效率，以图表形式体现，O-D 对数不限，并对算法复杂度进行分析。 |

1. 实验环境

|  |
| --- |
| * 1. 硬件环境  1. 计算机：OMEN by HP Laptop 15 2. CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 3. RAM：16GB    1. 软件环境 4. 操作系统：Windows10家庭中文版 5. 开发工具：PyCharm Community Edition 2021.3.1 |

1. 实验步骤及结果

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4.1网络读取代码分析  对于路网信息，存储在.txt文件中。  以sf路网为例，sf\_net存储路网中边的信息，包含节点数量、边数量以及边的起讫点、容量、长度等信息；sf\_nod存储各节点的相对位置信息。  设置Node与Link类存储路网信息。  对于Node类，在最短路中相关的属性为：  node\_id：表示节点编号，初始读取时赋值  l\_in: 存储流入节点的边，利用函数set\_l\_in赋值  l\_out: 存储流出节点的边，利用函数set\_l\_out赋值  u: 存储最短路中cost，利用函数set\_SPP\_u赋值  p: 存储最短路中节点的前置标号，利用函数set\_SPP\_p赋值  4.2 LC与LS算法函数编写  4.2.1 LC算法：  标号检验算法从初始点出发，循环遍历以点为起点的弧是否满足最优性原则，最终检验完整个网络，得到最短网络。  **流程图**如下：    图一：LC算法流程图  **伪代码**（为教学PPT第35页伪代码）：    利用读取的网络编写**代码**如下：   1. #label correcting algorithm 2. **def** SPP\_LC(o\_id,node): 3. node[o\_id].set\_SPP\_u(0) # 初始化起点权值 4. **for** t **in** node[1:]: 5. t.set\_SPP\_p(-1) # 初始化点的前置节点为-1 6. **if** t.node\_id != o\_id: 7. t.set\_SPP\_u(float('inf'))   # 初始化其他点权值为‘inf’ 9. C = [node[o\_id]] # C为SEL集，将初始点位放入 10. **while** len(C)!=0:    # 循环结束条件：当SEL集为空时 11. i = C[0] 12. # 遍历以i节点为起点的边 13. **for** l **in** i.l\_out: 14. n = node[l.head\_node] 15. **if** i.u+l.length<n.u: # 更新条件 16. n.u = i.u+l.length 17. n.p = i 18. **if** n **not** **in** C: # 若n节点不在C中，将n放入C 19. C.append(n) 20. **del** C[0] # 将C[0]从SEL中删除 21. shortestpath\_p\_list = [0] 22. **for** t **in** node[1:]: 23. shortestpath\_p\_list.append(t.p) 24. **return** shortestpath\_p\_list   4.2.1 LS算法：  标号设置算法以节点为核心，每次寻找非可达节点中最小权值，将该节点放入可达集中，直至所有节点进入可达集。LS里用贪心思想，每次装入点后，可以确定初始点到该点的最短距离。  **流程图**如下：    图二：LS算法流程图  **伪代码**（为教学PPT第49页伪代码）：    利用读取的网络编写**代码**如下：   1. #label setting algorithm 2. **def** SPP\_LS(o\_id,node): 3. node[o\_id].set\_SPP\_u(0)# 初始化起点权值 4. **for** t **in** node[1:]: 5. t.set\_SPP\_p(-1) # 初始化点的前置节点为-1 6. **if** t.node\_id != o\_id: 7. t.set\_SPP\_u(float('inf'))   # 初始化其他点权值为‘inf’ 9. C = list(range(1,len(node))) # 将初始点位编号放入C集 10. **while** len(C)!=0: 11. mmin = float('inf') 12. minpos = -1 13. # 寻找最小值所在位置 14. **for** i **in** C: 15. **if** node[i].u<mmin: 16. mmin = node[i].u 17. minpos = i 18. # 检验条件，当存在不可达集时退出 19. **if** minpos == -1: 20. **break** 21. C.remove(minpos) 22. # 遍历以编号为minpos节点为起点的边 23. **for** j **in** node[minpos].l\_out: 24. n = node[j.head\_node] 25. **if** n.u > mmin + j.length: # 更新条件 26. n.u = mmin + j.length 27. n.p = node[minpos] 29. shortestpath\_p\_list = [0] 30. **for** t **in** node[1:]: 31. shortestpath\_p\_list.append(t.p) 32. **return** shortestpath\_p\_list   4.3 最短路输出测试  4.3.1测试函数逻辑编写  首先设置随机数seed，以radom.randint(1,len(node)-1)生成20个od对。