

《数据结构与算法》作业一

姓名：余弦 学号：2021112893 班级：2103401

一、数据结构与算法的关系

大部分算法的实现需要利用数据结构。比如深度优先搜索就经常要用到栈、宽度优先搜索就需要用到队列。而数据结构则是对数据的一种抽象，用于更高效率的解决问题。数据结构和算法一起构成了可以高效执行程序。

二、简单的排序算法

1. 冒泡排序：

代码：

```
1 void bubble_sort(int num[N], int len){
2     for(int i = 0 ; i < len - 1 ; i++){
3         for(int j = i + 1 ; j < len ; j++){
4             if(num[i] < num[j]) swap(num[i], num[j]);
5         }
6     }
7 }
```

时间复杂度分析：双重循环，且每层循环次数都与n相关，故时间复杂度为 $O(n^2)$

空间复杂度分析：在进行swap时需要一个tmp变量，所以其空间复杂度为 $O(1)$

2. 选择排序

代码：

```
1 void select_sort(int num[N], int len){
2     for(int i = 0 ; i < len - 1 ; i++){
3         int min = i;
4         for(int j = i + 1; j < len ; j++){
5             if(num[j] < num[min]){
6                 min = j;
7             }
8         }
9         swap(num[i], num[min]);
10    }
11 }
```

时间复杂度分析：双重循环，且每层循环都与n相关，故时间复杂度为 $O(n^2)$

空间复杂度分析：在进行swap时需要一个tmp变量，所以其空间复杂度为 $O(1)$

3.插入排序

代码：

```
1 void insert_sort(int num[N], int len){
2     for(int i = 1; i < len; i++){
3         int val = num[i];
4         int j;
5         for(j = i - 1; j >= 0 && val < num[j]; j--){
6             num[j + 1] = num[j];
7         }
8         num[j + 1] = val;
9     }
10 }
```

时间复杂度分析：双重循环，且都与n相关，所以时间复杂度为 $O(n^2)$

空间复杂度分析：只需一个val来临时储存，所以空间复杂度为 $O(1)$

4.冒泡排序（递归）

代码：

```
1 void bubble_sort_recursion(int num[N], int len){
2     if(len == 1) return;
3     for(int i = 0 ; i < len - 1 ; i++){
4         if(num[i] > num[i + 1]) swap(num[i], num[i + 1]);
5     }
6     bubble_sort_recursion(num, len - 1); //相当于非递归实现的外层循环，只不过是
    前进行循环
7 }
```

时间复杂度分析：本质上递归的实现只是取代了原冒泡排序中外层循环的作用，所以也可以当做双重循环，时间复杂度为 $O(n^2)$

空间复杂度分析：只在swap使用额外的tmp变量，所以空间复杂度为 $O(1)$

5.选择排序（递归）

代码：

```
1 void select_sort_recursion(int num[N], int len){
2     if(len == 1) return;
3     int max = 0;
4     for(int i = 1 ; i < len ; i++){
5         if(num[i] > num[max]) max = i;
6     }
7     swap(num[max], num[len - 1]);
8     select_sort_recursion(num, len - 1);
9 }
```

时间和空间复杂度分析同非递归实现的选择排序

6.插入排序（递归）

代码：

```
1 void insert_sort_recursion(int num[N], int len){
2     if(len == 1) return;
3     insert_sort_recursion(num, len - 1); //因为是从左到右进行插入，所以是先递归后执
    行操作
4     int val = num[len - 1];
5     int j;
6     for(j = len - 2; j >= 0 && val < num[j]; j--){
7         num[j + 1] = num[j];
8     }
9     num[j + 1] = val;
10 }
```

