

Program Structures and Algorithms
Spring 2023(SEC –8)

NAME: Junlong Qiao
NUID: 002784609

Task: Assignment 4(WQUPC)

Implement height-weighted Quick Union with Path Compression and check that the unit tests for this class all work. Develop a UF client to find the relationship between the number of objects and the pairs

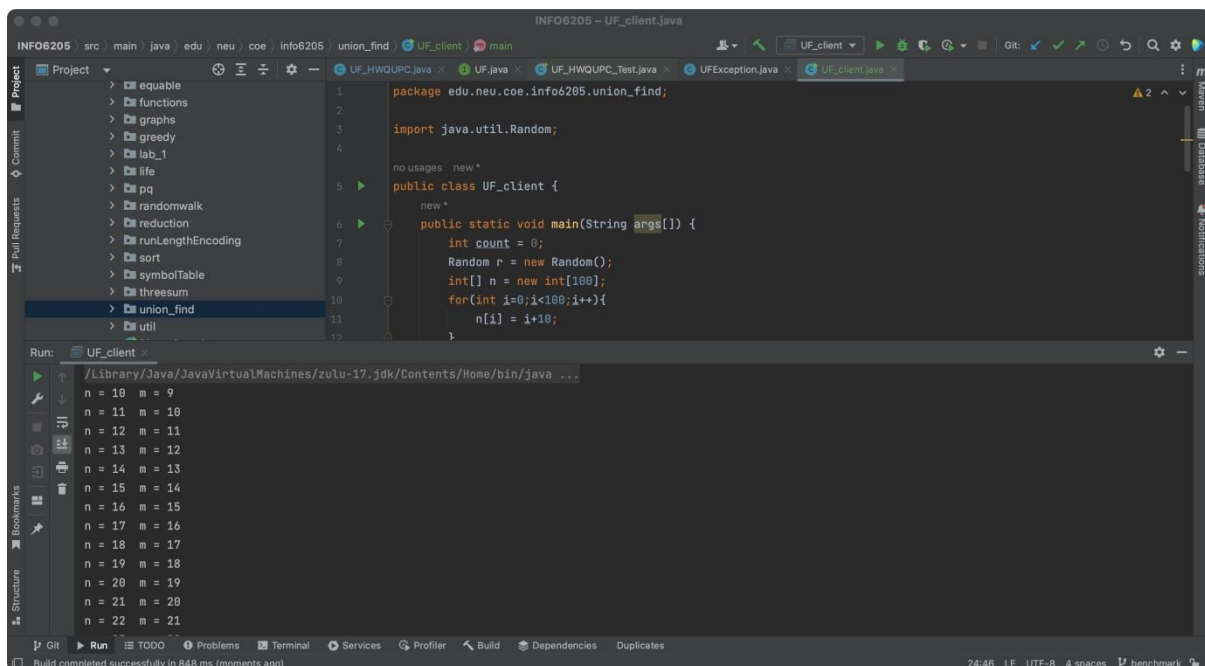
Runtie Relationship Conclusion:

At first I thought that m grew linearly with n , but as n increased, the rate of increase of m was accelerating and accelerating very slowly, not up to the quadratic power of n , so I speculated $m = k * n * \lg n$. After fitting, the relationship is obtained as follows

$$m = 0.54 * n * \lg n$$

Evidence to support that conclusion:

The average value of m is obtained by running n from 10 to 1009 each for 1000 times as follows



```
package edu.neu.coe.info6205.union_find;

import java.util.Random;

no usages new *
public class UF_client {
    new *
    public static void main(String args[]) {
        int count = 0;
        Random r = new Random();
        int[] n = new int[100];
        for(int i=0; i<100; i++){
            n[i] = i+10;
        }
    }
}
```

Run: UF_client

```
/Library/Java/JavaVirtualMachines/zulu-17.jdk/Contents/Home/bin/java ...
n = 10 m = 9
n = 11 m = 10
n = 12 m = 11
n = 13 m = 12
n = 14 m = 13
n = 15 m = 14
n = 16 m = 15
n = 17 m = 16
n = 18 m = 17
n = 19 m = 18
n = 20 m = 19
n = 21 m = 20
n = 22 m = 21
```

Build completed successfully in 848 ms (moments ago) 24:46 LF UTF-8 4 spaces benchmark

```

package edu.neu.coe.info6205.union_find;

import java.util.Random;

public class UF_client {
    public static void main(String args[]) {
        int count = 0;
        Random r = new Random();
        int[] n = new int[100];
        for(int i=0; i<100; i++){
            n[i] = i+10;
        }
    }
}

```

Run: UF_client

```

n = 96 m = 95
n = 97 m = 96
n = 98 m = 97
n = 99 m = 98
n = 100 m = 99
n = 101 m = 100
n = 102 m = 101
n = 103 m = 102
n = 104 m = 103
n = 105 m = 104
n = 106 m = 105
n = 107 m = 106
n = 108 m = 107
n = 109 m = 108

