

第 5 周上机实验报告

姓名: 思子华
学号: 2018202181
学院: 信息学院
日期: 2020. 4. 20

一、题目:

顶点覆盖问题 (the set-covering problem) 的贪心、近似算法实现。

2-近似算法的实现。时间复杂度为 $O(\sum_{S \in \mathcal{F}} |S|)$

GREEDY-SET-COVER 伪代码

GREEDY-SET-COVER(X, \mathcal{F})

```
1   $U = X$ 
2   $\mathcal{C} = \emptyset$ 
3  while  $U \neq \emptyset$ 
4      select an  $S \in \mathcal{F}$  that maximizes  $|S \cap U|$ 
5       $U = U - S$ 
6       $\mathcal{C} = \mathcal{C} \cup \{S\}$ 
7  return  $\mathcal{C}$ 
```

二、算法思路：

具体描述算法的设计思想：

为了达到目标的时间复杂度，需要尽量降低 while 循环中各个操作的时间复杂度。采用以空间换时间的思路，除了用数组保存各个集合的元素以外，还维护了一些数据结构：

1. 布尔数组 `vector<bool>covered`：记录结点是否已经被覆盖
2. Int 类型二维数组 `vector<vector<int>>node_set`：按照结点记录其对应的所有集合
3. 链表数组 `vector<list<int>>L`：根据 $|S \cap U|$ 的大小，将集合穿起来，L 的 size 是初识图中最大子集的大小
4. Int `L_max` 保存当前 L 中集合的最大 size
5. `Vector<int>set_length` 记录每一个集合中 $|S \cap U|$ 的元素的数量

对应伪代码

- U 用 `vector<bool>covered` 实现， $U - S$ 是将 $|S \cap U|$ 中的元素均标为 covered，显然这是 $O(|S_i|)$ 的。
- Select S 选取 L 中记录的最大 size 的集合中的第一个，即选取 `L[L_max].front()`。在 `L_max` 维护好的情况下，这个操作是 $O(1)$
- $C = C \cup \{S\}$ 用一个 `vector<int>` 储存最终结果的子集的集合的序号，`res.push_back(L[L_max].front())` 是 $O(1)$

于是，实现该算法的关键是如何维护 $|S \cap U|$ 和 L 以及 `L_max`。

维护策略：

- 对于选出的集合 S_i ，遍历其中每一个元素，单独进行维护
 - 对于 S_i 的任意元素 m，使用 `node_set` 访问每一个包含 m 的集合
 - 将每一个包含 m 的集合 S_j 的大小减 1
 - ◆ `Set_length[j]` 的值减 1

- ◆ 将 L 中存储的 S_j (位置记作 $L[t]$), 移动到 $L[t-1]$
- 最后更新 L_{\max}
 - 若 L 中原 L_{\max} 处有元素, 则不变
 - 否则, 将 L_{\max} 减 1, 直到 L_{\max} 处有元素

算法时间复杂度分析

根据上述内容可知, 伪代码中对应部分的实现是 $O(\sum_{S \in \mathcal{F}} |S|)$ 。

下面证明“维护”部分是 $O(\sum_{S \in \mathcal{F}} |S|)$ 。

- 由于每一次维护部分作用了 $|S \cap U|$ 个点, 只需证明每一个点的操作都是 $O(1)$, 便可以证明“维护”部分是 $O(\sum_{S \in \mathcal{F}} |S|)$ 。
- 单独考虑一个点, 一个点被常数个集合覆盖, 根据这个点去修改常数个集合的 size, 并且调整这些集合在 L 中的位置, 这是 $O(1)$ 的。

三、程序设计框架：

核心函数的名称和功能

Greedy_set_cover()

满足伪代码架构的实现，update 为“维护”数据结构部分的实现。

```

1. vector<int> greedy_set_cover()
2. {
3.     vector<int> res;
4.     while(n > 0)
5.     {
6.         n = n - L_max;
7.
8.         res.push_back(L[L_max].front());
9.
10.        update(L_max);
11.    }
12.
13.    return res;
14. }
```

Update

遍历选中的 S_i 集合，对于每一个元素更新包含它的集合的大小以及该集合在 L 中的位置。最后，更新 L_max（当前所有集合中最大的 size）

```

1. void update(int set_size)
2. {
3.     int index = L[set_size].front();
4.     L[set_size].pop_front();
5.
6.     for(auto vertex : S[index])
7.     {
8.         decrease(vertex);
9.     }
10. }
```

```

11.     while(L_max>=0&&L[L_max].empty())
12.     {
13.         L_max--;
14.     }
15. }

```

Decrease()