时间和空间复杂度分析同非递归实现的选择排序

7.验证

排序效果：排序前如下

```
100000
42 8468 6335 6501 9170 5725 1479 9359 6963 4465 5706 8146 3282 6828 9962 492 2996 1943 4828 5437 2392 4605 3903 154 293
2383 7422 8717 9719 9896 5448 1727 4772 1539 1870 9913 5668 6300 7036 9895 8704 3812 1323 334 7674 4665 5142 7712 8254 6
869 5548 7645 2663 2758 38 2860 8724 9742 7530 779 2317 3036 2191 1843 289 107 9041 8943 9265 2649 7447 3806 5891 6730 4
371 5351 5007 1102 4394 3549 9630 2624 4085 9955 8757 1841 4967 7377 3932 6309 6945 2440 4627 1324 5538 1539 6119 2083 2
930 6542 4834 1116 4640 9659 2705 9931 3978 2307 1674 2387 5022 8746 6925 9073 6271 5830 6778 5574 5098 6513 3987 3291 9
162 8637 2356 4768 3656 5575 4032 2053 7351 1151 6942 1725 3967 3431 1108 192 8008 1338 5458 2288 7754 384 4946 8910 221
0 9759 4222 8589 6423 4947 7507 3031 6414 9169 901 2592 8763 1656 7411 6360 7625 538 1549 6484 7596 4042 3603 4351 292 8
37 9375 1021 4597 4022 7349 3200 9669 4485 8282 4735 54 2000 6419 7939 6901 3789 8128 468 3729 4894 4649 2484 7808 2422
4311 6618 2814 9515 4310 7617 8936 7452 601 5250 6520 1557 2799 304 6225 1009 5845 2610 4990 2703 3196 486 3094 4344 524
1588 9315 9504 7449 5201 3459 6619 581 9797 4799 5282 9590 799 8010 7158 473 3623 8539 2293 6039 4180 8191 9658 7959 61
92 9816 2889 9157 1512 6203 2635 4273 56 329 2647 6363 4887 8876 8434 9870 143 3845 1417 1882 1999 323 8652 22 5700 3558
8477 7893 4390 5076 713 2601 2511 1004 6870 7862 4689 3402 9790 5256 6424 5003 586 4183 286 7089 1427 8618 3758 9833 93
3 4170 2155 5722 7190 9977 1330 2369 8693 1426 556 3435 6550 7442 9513 146 8061 1719 3754 6140 2424 6280 5997 6688 2530
2550 7438 9867 2950 194 3196 3298 417 8287 6106 4489 6283 2456 5735 8115 1702 1317 672 5787 2264 4314 4356 1186 54 913 8
09 1833 946 4314 7757 8322 9559 3647 7983 482 4145 3197 223 7130 2162 5536 451 1174 467 2045 1660 6293 6440 7254 25 6155
9511 4746 650 3187 8314 4475 8023 2169 4019 8788 9906 7959 7392 203 3626 6478 4415 9315 5825 9335 5875 4373 160 1834 80
71 7488 8298 7519 8178 7774 2271 1764 2669 7193 3986 3103 8481 9214 7628 4803 4100 528 2626 1544 1925 1024 9973 3062 418
2 1004 7433 7506 7594 2726 3032 8493 143 7223 1287 3065 7901 9188 8361 2414 975 4271 9171 236 834 9712 5761 8897 4668 72
86 2551 141 3695 2696 1625 8020 2126 6577 1695 2659 6303 7372 2467 4679 2594 3852 5485 1019 8465 1120 3153 2801 8088 106
1 1927 9011 4758 2171 316 9577 228 2044 2759 7165 5110 7883 7087 9566 3488 9578 4475 2626 5628 5630 1929 5424 8521 6903
4963 124 4597 3738 3262 196 2526 1265 8261 6203 8117 5031 327 9012 772 6412 5548 1154 1521 9791 4925 189 1764 4941 852 8
663 3830 901 7714 8959 7579 8366 3008 1478 1201 6059 6440 2304 2761 9358 2325 6478 5109 1114 4888 9802 2851 4461 2429 29
94 7385 9406 6541 1112 8705 2836 2357 6073 9351 8824 4486 557 3217 1627 9358 8527 3358 9338 3272 3870 9362 2897 3023 961
8 113 2718 8697 1586 4042 4424 4130 4230 4566 6560 8933 2297 9856 2054 6963 3585 9735 6655 6973 1458 4370 2533 2964 2608
2484 912 1636 68 2849 4676 2939 2224 2143 3755 6512 2742 176 1460 7826 3222 7871 1627 1935 5206 1784 3851 7399 2280 270
2 2194 2735 1638 6535 5557 1994 177 5706 6963 549 5882 301 4414 6642 9856 4856 3143 1463 7612 878 425 2679 1753 8444 829
7 2674 41 9314 876 73 2819 611 1018 4933 8113 696 3170 3832 41 6489 8686 9091 9498 2590 5991 5146 9354 9315 8652 6741 20
45 1259 336 8760 1193 7606 5265 2182 8504 3830 3776 609 9293 5998 7550 9557 5562 1628 6468 9542 6130 1241 7814 9175 602
```

