

58. Length of Last Word

April 13, 2020 | 2K views

Given a string s consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word (last word means the last appearing word if we loop from left to right) in the string.

If the last word does not exist, return 0.

Note: A word is defined as a **maximal substring** consisting of non-space characters only.

Example:

Input: "Hello World"
Output: 5

Solution

Approach 1: String Index Manipulation

Intuition

There is no doubt that this is an easy problem. Yet, it could be a good exercise for one to practice string manipulation, which is definitely common during interviews.

In this article, we start with some approaches that manipulate string indexes, then we look at how to use the built-in string functions to solve the problem.

The intuition is simple, as it pretty much given away from the name of the problem, i.e. first we **locate** the last word, then we **count** the length of the last word.

One should pay attention to some edge cases though:

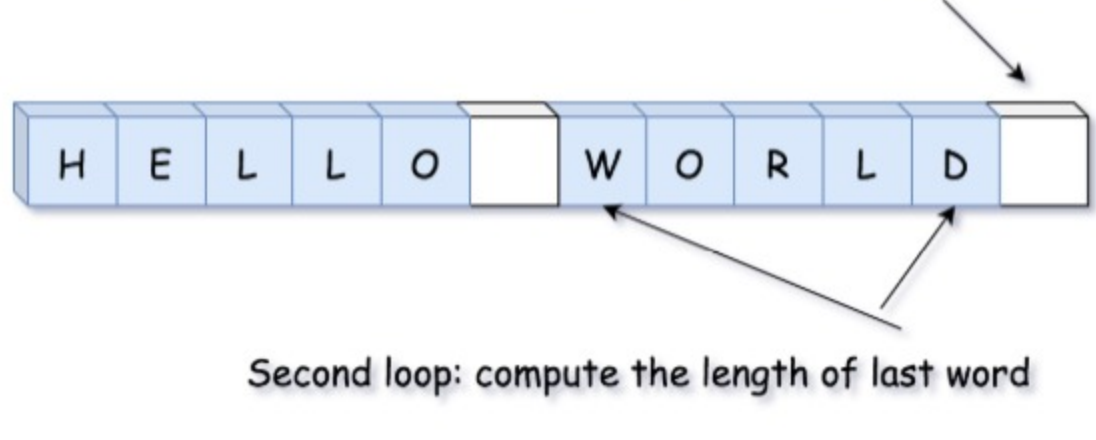
- The input string could be empty.
- There could be some trailing spaces in the input string, e.g. `hello <space>`.
- There might only be one word in the given string.

The challenge is to build a *concise* yet *comprehensive* solution that could handle all above cases.

Algorithm

One can break down the solution into two steps:

- First, we would try to locate the last word, starting from the end of the string. We iterate the string in reverse order, consuming the empty spaces. When we first come across a non-space character, we know that we are at the last character of the last word.
- Second, once we locate the last word. We count its length, starting from its last character. Again, we could use a loop here.



Here is what it looks like, a solution with **two loops**:

JavaPython3Copy

```
1 class Solution:
2     def lengthOfLastWord(self, s: str) -> int:
3         # trim the trailing spaces
4         p = len(s) - 1
5         while p >= 0 and s[p] == ' ':
6             p -= 1
7
8         # compute the length of last word
9         length = 0
10        while p >= 0 and s[p] != ' ':
11            p -= 1
12            length += 1
13        return length
```

Complexity

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

In the worst case, the input string might contain only a single word, which implies that we would need to iterate through the entire string to obtain the result.

- Space Complexity: $\mathcal{O}(1)$, only constant memory is consumed, regardless the input.

