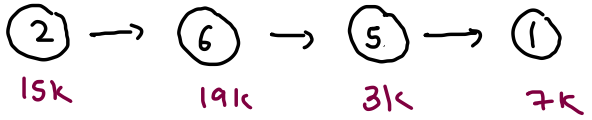
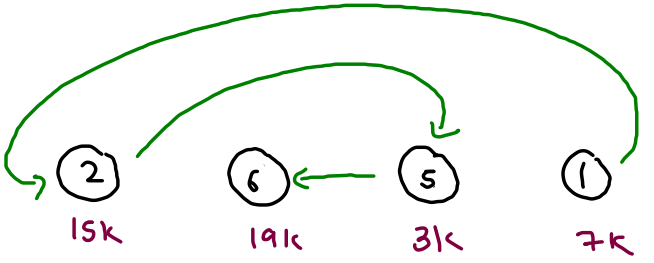
148. Sort List

Medium 6791 223 Add to List Share

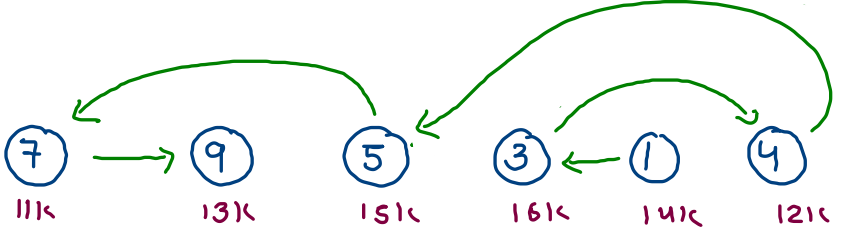
Given the `head` of a linked list, return *the list after sorting it in* ***ascending order***.