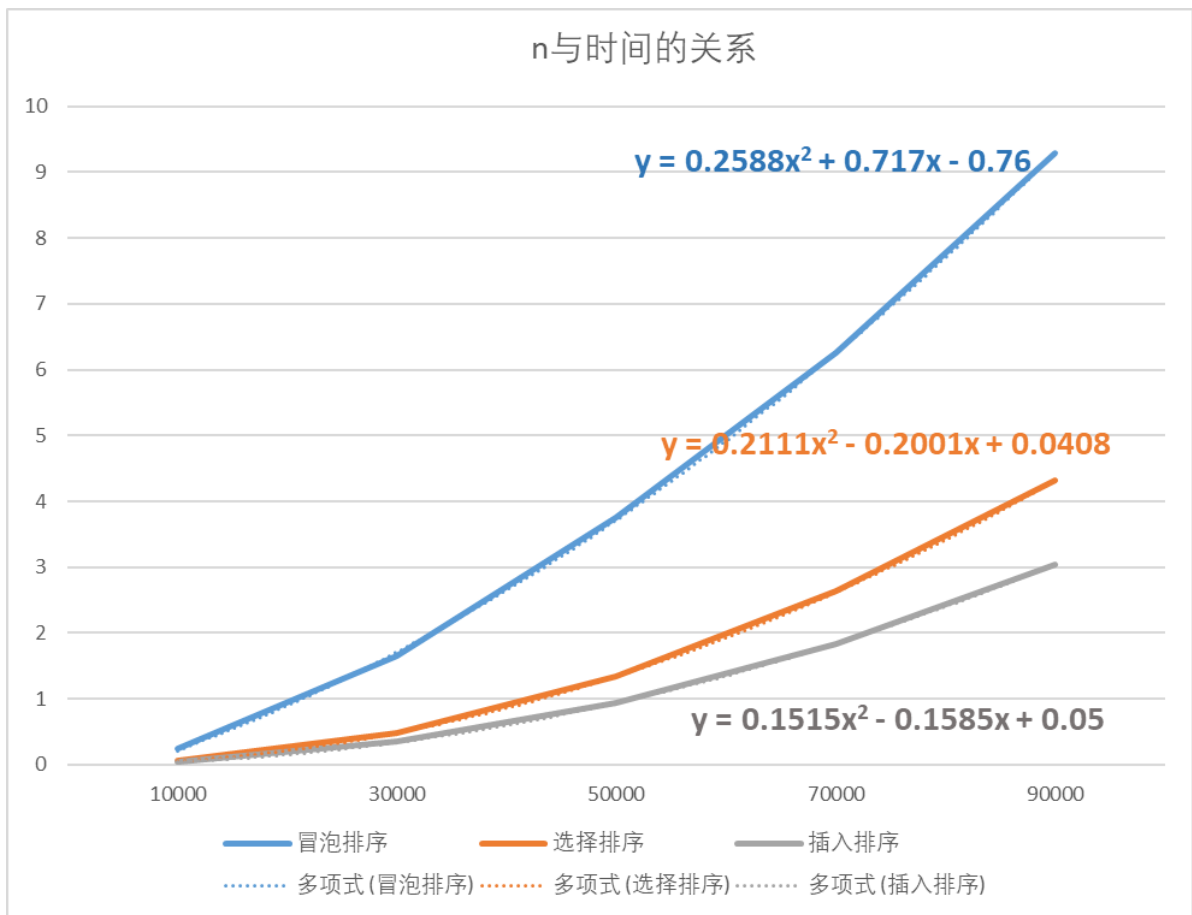
排序后如下：

1	1	3	4	4	4	5	5	7	9	9	10	13	13	13	13	15	17	17	18	21	21	22	22	23	25	25	27	28	29	29	31
32	32	32	33	33	33	34	36	36	37	37	38	38	38	39	40	40	40	41	41	41	42	42	42	43	43	44	45	46	48		
48	49	49	49	50	51	51	54	54	54	54	54	56	58	58	58	58	58	58	59	59	60	62	63	66	66	66	67	67	68	68	
68	71	71	72	73	73	73	74	74	74	75	76	76	76	77	78	79	80	81	81	81	81	81	84	84	84	84	85	85	86	89	89
90	90	91	91	91	94	94	94	96	97	99	100	102	102	102	102	102	103	103	103	104	104	104	104	105	107	107	107	107	107		
107	110	112	113	114	114	114	115	117	118	119	119	120	120	120	120	121	122	122	123	124	124	124	125	128	128	128	129	130			
130	131	131	133	133	133	133	134	135	135	135	135	136	136	137	138	138	138	138	139	140	140	140	140	141	142	143	143				
143	143	143	143	143	144	145	145	145	146	146	147	147	147	147	149	149	149	149	149	150	150	151	151	154	154	155	156	159			
160	160	160	161	163	164	166	166	168	169	169	169	169	169	170	170	170	171	171	171	171	172	174	175	176	176						
176	177	179	179	181	181	181	183	184	184	184	184	184	184	185	185	186	188	189	190	190	190	190	190	190	190	190	190	190			
191	192	192	194	194	194	194	195	196	196	196	199	200	200	200	202	203	203	203	204	205	205	206	206	206	209	210					
212	213	213	214	216	217	217	217	218	220	222	223	224	225	225	226	227	228	228	228	229	230	230	232								
232	235	236	237	237	238	238	239	239	239	242	242	243	246	246	249	251	253	254	254	254	259	259	260								
260	260	260	260	261	261	264	265	266	268	268	269	269	269	269	270	271	271	271	271	272	273	274	275								