利用 node_set, 遍历该包含结点的所有集合, 将该集合的 size 减 1, 再移动它在 L 中的位置。

```

1. void decrease(int vertex)
2. {
3.     if(covered[vertex]) return;
4.
5.     covered[vertex] = true;
6.
7.     for(auto S_i : node_set[vertex])
8.     {
9.         for(list<int>::iterator l = L[ set_length[S_i] ].begin(); l != L[ set
           _length[S_i] ].end(); l++)
10.        {
11.            if(*l == S_i)
12.            {
13.                L[ set_length[S_i] ].erase(l);
14.                break;
15.            }
16.        }
17.        set_length[S_i] --;
18.        L[ set_length[S_i] ].push_back(S_i);
19.    }
20. }

```

各核心类、各核心函数之间的关系

-----> 选中 S_i
greedy_set_cover -----> 更新 解集
-----> 维护数据结构 update -----> 包含结点的每一个集合 decrease

输入输出的格式等（输入文件请写为相对路径）

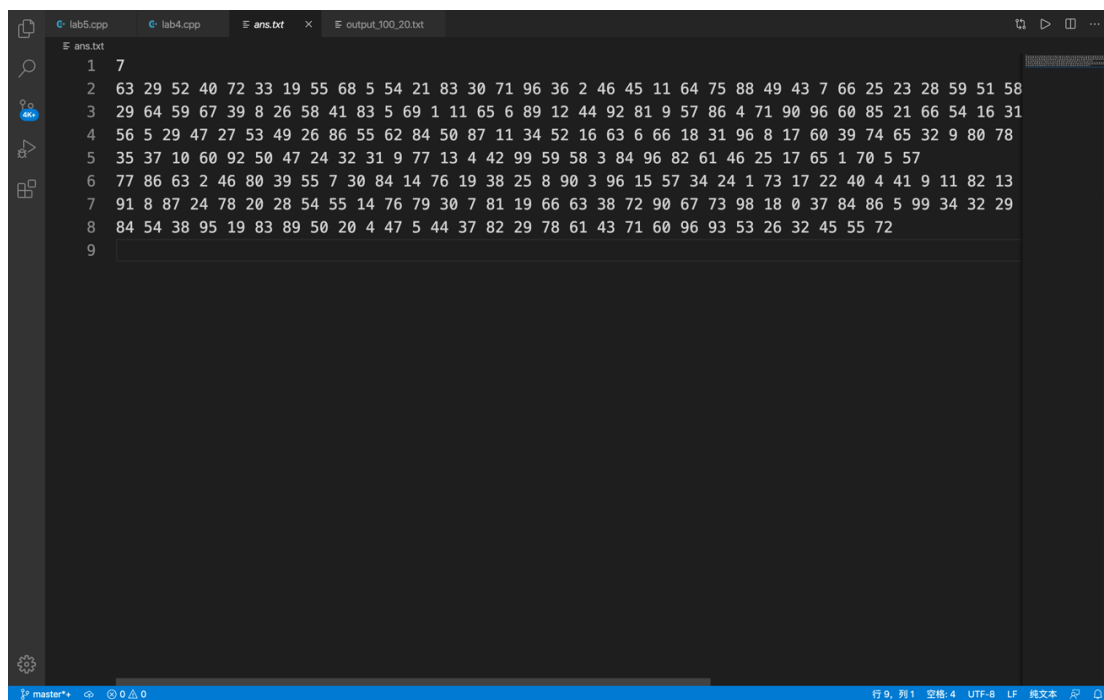
程序使用 fstream 中的 open()函数打开文件，故输入格式为相对、绝对均可。

四、实验结果说明：

(1) 实验结果截图

```
sizihua@MacBookPro:Week5_instructions/Data_Week4 <master*>$ g++ -std=c++11 lab5.cpp -o a.out
sizihua@MacBookPro:Week5_instructions/Data_Week4 <master*>$ ./a.out
please type the file path, absolute path and relative path are both OK.
setdata_100_20.txt
Compute dataset setdata_100_20.txt costs 0.498 ms
sizihua@MacBookPro:Week5_instructions/Data_Week4 <master*>$ ./a.out
please type the file path, absolute path and relative path are both OK.
setdata_1000_200.txt
Compute dataset setdata_1000_200.txt costs 4.541 ms
sizihua@MacBookPro:Week5_instructions/Data_Week4 <master*>$ ./a.out
please type the file path, absolute path and relative path are both OK.
setdata_10000_1000.txt
Compute dataset setdata_10000_1000.txt costs 72.875 ms
sizihua@MacBookPro:Week5_instructions/Data_Week4 <master*>$
```