```

Build completed successfully in 848 ms (moments ago)

```

private static int count(int n){
    int count = 0;
    for(int i=0; i<1000; i++){
        Random r = new Random();
        UF h = new UF_HWQUPC(n);
        while (!h.isAllConnect(h, n)) {
            count = count + 1;
            int p = r.nextInt(n);
            int q = r.nextInt(n);
            if (!h.isConnected(p, q)) {
                h.union(p, q);
            }
        }
        int q = r.nextInt(n);
    }
}

```

Run: UF_client

```

999, 3737
1000, 3755
1001, 3781
1002, 3763
1003, 3745
1004, 3743
1005, 3766
1006, 3758
1007, 3800
1008, 3801
1009, 3791

```

Process finished with exit code 0

Build completed successfully in 905 ms (20 minutes ago)

n	m
10	16
11	18
12	20
13	22
14	24
15	27

16	29
17	30
18	33
19	35
...	...
1009	3817

Use python to fit with $k \cdot n \cdot \log n$ as the target.

```

1  import matplotlib.pyplot as plt
2  import numpy as np
3  import pandas as pd
4  from scipy import optimize as op
5  x=[]
6  for num in range(10,1010):
7      x.append(num)
8  y=[]
9  with open('f:/data.csv','r') as file:
10     while (line:=file.readline().rstrip()):
11         line1 = int(line)
12         y.append(line1)
13     print(x)
14     print(y)
15     def f(x,A):
16         return A * x*np.log(x)
17     A = op.curve_fit(f,x,y)[0]
18     print(A)
19     plt.scatter(x,y,marker='o',label='true')
20     x1 = x
21     y1 = A * x*np.log(x)
22     plt.plot(x1,y1,color='red',label='predict')
23     plt.legend()
24     plt.show()
25

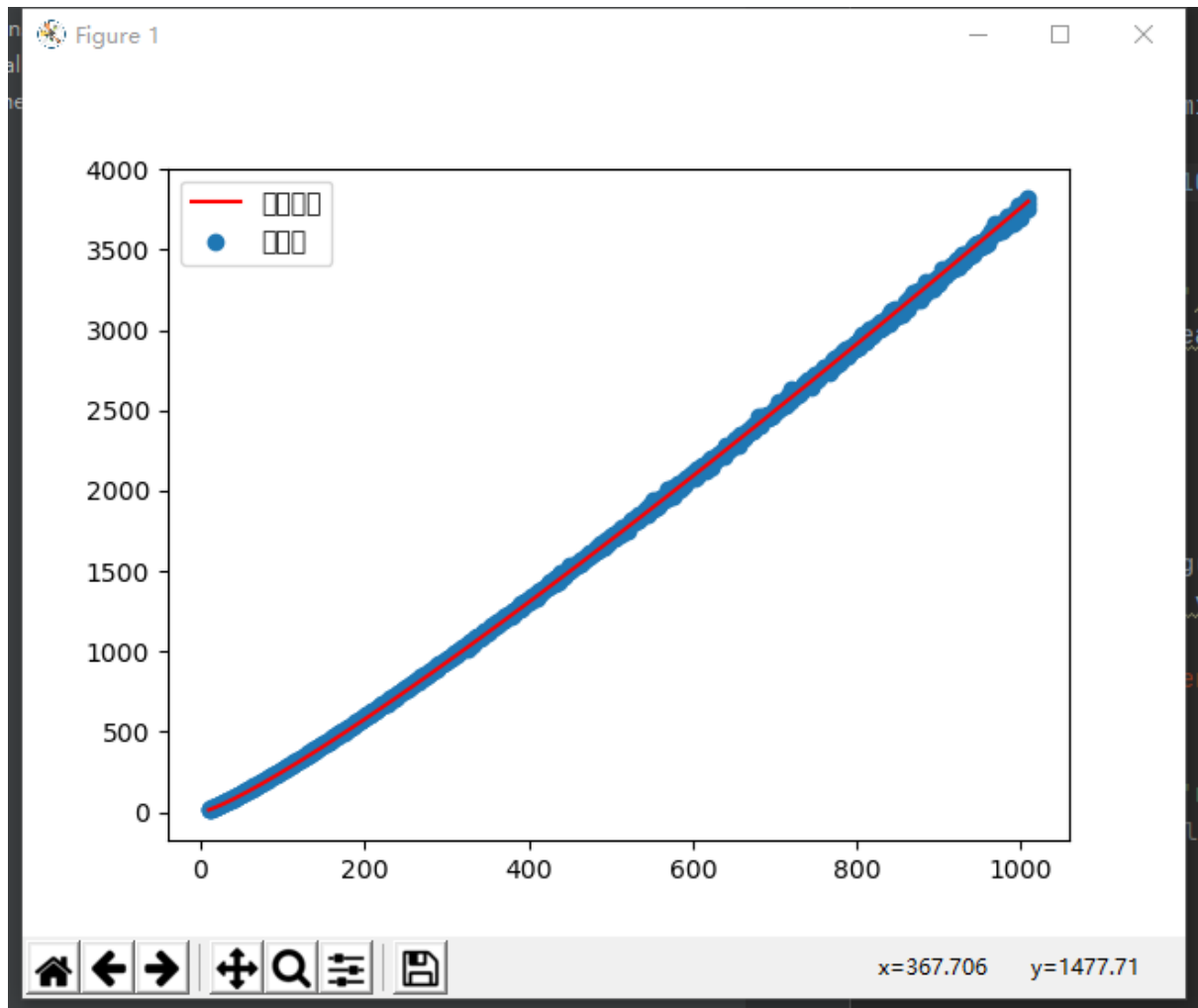
```