277	279	279	280	281	283	283	284	285	285	286	286	286	288	288	289	289	290	290	291	292	293	294	294								
294	298	298	299	300	301	301	301	302	302	303	304	304	305	305	306	308	309	309	310	310	310	312	312								
313	314	314	316	317	319	319	319	320	321	321	323	324	324	324	325	327	327	329	329	331	334	334	334	334							
334	336	336	336	337	338	338	338	339	340	341	341	342	343	343	345	346	349	351	351	351	352	354	354								
354	355	356	357	358	360	362	363	364	366	367	367	370	370	371	372	372	373	374	374	378	378	379	379								
381	381	382	382	383	383	383	384	384	384	384	387	387	388	389	389	390	390	391	392	392	392	393	394								
394	394	395	396	397	397	397	398	398	399	401	402	403	404	404	404	406	407	407	407	409	412	413	416								
416	416	416	417	417	417	419	419	420	421	421	422	422	423	423	424	425	425	425	426	427	427	429	430								
431	433	433	433	433	434	436	436	438	438	440	440	441	441	442	444	445	445	445	445	448	449	449	449								
450	450	451	453	455	457	457	460	461	462	463	464	464	466	467	467	467	468	468	469	469	469	470	471								
471	471	472	472	473	473	475	475	475	475	476	476	477	477	477	479	480	480	480	482	482	482	483	483								
485	486	486	486	486	486	488	488	489	489	490	490	490	491	492	494	496	498	498	498	499	499	499	500								
503	503	503	504	504	504	505	507	507	507	509	510	510	511	512	515	519	519	520	522	523	523	524	524								
524	525	525	526	527	528	528	529	529	529	529	529	531	531	531	532	533	533	538	538	539	539	539	540								