循环20次打印最短路径与最短路长度。  流程图如下：    图三：测试代码流程图  代码函数如下：   1. **def** al\_rand20(): 2. random.seed(4)  # 设置随机数seed 3. # 生成随机od表 4. m = len(NODE)-1 # m为标号的上界 5. o\_id = [random.randint(1, m) **for** i **in** range(20)] 6. d\_id = [random.randint(1, m) **for** i **in** range(20)] 7. # 循环打印每次的结果 8. **for** i **in** range(0,20): 9. Astarsp, Astarspnode = Test\_SPP\_LS(o\_id[i], d\_id[i]) 10. **print**(Astarspnode) 11. get\_length(Astarsp) 12. # 进行可视化绘图 13. **if** i == 0: 14. Visual\_path(NODE, Astarspnode)   调用代码片段如下：    可视化代码如图：    4.3.2测试结果   * **对于sf：**   生成的od随机表：  o\_id: [8, 10, 4, 24, 13, 16, 5, 3, 3, 1, 13, 18, 10, 2, 8, 17, 18, 12, 9, 6]  d\_id: [4, 9, 7, 1, 21, 9, 9, 7, 6, 10, 10, 21, 24, 12, 3, 20, 11, 22, 13, 17]   |  |  |  | | --- | --- | --- | | **编号** | **最短路径** | **最短路长度** | | **1** | **[8, 6, 5, 4]** | **8** | | **2** | **[10, 9]** | **3** | | **3** | **[4, 5, 6, 8, 7]** | **11** | | **4** | **[24, 13, 12, 3, 1]** | **15** | | **5** | **[13, 24, 21]** | **7** | | **6** | **[16, 10, 9]** | **7** | | **7** | **[5, 9]** | **5** | | **8** | **[3, 4, 5, 6, 8, 7]** | **15** | | **9** | **[3, 4, 5, 6]** | **10** | | **10** | **[1, 3, 4, 5, 9, 10]** | **18** | | **11** | **[13, 12, 11, 10]** | **14** | | **12** | **[18, 20, 21]** | **10** | | **13** | **[10, 15, 22, 21, 24]** | **14** | | **14** | **[2, 1, 3, 12]** | **14** | | **15** | **[8, 6, 5, 4, 3]** | **12** | | **16** | **[17, 19, 20]** | **6** | | **17** | **[18, 16, 10, 11]** | **12** | | **18** | **[12, 13, 24, 21, 22]** | **12** | | **19** | **[9, 10, 11, 12, 13]** | **17** | | **20** | **[6, 8, 16, 17]** | **9** |   表1：sf最短路表    图1：sf网络编号1的最短路可视化   * **对于cs：**   生成的od随机表：  o\_id: [242, 311, 106, 739, 406, 491, 159, 93, 69, 21, 412, 563, 297, 820, 784, 61, 228, 533, 550, 369]  d\_id: [284, 799, 177, 847, 109, 269, 220, 27, 849, 657, 827, 267, 820, 279, 199, 169, 318, 297, 643, 889]   |  |  |  | | --- | --- | --- | | **编号** | **最短路径** | **最短路长度** | | **1** | **[242, 788, 790, 796, 798, 804, 806, 710, 712, 393, 716, 718, 722, 724, 728, 730, 413, 734, 736, 698, 810, 812, 818, 820, 830, 284]** | **64.37097** | | **2** | **[311, 857, 847, 839, 837, 827, 825, 638, 707, 475, 706, 704, 539, 694, 692, 682, 488, 405, 404, 403, 606, 604, 605, 584, 808, 804, 805, 799]** | **74.61931** | | **3** | **[106, 652, 645, 646, 507, 506, 505, 504, 477, 478, 479, 480, 486, 535, 487, 488, 