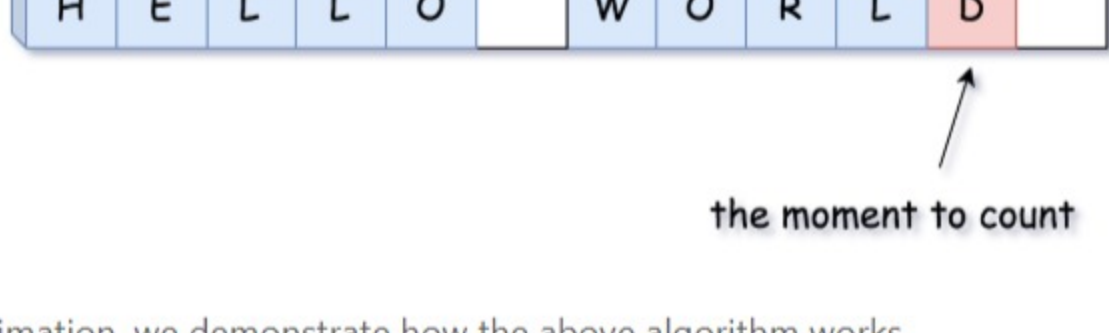
Approach 2: One-loop Iteration

Intuition

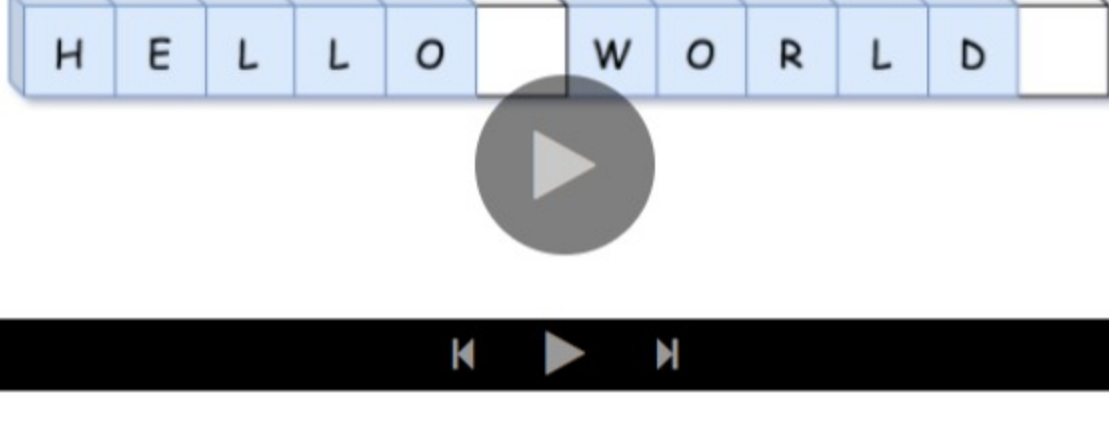
In the above approach, we applied two loops. One is used to locate the last word, and the other one to calculate its length.

We could actually complete the same tasks within a single loop.

The trick is that we could define a condition, i.e. the precise moment that we should start to count the length of the word.



In the following animation, we demonstrate how the above algorithm works.



Algorithm

Here are some sample implementations with comments.

JavaPython3Copy

```
1 class Solution:
2     def lengthOfLastWord(self, s: str) -> int:
3         p, length = len(s), 0
4
5         while p > 0:
6             p -= 1
7             # we're in the middle of the last word
8             if s[p] != ' ':
9                 length += 1
10            # here is the end of last word
11            elif length > 0:
12                return length
13
14        return length
```

Complexity

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

This approach has the same time complexity as the previous approach. The only difference is that we combined two loops into one.

- Space Complexity: $\mathcal{O}(1)$, again a constant memory is consumed, regardless the input.

Approach 3: Built-in String Functions

Intuition

As we mentioned at the beginning of the article, we could also resort to the **built-in** functions of the String data structure, in order to solve the problem.

In fact, String is such an important data type in many programming languages, that often it comes with a rich set of *built-in* functions that can help one to accomplish many common tasks, such as trimming the empty spaces in the string etc.

It would be really helpful to get proficient with those built-in functions.

Algorithm

In different programming languages, the sets of built-in functions associated with String are different though. In this section, we showcase some examples.

In Python, we would use the following built-in functions to accomplish the tasks:

- `str.isspace()`: this function determines if the `str` contains only spaces.
- `str.split(delimiter)`: this function could split the input string into several substrings, based on the given *delimiter* (by default, the delimiter is space).

One can find the list of built-in functions in Python from the [official documentation](#).

In Java, here is a list of built-in functions that we would use:

- `String.trim()`: this function returns a copy of the string, with the leading and trailing whitespaces trimmed.
- `String.length()`: this function returns the length of the string.
- `String.lastIndexOf(char)`: this function returns the index of the last occurrence of the given character.

Again, one can find more details about the APIs for the String data structure in Java, from the [official documentation](#).

JavaPython3Copy

```
1 class Solution:
2     def lengthOfLastWord(self, s: str) -> int:
3         return 0 if not s or s.isspace() else len(s.split()[-1])
```

Complexity Analysis

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

Since we use some built-in function from the String data type, we should look into the complexity of each built-in function that we used, in order to obtain the overall time complexity of our algorithm.

It would be safe to assume the time complexity of the methods such as `str.split()` and `String.lastIndexOf()` to be $\mathcal{O}(N)$, since in the worst case we would need to scan the entire string for both methods.

- Space Complexity: $\mathcal{O}(N)$. Again, we should look into the built-in functions that we used in the algorithm.

In the Java implementation, we used the function `String.trim()` which returns a **copy** of the input string without leading and trailing whitespaces. Therefore, we would need $\mathcal{O}(N)$ space for our algorithm to hold this copy.


In the Python implementation, we used `str.split()`, which returns a list of **substrings** that are separated by the space delimiter. As a result, we would need $\mathcal{O}(N)$ space for our algorithm to store this list.

Rate this article: ★★★★★


PreviousNext

Comments: 8

Sort By

- 

Type comment here... (Markdown is supported)

PreviewPost
- 

londhehimanshu ★0 2 days ago

Python3 solution (36 ms), O(n)

```
def lengthOfLastWord(self, s: str) -> int:
    s = s.strip()
    Read More
```

0 1 2 3 4 5 6 7 8 9 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

0 1 2 3 4 5 6 7 8 9 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

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31