通过使用时间函数来对算法的时间复杂度进行验证，通过使用随机数种子来使得每次数组的数据都是相同的。可以绘制出得出如下的结果。由于递归次数过多会导致栈溢出，所以50000以后的数据为空。

次数	10000	30000	50000	70000	90000
冒泡排序	0.236	1.657	3.755	6.254	9.286
选择排序	0.054	0.483	1.335	2.629	4.315
插入排序	0.041	0.344	0.935	1.839	3.046
冒泡递归	0.205	2.057	NULL	NULL	NULL
选择递归	0.053	0.485	NULL	NULL	NULL
插入递归	0.039	0.366	NULL	NULL	NULL

对数据进行拟合，可以得到公式，可以看出与次数都是成平方阶的关系。



三、汉诺塔问题

代码

```

1 void hanoi(int n, char a, char b, char c){
2     cout << "hanoi(" << n << ", " << a << ", " << b << ", " << c << ")" <<
endl;
3     if (n == 1){
4         cout << a << "-->" << c << endl;
5         count++;
6     }
7     else{
8         hanoi(n - 1, a, c, b);
9         cout << a << "-->" << c << endl;
10        count++;
11        hanoi(n - 1, b, a, c);
12    }
13 }

```

复杂度分析:

$$h(1) = 1$$

$$h(2) = 2h(1) + 1 = 2 + 1$$

$$h(3) = 2h(2) + 1 = 2^2 + 2 + 1$$

...

由数学归纳法 $h(n) = 2^{n-1} + 2^{n-2} \dots + 1 = 2^n - 1$, 所以其时间复杂度为 $O(2^n)$

无需额外的辅助空间, 所以其空间复杂度为 $O(1)$

四、角谷猜想

代码

```
1  #include "iostream"
2  #include "vector"
3  using namespace std;
4  //角谷猜想，并找出最长序列
5  vector<vector<int>> sequence;
6  vector<int> tmp;
7  void jiaogu(int n){
8      tmp.push_back(n);
9      if (n == 1){
10         sequence.push_back(tmp);
11         tmp.clear();
12     }
13     else{
14         if (n % 2 == 0){
15             jiaogu(n / 2);
16         }
17         else{
18             jiaogu(3 * n + 1);
19         }
20     }
21 }
22
23 int main(){
24     int n;
25     for(int i = 2 ; i <= 100 ; i++){
26         jiaogu(i);
27     }
28     //输出sequence中最长的序列
29     int max = 0;
30     int index = 0;
31     for(int i = 0 ; i < sequence.size() ; i++){
32         if(sequence[i].size() > max){
33             max = sequence[i].size();
34             index = i;
35         }
36     }
37     cout << "max sequence: ";
38     for(int i = 0 ; i < sequence[index].size() ; i++){
39         cout << sequence[index][i] << " ";
40     }
41     cout << endl;
42     return 0;
43 }
```

97时会出现最长序列

```
cos@LAPTOP-74D9CE9C:/mnt/c/Users/PC/Desktop/乱七八糟/数据结构/hw/hw1$ ./a.out
max sequence: 97 292 146 73 220 110 55 166 83 258 125 376 188 94 47 142 71 214 107 322 161 484 242 121 364 182 91 274 137 412 206 103 310 155 466 233 700 350 175 526 263 790 395 1
186 593 1780 890 445 1336 668 334 167 502 251 754 377 1132 566 283 850 425 1276 638 319 958 479 1438 719 2158 1079 3238 1619 4858 2429 7288 3644 1822 911 2734 1367 4102 2651 6154
3077 9232 4616 2308 1154 577 1732 866 433 1300 650 325 976 488 244 122 61 184 92 46 23 70 35 186 53 160 80 40 20 10 5 16 8 4 2 1
```