Run

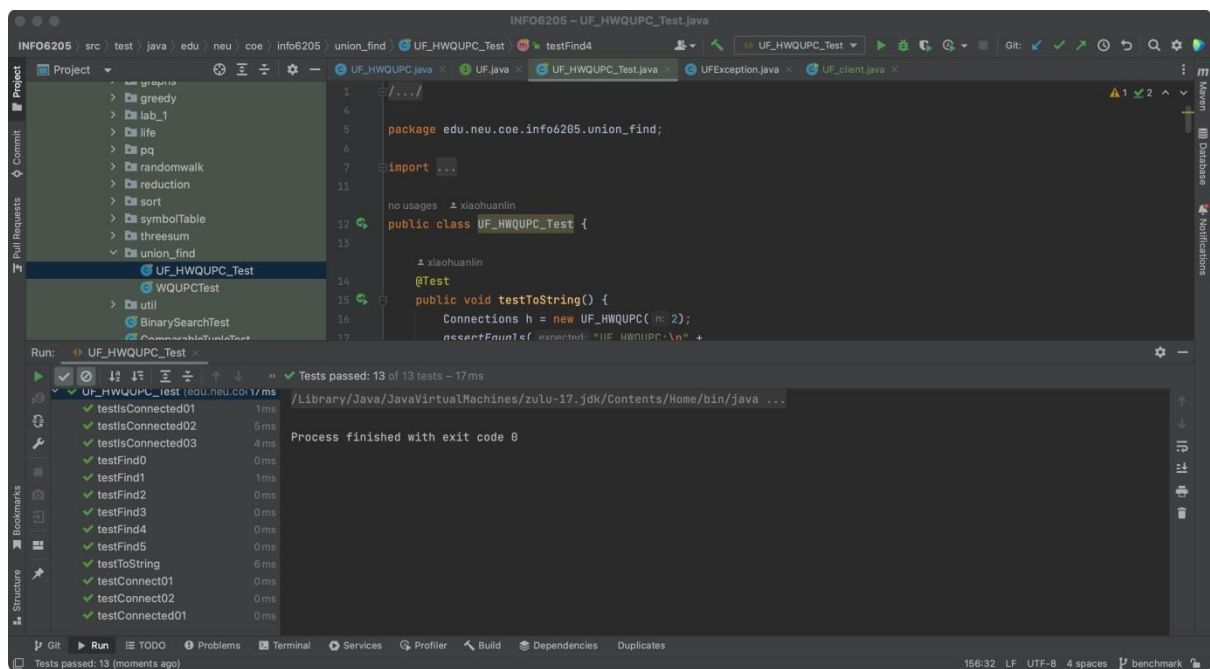
C:\ProgramData\Anaconda3\envs\plot\python.exe D:/pythonProject/main.py

[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, 590, 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, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 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, 670, 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, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 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, 770, 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, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 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, 870, 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, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 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, 970, 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, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010]

PyCharm 2020.2.5 available
Update...



Unit Test Screenshots:



Code of 3-Sum:

UF-client.java

```
package edu.neu.coe.info6205.union_find;

import java.util.Random;

public class UF_client {
    public static void main(String args[]) {
        int[] n = new int[1000];
        for(int i=0;i<1000;i++){
            n[i] = i+10;
        }
        for (int i = 0; i < n.length; i++) {
            int m = count(n[i]);
            System.out.println((i+10) + "," + m);
        }
    }

    private static int count(int n){
        int count = 0;
        for(int i=0;i<1000;i++){
            Random r = new Random();
            UF h = new UF_HWQUPC(n);
            while (!isAllConnect(h, n)) {
                count = count + 1;
                int p = r.nextInt(n);
                int q = r.nextInt(n);
                if (!h.isConnected(p, q)) {
                    h.union(p, q);
                }
            }
        }
        return count/1000;
    }

    private static boolean isAllConnect(UF h, int n){
        for(int i=1;i<n;i++){
            if(!h.isConnected(0,i)){
                return false;
            }
        }
        return true;
    }
}
```