405, 404, 675, 676, 671, 672, 722, 723, 177]** | **49.81395** | | **4** | **[739, 741, 747, 749, 758, 760, 769, 771, 585, 587, 400, 398, 403, 404, 405, 488, 682, 692, 694, 539, 704, 706, 475, 707, 638, 825, 827, 837, 839, 847]** | **73.42438** | | **5** | **[406, 682, 487, 535, 486, 480, 479, 478, 477, 504, 505, 506, 507, 646, 653, 655, 109]** | **32.93877** | | **6** | **[491, 490, 631, 636, 501, 502, 503, 477, 476, 475, 473, 472, 815, 269]** | **25.8781** | | **7** | **[159, 705, 703, 693, 479, 480, 486, 535, 438, 439, 440, 441, 596, 594, 427, 779, 777, 767, 766, 220]** | **42.39136** | | **8** | **[93, 639, 505, 634, 637, 572, 573, 27]** | **14.10242** | | **9** | **[69, 615, 622, 555, 556, 557, 490, 631, 571, 637, 644, 646, 653, 655, 663, 665, 849]** | **45.10727** | | **10** | **[21, 567, 499, 569, 573, 577, 578, 524, 647, 657]** | **17.67077** | | **11** | **[412, 411, 695, 700, 699, 701, 474, 706, 475, 707, 638, 825, 827]** | **28.55953** | | **12** | **[563, 562, 567, 566, 500, 501, 502, 503, 477, 476, 475, 473, 813, 267]** | **28.00314** | | **13** | **[297, 843, 841, 842, 837, 838, 833, 834, 829, 819, 820]** | **27.18355** | | **14** | **[820, 818, 817, 470, 822, 821, 826, 825, 279]** | **19.14413** | | **15** | **[784, 738, 742, 741, 743, 745, 199]** | **20.56472** | | **16** | **[61, 607, 602, 603, 718, 719, 715, 169]** | **17.96213** | | **17** | **[228, 774, 769, 770, 768, 584, 712, 393, 716, 717, 719, 723, 721, 727, 415, 733, 865, 864, 318]** | **61.61138** | | **18** | **[533, 532, 531, 573, 577, 578, 645, 652, 452, 451, 450, 453, 841, 843, 297]** | **28.42001** | | **19** | **[550, 560, 558, 491, 559, 566, 500, 570, 572, 576, 643]** | **18.58747** | | **20** | **[369, 915, 914, 389, 801, 913, 417, 416, 415, 733, 737, 866, 869, 872, 832, 459, 845, 467, 466, 862, 889]** | **113.28117** |   表2：cs最短路表    图2：cs网络编号1的最短路可视化   * **对于chi：**   o\_id: [3868, 4970, 1691, 11817, 6490, 7846, 2540, 1477, 1090, 325, 6580, 9002, 4742, 12537, 965, 3637, 8526, 8793, 5903, 4534]  d\_id: [12776, 2829, 1740, 4289, 3513, 421, 10498, 4265, 4453, 3170, 2701, 5077, 4746, 10273, 11992, 6102, 1421, 9927, 5529, 11006]  **\*由于chi网络过于庞大，导出的chi最短路表完整数据在附录（最短路径）**   |  |  |  |  | | --- | --- | --- | --- | | **编号** | **o** | **d** | **最短路长度** | | **1** | **3868** | **12776** | **23.48** | | **2** | **4970** | **2829** | **28.06** | | **3** | **1691** | **1740** | **56.38** | | **4** | **11817** | **4289** | **38.35** | | **5** | **6490** | **3513** | **44.05** | | **6** | **7846** | **421** | **13.26** | | **7** | **2540** | **10498** | **13.57** | | **8** | **1477** | **4265** | **61.51** | | **9** | **1090** | **4453** | **40.25** | | **10** | **325** | **3170** | **39.28** | | **11** | **6580** | **2701** | **62.04** | | **12** | **9002** | **5077** | **14.98** | | **13** | **4742** | **4746** | **0.72** | | **14** | **12537** | **10273** | **60.29** | | **15** | **965** | **11992** | **49.44** | | **16** | **3637** | **6102** | **93.61** | | **17** | **8526** | **1421** | **31.24** | | **18** | **8793** | **9927** | **19.7** | | **19** | **5903** | **5529** | **55.26** | | **20** | **4534** | **11006** | **31.71** |   表3：chi最短路表    图3：chi网络编号1的最短路可视化  4.4 时间复杂度分析  4.4.1分析逻辑  编写glc\_time\_tests()、lc\_time\_tests()、ls\_time\_tests()函数，函数的基本逻辑如下流程图（\*表示glc、lc、ls）：    图4：时间读取函数流程图  以ls\_time\_tests()为例，代码为：   1. **def** ls\_time\_tests(): 2. ls\_time = [] 3. **for** i **in** range(1,10): 4. start = time.time() #开始运行时间 5. SPP\_LS(o\_id[i], NODE) 6. end = time.time() #结束运行时间 7. ls\_time.append(end-start) 8. **print**('ls\_time:',end='') 9. **print**(ls\_time)   主程序中，对于不同网络，每次生成10对od对。如下：    4.4.2运行结果  对于cs网络：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **od** | (242, 412) | (311, 563) | (106, 297) | (739, 820) | (406, 784) | (491, 61) | (159, 228) | (93, 533) | (69, 550) | | **glc\_time(s)** | 0.7959 | 0.8435 | 0.7829 | 0.7790 | 0.7450 | 0.7131 | 0.7584 | 0.7979 | 0.8029 | | **lc\_time(s)** | 0.0050 | 0.0030 | 0.0040 | 0.0040 | 0.0030 | 0.0030 | 0.0030 | 0.0030 | 0.0040 | | **ls\_time**  **(s)** | 0.0379 | 0.0369 | 0.0399 | 0.0439 | 0.0479 | 0.0389 | 0.0419 | 0.0389 | 0.0466 |   表4：cs网络运行时间表  对于chi网络：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **od** | (3868, 6580) | (4970, 9002) | (1691, 4742) | (11817, 12537) | (6490, 965) | (7846, 3637) | (2540, 8526) | (1477, 8793) | (1090, 5903) | | **glc\_time(s)** | 214.967 | 217.525 | 206.690 | 211.358 | 205.541 | 212.362 | 215.388 | 215.893 | 219.841 | | **lc\_time(s)** | 2.076 | 7.544 | 4.244 | 2.451 | 3.871 | 6.656 | 6.778 | 5.447 | 1.907 | | **ls\_time**  **(s)** | 9.799 | 9.755 | 9.300 | 10.388 | 9.713 | 9.799 | 10.152 | 9.268 | 9.325 |   表5：chi网络运行时间表  对以上两表进行整合统计如下：   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **网络** | **node\_num** | **link\_num** | **算法** | **max\_time(s)** | **min\_time(s)** | **avg\_time(s)** | | **cs** | **933** | **2950** | **glc\_al** | **219.841** | **205.541** | **213.285** | | **lc\_al** | **7.544** | **1.907** | **4.553** | | **ls\_al** | **10.388** | **9.268** | **9.722** | | **chi** | **12982** | **39018** | **glc\_al** | **0.844** | **0.713** | **0.780** | | **lc\_al** | **0.005** | **0.003** | **0.004** | | **ls\_al** | **0.048** | **0.037** | **0.041** |   