

//split linked list into parts

Problem List < > < >

Description Editorial Solutions Submissions

725. Split Linked List in Parts

Solved

Medium Topics Companies Hint

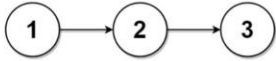
Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the k parts.

Example 1:



Input: head = [1,2,3], k = 5
Output: [[1], [2], [3], [], []]
Explanation:
The first element output[0] has output[0].val =

3.8K 49

</> Code

C Auto

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     struct ListNode *next;
6  * };
7  */
8 /**
9  * Note: The returned array must be malloced, assume caller calls free().
10 */
11 struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize) {
12
13     *returnSize = k;
14     struct ListNode** arr = (struct ListNode**)malloc(k*sizeof(struct ListNode));
15     for (int i=0; i<k; i++){
16         arr[i] = (struct ListNode*)malloc(sizeof(struct ListNode));
17     }
18     int i=0, size=0, c;
19     struct ListNode* curr=head;
20     struct ListNode* temp;
21     while (curr){
22         size++;
23         curr = curr->next;
24     }
25     while (k && size){
26         c = size/k;
27         if (size % k != 0){
28             c++;
29         }
30         curr = head;
31         temp = head;
32         for (int j=1; j<=c; j++){
```

Saved to local

Ln 1, Col 1

Testcase Test Result

Problem List < > < >

Description Editorial Solutions Submissions

725. Split Linked List in Parts

Solved

Medium Topics Companies Hint

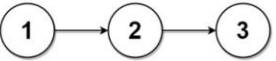
Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the k parts.

Example 1:



Input: head = [1,2,3], k = 5
Output: [[1], [2], [3], [], []]
Explanation:
The first element output[0] has output[0].val =

3.8K 49

</> Code

C Auto

```
23     curr = curr->next;
24 }
25 while (k && size){
26     c = size/k;
27     if (size % k != 0){
28         c++;
29     }
30     curr = head;
31     temp = head;
32     for (int j=1; j<=c; j++){
33         temp = temp->next;
34     }
35     head = temp->next;
36     temp->next = NULL;
37     arr[i] = curr;
38     i++;
39     k--;
40     size -= c;
41 }
42 if (k){
43     for (int j=0; j<k; j++){
44         arr[j] = NULL;
45         j++;
46     }
47 }
48
49 return arr;
50 }
51
52
53
```

Saved to local

Ln 1, Col 1

Testcase Test Result

Output

Problem List

725. Split Linked List in Parts

Solved

Medium

Topics

Companies

Hint

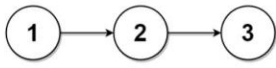
Given the head of a singly linked list and an integer k , split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the k parts.

Example 1:



Input: head = [1,2,3], $k = 5$
Output: [[1], [2], [3], [], []]
Explanation:
The first element output[0] has output[0].val = 1

3.8K 49

Code

```
23 curr = curr->next;
24 }
25 while (k && size){
26     c = size/k;
```

Saved to local

Ln 1, Col 1

Testcase

Test Result

Accepted

Runtime: 6 ms

Case 1

Case 2

Input

head =

[1,2,3]

k =

5

Output

[[1], [2], [3], [], []]

Expected

[[1], [2], [3], [], []]

Contribute a testcase

Problem List

725. Split Linked List in Parts

Solved

Medium

Topics

Companies

Hint

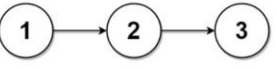
Given the head of a singly linked list and an integer k , split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the k parts.

Example 1:



Input: head = [1,2,3], $k = 5$
Output: [[1], [2], [3], [], []]
Explanation:
The first element output[0] has output[0].val = 1

3.8K 49

Code

```
23 curr = curr->next;
24 }
25 while (k && size){
26     c = size/k;
```

Saved to local

Ln 1, Col 1

Testcase

Test Result

Accepted

Runtime: 6 ms

Case 1

Case 2

Input

head =

[1,2,3,4,5,6,7,8,9,10]

k =

3

Output

[[1,2,3,4], [5,6,7], [8,9,10]]

Expected

[[1,2,3,4], [5,6,7], [8,9,10]]

Contribute a testcase

Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
```

```

struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize)
{
    *returnSize = k;
    struct ListNode** arr = (struct ListNode*)malloc(k*sizeof(struct ListNode));
    for (int i=0; i<k; i++){
        arr[i] = (struct ListNode*)malloc(sizeof(struct ListNode));
    }
    int i=0, size=0, c;
    struct ListNode* curr=head;
    struct ListNode* temp;
    while (curr){
        size++;
        curr = curr->next;
    }
    while (k && size){
        c = size/k;
        if (size % k != 0){
            c++;
        }
        curr = head;
        temp = head;
        for (int j=1; j<c; j++){
            temp = temp->next;
        }
        head = temp->next;
        temp->next = NULL;
        arr[i] = curr;
        i++;
        k--;
        size -= c;
    }
    if (k){
        for (int j=0; j<k; j++){
            arr[i] = NULL;
            i++;
        }
    }

    return arr;
}

```