图表 a: 运行程序



```
1 7
2 63 29 52 40 72 33 19 55 68 5 54 21 83 30 71 96 36 2 46 45 11 64 75 88 49 43 7 66 25 23 28 59 51 58
3 29 64 59 67 39 8 26 58 41 83 5 69 1 11 65 6 89 12 44 92 81 9 57 86 4 71 90 96 60 85 21 66 54 16 31
4 56 5 29 47 27 53 49 26 86 55 62 84 50 87 11 34 52 16 63 6 66 18 31 96 8 17 60 39 74 65 32 9 80 78
5 35 37 10 60 92 50 47 24 32 31 9 77 13 4 42 99 59 58 3 84 96 82 61 46 25 17 65 1 70 5 57
6 77 86 63 2 46 80 39 55 7 30 84 14 76 19 38 25 8 90 3 96 15 57 34 24 1 73 17 22 40 4 41 9 11 82 13
7 91 8 87 24 78 20 28 54 55 14 76 79 30 7 81 19 66 63 38 72 90 67 73 98 18 0 37 84 86 5 99 34 32 29
8 84 54 38 95 19 83 89 50 20 4 47 5 44 37 82 29 78 61 43 71 60 96 93 53 26 32 45 55 72
9
```

图表 b: 运行结果 1

```

1 42
2 54 145 783 958 594 51 732 487 512 375 993 498 350 749 595 745 91 385 132 62 133 494 579 493 994 98
3 555 59 675 43 947 351 619 891 33 459 266 17 197 954 639 612 364 912 272 325 81 894 322 304 737 758
4 709 209 741 116 730 491 127 918 318 957 331 277 212 856 592 801 564 414 216 823 905 984 91 843 97
5 542 400 799 551 590 903 726 49 221 142 258 758 524 981 584 509 0 723 814 479 514 344 254 941 725 8
6 241 82 796 33 637 359 838 267 947 252 335 972 210 181 327 416 460 733 677 351 939 928 4 897 973 3
7 582 82 915 439 373 820 741 279 670 896 559 165 148 495 736 398 288 792 592 223 920 405 392 644 642
8 592 506 630 798 658 722 832 880 468 979 44 994 404 450 452 457 432 110 228 600 623 157 444 879 733
9 205 245 502 713 18 663 637 38 352 348 310 295 321 823 671 863 532 299 796 8 940 180 55 439 483 324
10 474 548 185 804 750 19 155 880 248 725 176 253 797 639 830 272 696 362 190 129 876 894 812 450 265
11 976 133 568 186 287 281 607 571 586 774 60 337 567 232 110 128 869 727 854 832 634 818 256 407 48
12 854 141 993 927 297 104 219 574 886 783 761 668 272 307 809 959 499 518 793 849 150 686 786 409 54
13 76 182 632 424 598 393 457 891 975 288 18 211 382 538 911 112 890 334 356 747 126 421 823 791 543
14 485 411 202 761 341 628 898 121 529 935 707 68 358 984 444 863 691 880 695 482 422 670 344 756 113
15 318 860 596 378 972 279 610 167 120 479 322 789 202 497 573 79 662 226 770 720 408 913 903 726 15
16 647 439 699 290 6 408 917 893 469 962 183 10 485 447 41 227 337 892 652 454 103 483 25 433 778 238
17 351 690 375 681 373 547 881 736 3 394 649 758 138 4 693 517 806 180 524 773 879 548 692 629 410 22
18 130 430 276 870 507 266 446 39 147 800 849 561 442 520 690 402 949 539 808 578 518 894 597 918 45
19 364 388 920 744 235 463 969 611 730 605 191 297 52 905 107 171 532 927 429 505 124 217 417 19 24 7
20 101 553 128 921 986 62 287 983 121 936 33 677 26 648 873 209 952 547 510 199 399 950 454 765 866 8
21 714 560 773 793 358 186 159 970 961 451 612 287 418 350 787 363 725 767 280 611 158 65 229 320 235
22 558 312 167 499 926 365 547 9 769 818 306 552 580 1 749 613 318 513 750 751 349 854 976 990 916 26
23 307 927 846 163 894 335 501 573 471 869 233 151 912 412 117 711 453 712 132 957 905 951 225 115 52
24 293 974 875 765 179 342 877 220 40 693 483 87 287 918 132 394 457 731 536 711 601 418 579 163 404
25 748 5 955 847 601 988 250 804 851 545 394 393 421 687 945 281 284 165 311 288 56 156 724 749 305 2
26 605 44 733 795 976 963 109 353 890 293 788 40 592 938 112 369 857 358 837 583 325 922 618 764 273
27 532 737 630 217 935 690 489 739 728 104 374 838 303 1 823 47 733 687 588 537 150 549 84 825 53 516
  