表6：综合时间统计表  以该数据做柱状图对比  图6：chi网络运行时间统计图  图7：cs网络运行时间统计图  4.4.3结果分析  利用多组随机od对对三个算法进行分析，在cs与chi网络中，无论是最长运行时间、最短运行时间还是平均运行时间，都可以看到glc的运行时间远远大于lc与ls。lc算法运行时间略短于ls。  从时间复杂度情况考虑：  扫描单个起点到图中各点的最短距离，最坏情况下：   * glc算法：   图中的节点数为n，边个数为m。  循环没有明确指出迭代次数的值，即节点与边的数量，初始化节点为n，遍历所有的点与边n\*m。  即时间复杂度为：O（n\*m）   * lc算法：   图中的节点数为n，边个数为m。假设可能存在负回路。  循环并没有明确指出迭代次数的值，假设最大弧长值为C，源节点到其他节点的路径长度上界nC，下界为-nC，若假设每次更新距离标签只减少1（最坏情况），且则至多为2nC  O（nC）   * ls算法：   图中的节点数为n，边个数为m，平均每个点的边数k = m / n ，选出一个与集合S距离最短的点v，顺序遍历。  时间复杂度=(n−1)∗(n+1+k)=n∗(n+k)=n2+m  ​即O（n2） |

1. 实验总结

|  |
| --- |
| 通过本次实验，掌握了基本的GLC、LC、LS算法的原理，丰富了对最短路算法的了解，编程实现了上述算法在大型图网络上的应用。对编译工具的使用更加熟悉，对比GLC/LC/LS三个算法的时间复杂度，对算法+数据结构的理解更加深刻。 |

2

附录：

chi最短路径输出表：

|  |  |  |
| --- | --- | --- |
| **编号** | **最短路径** | **最短路长度** |
| **1** | **[3868, 12165, 2192, 9077, 11000, 11001, 8910, 11774, 11004, 11778, 2416, 10965, 11772, 12038, 10971, 12394, 12403, 12399, 2199, 2201, 2124, 12209, 12208, 2204, 2384, 12218, 7478, 10816, 10822, 12364, 10828, 12407, 12406, 10834, 12776]** | **23.48** |
| **2** | **[4970, 11651, 1833, 10937, 2600, 7523, 10701, 2401, 12946, 12895, 12832, 10683, 12829, 10677, 10671, 12753, 10665, 12754, 10110, 10660, 12670, 10648, 7778, 10643, 8361, 9223, 10637, 2090, 8364, 2441, 4008, 10625, 8366, 7585, 6742, 7587, 7589, 7560, 4378, 7559, 7904, 7902, 7457, 7067, 7060, 7010, 7014, 7012, 7011, 7899, 2834, 2830, 6906, 6905, 2829]** | **28.06** |
| **3** | **[1691, 6243, 6228, 6227, 6246, 2532, 2531, 11920, 5961, 5960, 5959, 5958, 11986, 5956, 5955, 5954, 11921, 5970, 11899, 5969, 5990, 5989, 5988, 5987, 5985, 2534, 5984, 5983, 1998, 5130, 1997, 10322, 10498, 2037, 1986, 5028, 10497, 1978, 10496, 2361, 1972, 2717, 2402, 11642, 5615, 5613, 5612, 2001, 5632, 7318, 2002, 5640, 5652, 11643, 5681, 5693, 5698, 5711, 5364, 2009, 6502, 5365, 5366, 5379, 5392, 5406, 5403, 11944, 10231, 10229, 6762, 12020, 1740]** | **56.38** |
| **4** | **[11817, 5270, 5269, 5268, 11659, 5266, 2000, 5265, 5263, 5261, 5260, 1999, 5258, 11686, 11684, 5239, 5223, 4908, 4876, 4685, 11731, 4689, 11727, 4696, 11726, 4717, 4718, 6859, 11720, 4725, 4733, 6685, 6686, 4737, 4746, 4745, 7533, 11713, 4743, 7363, 7362, 4741, 7359, 11706, 4846, 4831, 4845, 2705, 2704, 11751, 3052, 4843, 2664, 4858, 9822, 4857, 4856, 4855, 4854, 3676, 11948, 4280, 4294, 4291, 4289]** | **38.35** |