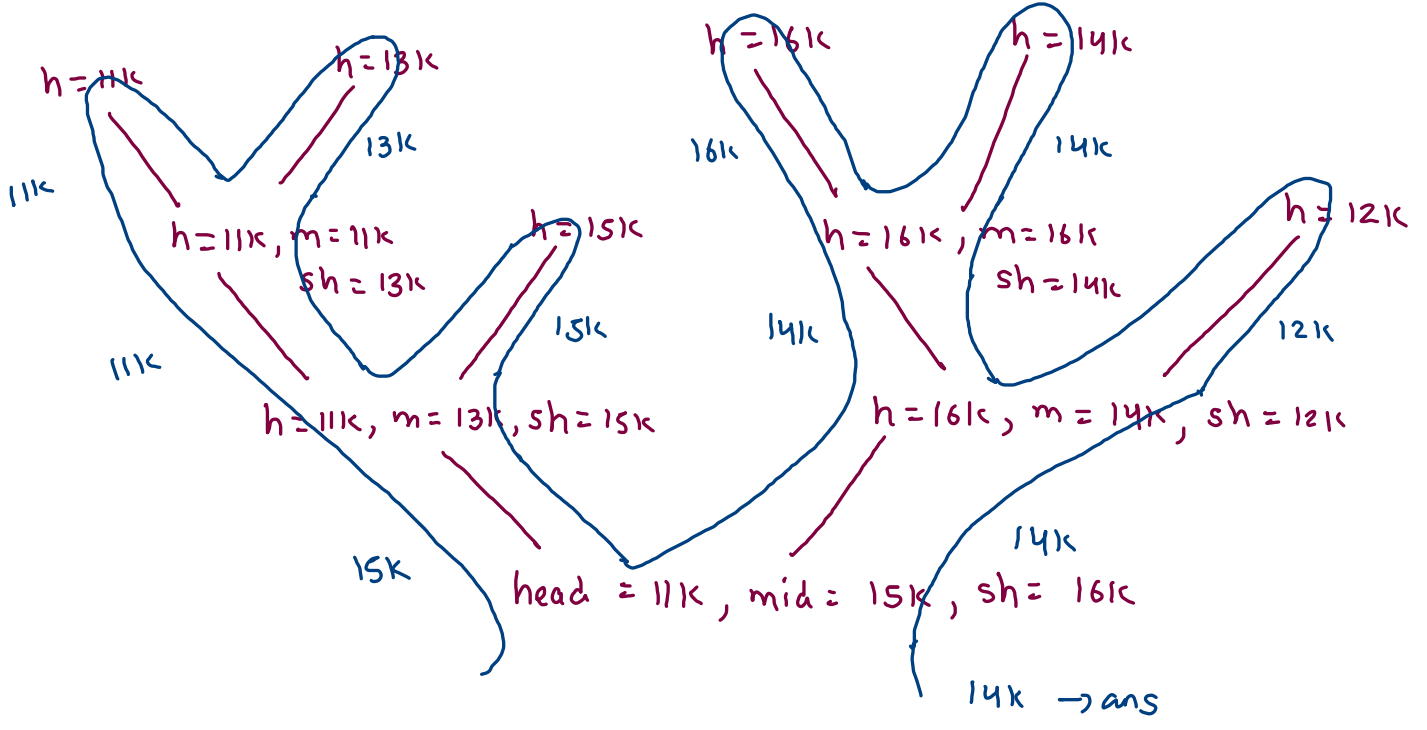




ans : 7k

n = 6


7
9
5
3
1
4
11k
13k
15k
16k
14k
12k


h = 11k
h = 13k
11k
13k
h = 11k, m = 11k
sh = 13k
h = 15k
15k
11k
h = 11k, m = 13k, sh = 15k
15k
h = 16k
h = 14k
16k
14k
h = 16k, m = 16k
sh = 14k
h = 12k
14k
12k
h = 16k, m = 14k, sh = 12k
14k
head = 11k, mid = 15k, sh = 16k
14k → ans

```java
public ListNode sortList(ListNode head) {
    if(head == null || head.next == null) {
        //there is only one node
        return head;
    }

    ListNode mid = mid(head);
    ListNode sh = mid.next;
    mid.next = null;

    ListNode lh = sortList(head);
    ListNode rh = sortList(sh);

    return mergeTwoSortedLL(lh,rh);

}
```

$T(n) = n + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n$

$T(n) = 2n + 2T\left(\frac{n}{2}\right)$ — (i)

$2T\left(\frac{n}{2}\right) = 2\frac{n}{2} + 4T\left(\frac{n}{4}\right)$ — (ii) X 2

$4T\left(\frac{n}{4}\right) = 2\frac{n}{4} + 8T\left(\frac{n}{8}\right)$ — (iii) X 4

$\vdots$

$T(1) = c$

$T(n) = 2n * t$

$T(n) = 2n * \log_2 n$

$\boxed{T(n) \propto n\log_2 n}$

$n, \frac{n}{2}, \frac{n}{4} \ldots\ldots 1$

$a = n,\ r = \frac{1}{2},\ t = t$

$ar^{t-1} = 1$

$n\left(\frac{1}{2}\right)^{t-1} = 1$

$n = 2^{t-1}$

$\log_2 n = t-1$

$\boxed{t = \log_2 n + 1}$

space : recursive

height = $\log_2 n$

143. Reorder List

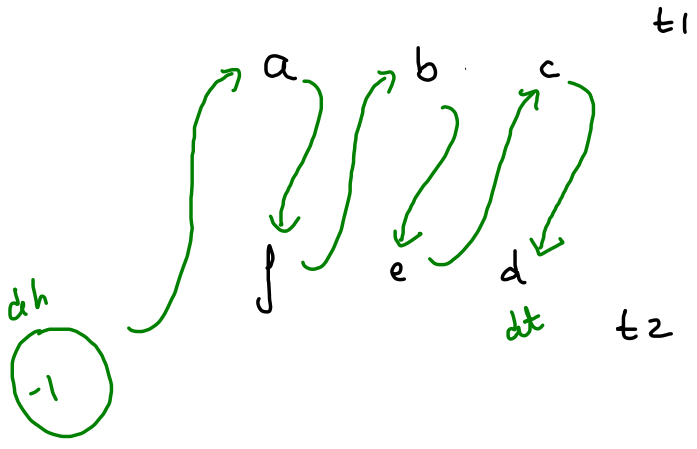
Medium 5425 201 Add to List Share

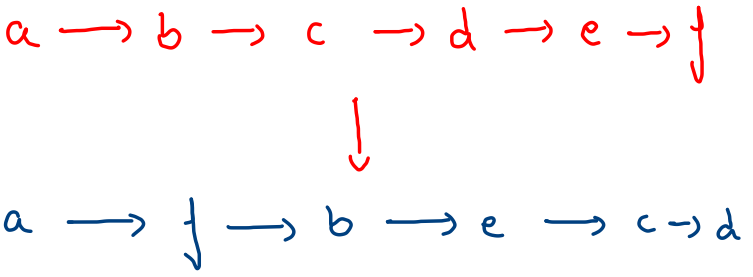
You are given the head of a singly linked-list. The list can be represented as:

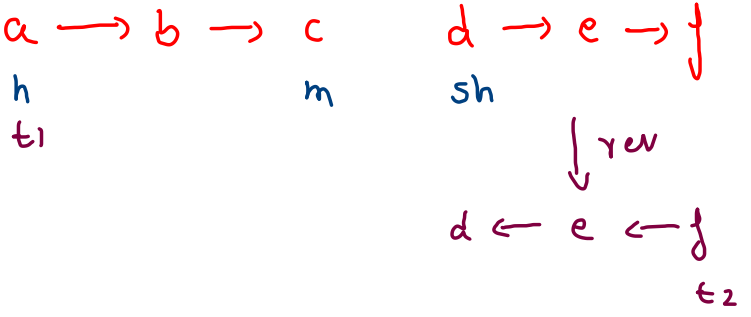
$L_0 \rightarrow L_1 \rightarrow \ldots \rightarrow L_{n-1} \rightarrow L_n$

Reorder the list to be on the following form:

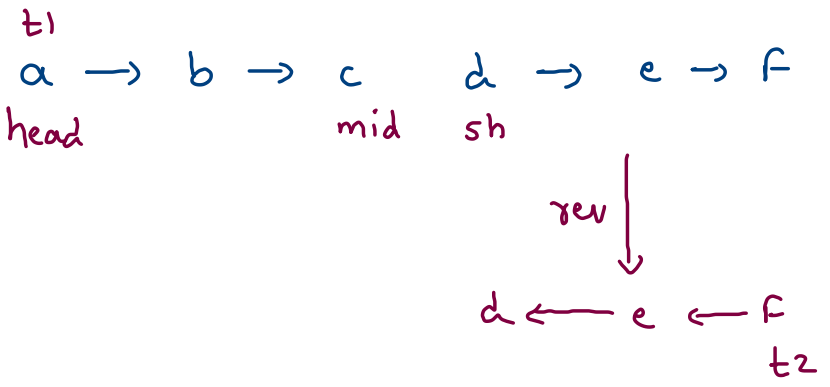
$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \ldots$