```

图表 c: 运行结果 2

```

1 209
2 8632 1205 8765 5389 7383 8283 2020 9796 484 2712 4217 8280 9320 3450 8905 7949 5673 9357 4305 506
3 4128 7519 1701 6741 7918 173 5646 6455 9805 1352 9734 1018 7179 6422 6541 552 251 7809 8340 9059 2
4 5815 1167 4044 9931 6927 3839 3718 9593 4158 3022 1302 7604 1982 7306 5183 6688 9778 9143 3006 429
5 1250 859 7906 5653 3125 9595 1155 751 6432 3012 8246 4197 2881 2490 7443 3317 7552 9392 6687 4166
6 8842 3415 1707 4365 8566 1976 8706 3326 9475 3429 6092 7753 5772 5803 6649 3850 2124 9341 1523 377
7 6133 5051 2737 5824 1411 318 3640 6503 2388 1224 9792 8829 1991 4996 5750 7763 9124 9663 7434 9648
8 1728 5354 9284 8121 8214 6902 5943 4190 6798 3169 8458 3956 7140 4869 5957 4151 2942 5838 2619 835
9 4323 6844 47 6515 3935 898 127 1662 5140 4477 9185 6131 4428 7542 423 7050 5268 9841 219 8972 6424
10 9173 8685 4872 3003 7844 6125 7305 3417 2003 6276 7084 6512 4286 7537 7729 541 168 5786 6444 8261
11 6225 9615 3702 2386 8126 5637 3505 9697 893 2183 7168 492 8598 446 2804 3740 119 5574 9027 3151 42
12 3443 8480 5313 7994 9073 5422 5026 4558 9949 9989 2895 9095 8943 224 1456 4359 3722 5957 6593 7441
13 3212 8640 4091 3967 206 9052 840 3408 5863 52 9570 5375 951 9645 6543 8320 6100 1605 771 5516 2397
14 1488 7967 4976 2018 9821 1001 3335 4886 2036 7898 4622 8284 3366 545 8255 6596 4661 383 3224 1989
15 9275 6897 7973 851 1585 3692 2978 1948 2898 3938 3329 9184 5592 4190 1389 359 3389 3059 8705 1208
16 4475 1287 89 1823 4033 3904 9538 2184 1849 6066 8162 4540 7478 8324 3039 7149 9248 4927 3337 4420
17 8960 6867 1826 1985 4433 283 2909 7059 77 2161 9949 1444 9698 9962 9903 7919 7128 6343 3875 4357 4
18 618 7424 9893 1276 3150 5811 2674 4692 1160 4580 8187 8060 3656 2782 2962 5345 810 1846 8861 7137
19 3472 5968 9425 4991 3932 2427 166 8945 8015 3343 9902 4966 8084 9454 5842 6829 1401 2257 6281 2374
20 4822 874 7375 6266 6710 281 8759 3505 4011 9642 9698 36 6677 774 4399 1966 3172 9102 3986 6608 280
21 1233 4544 3568 6510 155 4305 7421 3453 5982 523 974 4044 335 2995 9401 3498 9957 4142 2142 7787 93
22 6288 8669 881 7866 6825 2888 8453 9479 5127 5607 9952 4989 6937 412 8053 7164 2634 1783 6342 846 3
23 573 3319 3446 7824 7338 4622 4723 2749 9037 5128 1687 5648 4913 2894 2750 5961 5295 4059 9918 424
24 259 7054 9199 8498 2779 5752 7869 6154 7983 304 425 9577 8023 258 7619 4835 9171 1728 8765 9804 76
25 8962 1328 9499 6571 369 2349 8505 2917 4637 1544 4056 2991 2190 2211 7403 3800 6483 1788 5224 7109
26 5049 7363 7538 8255 5918 5610 6525 3984 7695 9972 2980 693 5047 3518 1642 3457 192 9761 738 9483 9
27 3145 5129 8993 8317 6814 2432 9819 9006 8258 3817 7717 3808 5891 862 9939 7815 4795 1554 9711 6980
  
```

图表 d: 运行结果 3

(2) 实验结果的分析

从运行时间可以看出, 随着输入的 `setdata` 以 10 倍的比例增大, 运行时间也是以 10 倍左右的速率增大, 验证了算法的运行时间复杂度是 $O(\sum_{S \in \mathcal{F}} |S|)$

五、个人总结（选填）

本次实验的实现，如果使用 `stl` 中的 `set`，通过求交集等方式来实现，是很容易写代码的，诚然，这种实现空间复杂度也比较低，但是 `set` 的各种操作的时间复杂度最终算下来是无法达到要求的。

最终采取的方法的空间复杂度较大，大概是 $O(nm)$ 的（ n 代表结点数目， m 代表集合数目）。这体现出算法很难同时在时间和空间复杂度上都表现良好，本实验便是用空间换时间。