| **5** | **[6490, 12633, 4592, 4596, 11203, 7049, 12601, 12600, 12598, 4600, 12557, 11793, 12558, 11791, 12555, 11187, 12529, 12528, 11182, 8500, 12470, 12467, 2904, 12468, 3599, 2900, 11169, 11163, 12412, 11159, 9775, 3936, 6350, 11156, 6340, 11786, 6326, 11785, 3886, 4959, 11783, 8505, 11035, 3882, 11782, 12159, 12160, 12161, 11029, 11025, 2478, 12170, 12171, 12172, 2749, 2748, 12484, 12476, 12463, 12458, 12451, 11444, 12390, 12383, 7287, 1893, 3531, 1889, 3517, 11437, 3511, 3513]** | **44.05** |
| **6** | **[7846, 7842, 7837, 7833, 7831, 7826, 7782, 7822, 7819, 7810, 2342, 7806, 7803, 7796, 10646, 7789, 10651, 12224, 7788, 10656, 10655, 10654, 12670, 10653, 421]** | **13.26** |
| **7** | **[2540, 11925, 6080, 6078, 6077, 6059, 12378, 6058, 6057, 11904, 6040, 6039, 6038, 6009, 6007, 11900, 5983, 1998, 5130, 1997, 10322, 10498]** | **13.57** |
| **8** | **[1477, 11485, 2976, 3001, 3027, 3026, 3028, 3029, 3012, 3039, 3038, 11490, 3072, 3073, 3242, 3255, 3270, 9606, 11494, 3332, 3333, 3334, 3346, 3345, 3366, 3367, 3598, 11609, 3638, 3648, 3673, 3690, 3691, 3692, 11614, 3709, 3711, 3712, 3713, 3825, 3826, 3827, 3828, 1992, 6407, 11597, 2026, 6413, 11599, 2015, 11601, 2044, 6439, 6440, 11603, 6449, 2121, 7612, 7615, 6451, 4229, 4228, 11605, 2194, 2172, 2163, 11607, 2248, 2254, 2278, 2292, 11978, 4267, 4266, 4265]** | **61.51** |
| **9** | **[1090, 11246, 7178, 11250, 7177, 8379, 11254, 11257, 8384, 11260, 8387, 7281, 11263, 3578, 11267, 11270, 6494, 7250, 8066, 9739, 8054, 8055, 11284, 11289, 7750, 7749, 7746, 7747, 7745, 7742, 7738, 7737, 7736, 7257, 7732, 11304, 7724, 11311, 9035, 10753, 10499, 9037, 10500, 9039, 9072, 10502, 9074, 9108, 10505, 2436, 8451, 8452, 9146, 8792, 9145, 3948, 10513, 12070, 3955, 10520, 3965, 12076, 8465, 8847, 8467, 12134, 12230, 6619, 10325, 12293, 12295, 12343, 10328, 5074, 5076, 5077, 9629, 5371, 5410, 5442, 8985, 7818, 7817, 8782, 5713, 5854, 6090, 6248, 6543, 6656, 6671, 4066, 4088, 7487, 4393, 4453]** | **40.25** |
| **10** | **[325, 10557, 5489, 5484, 5481, 8518, 5087, 5085, 8516, 5084, 5083, 8483, 10805, 8941, 12315, 10798, 12255, 7464, 10791, 3925, 12105, 10784, 12099, 10777, 10776, 9116, 3900, 9086, 6871, 9084, 6739, 10954, 8898, 8263, 4852, 9055, 10948, 4810, 10938, 9925, 9926, 3844, 9013, 9347, 8773, 7868, 11377, 7873, 7874, 9716, 9708, 11373, 7886, 7989, 7986, 7987, 7984, 7949, 7954, 7966, 7226, 11395, 11393, 8207, 11410, 9317, 9316, 3564, 3587, 9313, 3171, 3170]** | **39.28** |