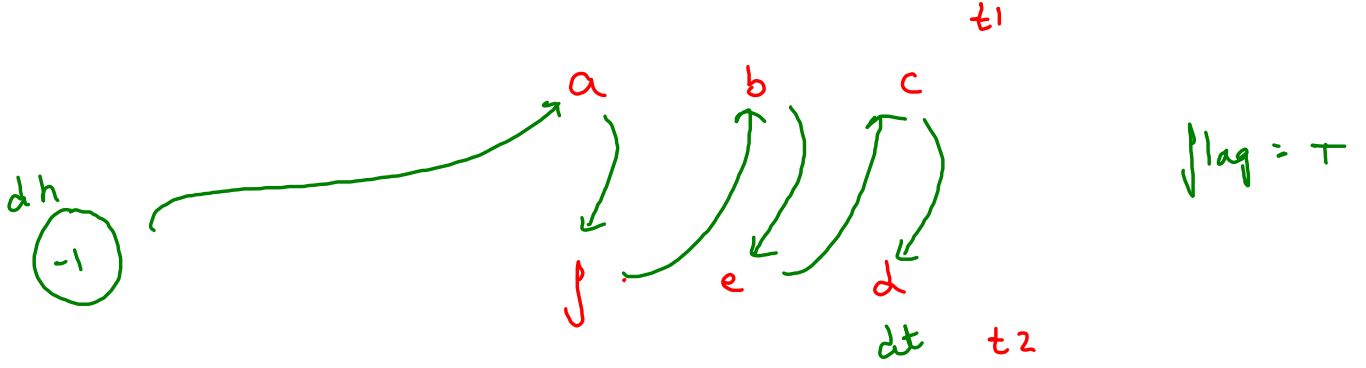




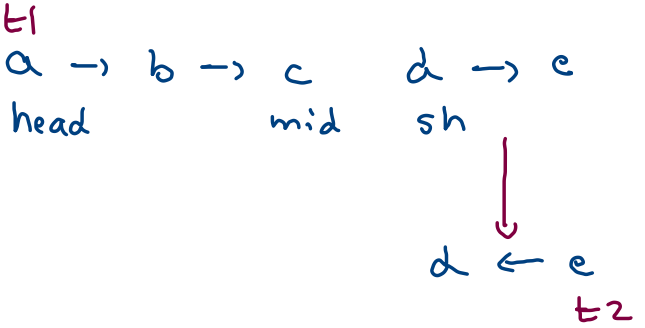


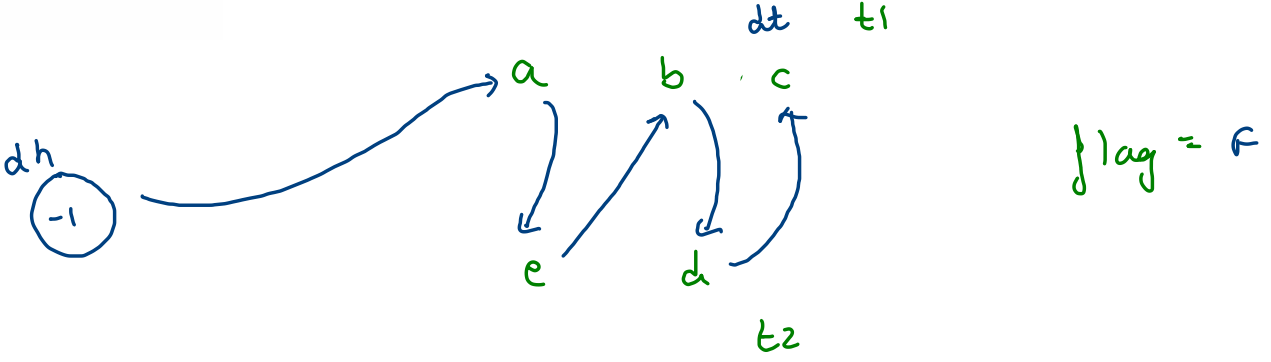
```
while(t1 != null || t2 != null) {
    if(flag == true) {
        dt.next = t1;
        t1 = t1.next;
    }
    else {
        dt.next = t2;
        t2 = t2.next;
    }

    flag = !flag;
    dt = dt.next;
}
```





```
while(t1 != null || t2 != null) {
    if(flag == true) {
        dt.next = t1;
        t1 = t1.next;
    }
    else {
        dt.next = t2;
        t2 = t2.next;
    }

    flag = !flag;
    dt = dt.next;
}
```

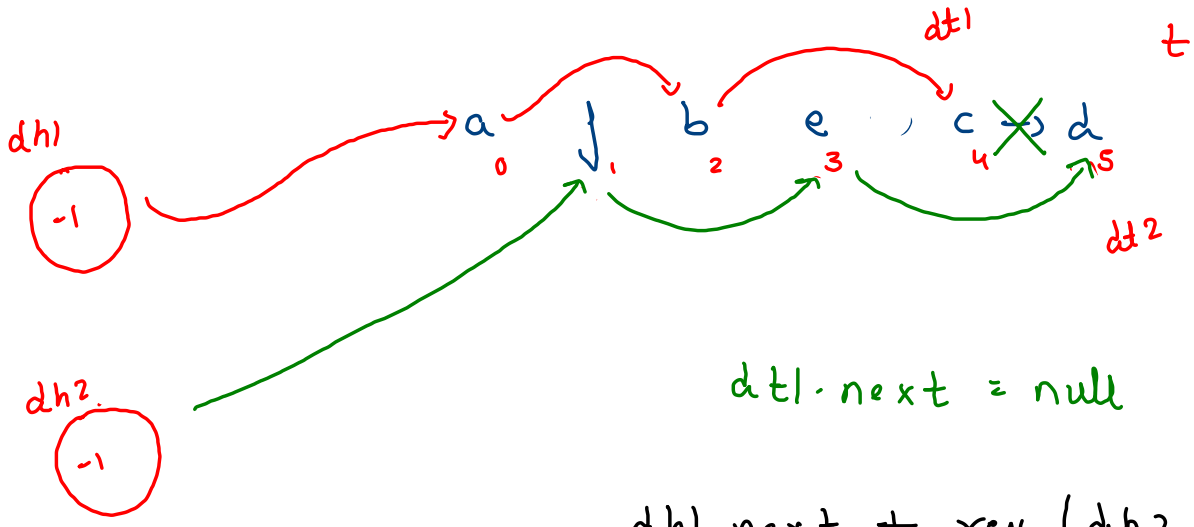




Unfold Of Linkedlist

folded a → f → b → e → c → d

original a → b → c → d → e → f



(segregation)