| **11** | **[6580, 6560, 6555, 6538, 6537, 5599, 5577, 5578, 11763, 9799, 5581, 5582, 5562, 7434, 5551, 5547, 5516, 5508, 5492, 5485, 5456, 5455, 5453, 5452, 5451, 9832, 4679, 11730, 4687, 4693, 7530, 4688, 4715, 6864, 4713, 6857, 4723, 7381, 7380, 6685, 6686, 6687, 4739, 7350, 11708, 4752, 4755, 7331, 4756, 11701, 7328, 4759, 12930, 4761, 4762, 2273, 12885, 11666, 2276, 12886, 12887, 11037, 2881, 12808, 2256, 12742, 11041, 12740, 10914, 12660, 10910, 12642, 2894, 10905, 7483, 10900, 10895, 10889, 10623, 6563, 6556, 7581, 9203, 7548, 7547, 7546, 10611, 7450, 10612, 7451, 8695, 10607, 2847, 2825, 7047, 2846, 2845, 7055, 4383, 7010, 7014, 10604, 12957, 8328, 7020, 10358, 7866, 4333, 4334, 2813, 6916, 2815, 2816, 2819, 12939, 9341, 2466, 4310, 4311, 10355, 2874, 2645, 2644, 2678, 2683, 2695, 2701]** | **62.04** |
| **12** | **[9002, 2469, 7776, 11311, 9035, 10753, 10499, 9037, 10500, 9039, 9072, 10502, 9074, 9108, 10505, 2436, 8451, 8452, 9146, 8792, 9145, 3948, 10513, 12070, 3955, 10520, 3965, 12076, 8465, 8847, 8467, 12134, 12230, 6619, 10325, 12293, 12295, 12343, 10328, 5074, 5076, 5077]** | **14.98** |
| **13** | **[4742, 4743, 11713, 7533, 4745, 4746]** | **0.72** |
| **14** | **[12537, 12539, 12540, 8577, 3176, 4050, 3183, 12570, 3210, 12613, 7295, 12637, 3213, 4552, 2884, 2885, 4551, 12657, 12656, 12740, 11041, 12742, 11038, 12811, 11668, 12934, 12936, 11671, 12937, 10932, 1827, 10935, 2333, 4982, 4981, 7307, 11649, 4953, 11652, 4936, 4916, 4917, 5230, 5246, 2000, 11660, 5296, 5315, 5327, 11662, 5355, 5373, 5388, 2008, 6762, 10243, 10248, 10254, 10274, 10277, 10275, 10273]** | **60.29** |
| **15** | **[965, 7077, 12738, 11039, 12803, 12804, 2258, 4559, 12808, 11038, 12813, 10926, 12817, 12819, 2251, 12825, 10928, 12829, 10684, 2307, 10685, 2323, 10686, 12846, 12857, 5129, 2574, 10442, 2395, 10448, 12900, 12964, 5101, 12965, 12966, 5098, 12967, 2629, 12968, 5113, 5114, 10462, 12973, 10468, 1872, 1874, 10474, 1876, 1879, 10481, 2403, 1951, 1949, 1950, 2754, 2741, 2365, 1957, 10321, 2738, 1997, 5130, 1998, 5983, 11900, 6007, 6009, 6038, 6039, 6040, 11904, 6057, 11906, 6075, 6076, 6114, 6118, 6119, 6125, 6136, 6137, 11992]** | **49.44** |
| **16** | **[3637, 3634, 3635, 3659, 3660, 6521, 3661, 3672, 3673, 3690, 3691, 3692, 11614, 3709, 3711, 3712, 3713, 3825, 3826, 6409, 6410, 6412, 1995, 11598, 2007, 6421, 2042, 2043, 11541, 12650, 12651, 12659, 7123, 12692, 9759, 12706, 12708, 11545, 12723, 12827, 11546, 12820, 12828, 2499, 11216, 6378, 11220, 3450, 3435, 6383, 2516, 2509, 6397, 2519, 6403, 6404, 11802, 4617, 11788, 11790, 12555, 11791, 12558, 11793, 10003, 9993, 8503, 11200, 4575, 4010, 4570, 6753, 7300, 11124, 5069, 5070, 3747, 3832, 4496, 11131, 11132, 3885, 11127, 3208, 3206, 12615, 12616, 10890, 10884, 10891, 9205, 12640, 2151, 12639, 12717, 2165, 1960, 1961, 2167, 10042, 2168, 10044, 2175, 2035, 2036, 5052, 5058, 2177, 5065, 2729, 9639, 10480, 1878, 2631, 2632, 1944, 10486, 1948, 1950, 2754, 2741, 2365, 1957, 10321, 2738, 1997, 5130, 1998, 5983, 11900, 6007, 6009, 6038, 6039, 6040, 11904, 6057, 6058, 12378, 6059, 6077, 6078, 6080, 11925, 2540, 6094, 6095, 6101, 2688, 6102]** | **93.61** |
| **17** | **[8526, 8529, 8523, 11240, 8520, 11242, 8517, 11246, 7178, 7175, 7179, 7176, 8365, 3232, 8821, 11321, 8367, 8368, 8369, 3229, 3228, 8346, 4479, 4480, 11323, 8742, 8331, 8832, 8314, 8313, 8312, 8309, 8837, 8307, 8306, 8304, 8303, 8291, 11359, 8288, 8734, 8256, 8255, 8254, 8253, 8229, 8221, 8220, 11391, 8216, 8213, 8212, 6487, 9316, 9317, 11410, 3565, 3477, 11436, 1421]** | **31.24** |
| **18** | **[8793, 12030, 8455, 9146, 8452, 8451, 2436, 10505, 9108, 10504, 9105, 9104, 10503, 3946, 8315, 9100, 8203, 10766, 3890, 9064, 10765, 10761, 8880, 9063, 10755, 10751, 8881, 10750, 10925, 10918, 10942, 9019, 10941, 3887, 9018, 9017, 10944, 9349, 9348, 11768, 4700, 8903, 3847, 3846, 3413, 9927]** | **19.7** |
| **19** | **[5903, 6196, 6197, 6333, 6336, 2427, 6341, 6722, 6724, 6892, 6893, 6894, 10597, 12185, 7004, 10603, 7005, 7055, 2845, 2846, 7047, 2825, 2847, 10607, 8695, 7451, 10612, 7450, 10611, 7546, 12591, 7527, 12588, 4518, 10882, 2127, 2877, 12584, 2160, 10887, 10886, 12627, 12625, 12624, 10891, 9205, 12640, 2449, 2152, 12636, 2310, 2312, 2314, 2316, 9970, 12737, 12734, 10084, 12883, 4622, 4867, 11749, 4858, 9822, 4857, 4856, 4838, 4836, 4825, 5509, 6655, 9400, 6654, 11949, 6653, 5522, 11960, 9395, 5521, 5520, 11959, 5529]** | **55.26** |
| **20** | **[4534, 4473, 2779, 2760, 4437, 4435, 4427, 2764, 2765, 2766, 2767, 2768, 2769, 2678, 4283, 4151, 4138, 9340, 2459, 4345, 10352, 4305, 8756, 10347, 6386, 2426, 6377, 6374, 8828, 8819, 4408, 8798, 6206, 8796, 5930, 5921, 10573, 5896, 5884, 10566, 4430, 5794, 5528, 5518, 8642, 8641, 8640, 5158, 5119, 5105, 10545, 8483, 10805, 8941, 12315, 12314, 10804, 9623, 8890, 12252, 8260, 12251, 12250, 12249, 10794, 12244, 12241, 10982, 7558, 12238, 7522, 10975, 10970, 7109, 7107, 12085, 10923, 12035, 12034, 3872, 8290, 8430, 11006]** | **31.71** |