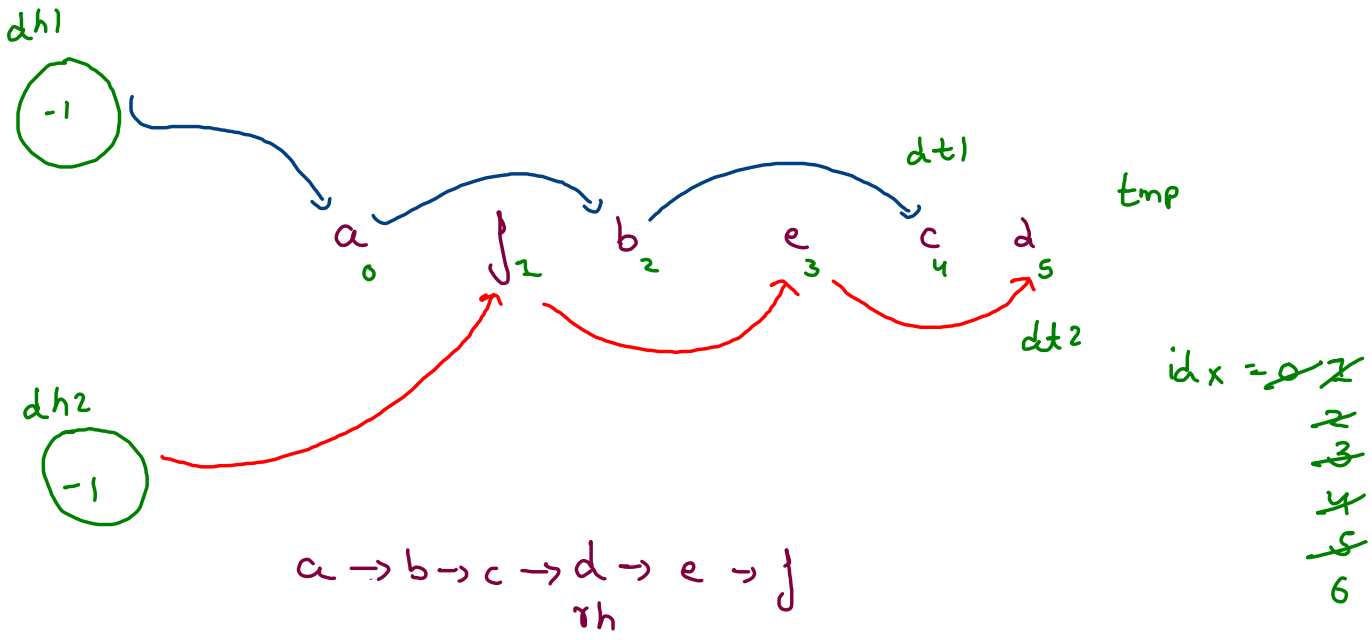
c = ~~0 1 2 3 4 5~~
6

dtl.next = null

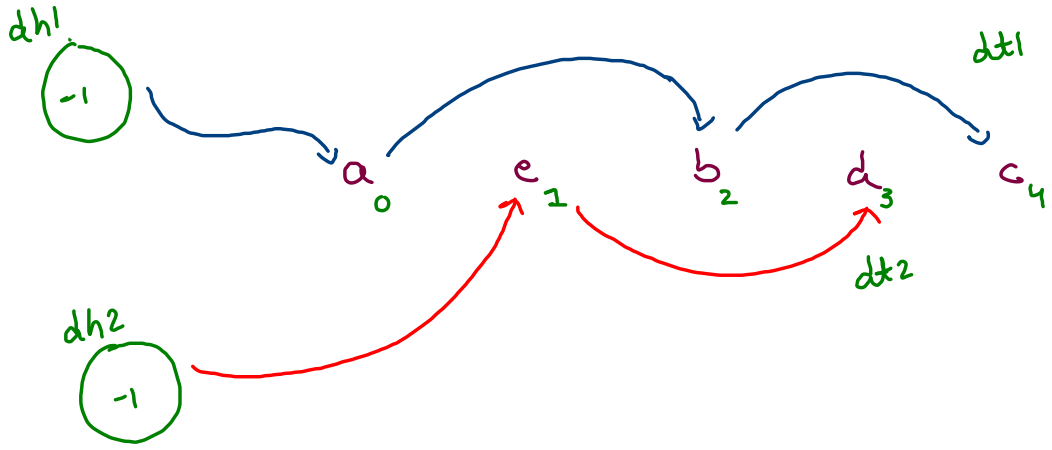
dhl.next + rev (dh2.next)

↳ appending

```java
int idx = 0;
ListNode temp = head;

while(temp != null) {
    if(idx % 2 == 0) {
        dt1.next = temp;
        dt1 = dt1.next;
    }
    else {
        dt2.next = temp;
        dt2 = dt2.next;
    }
    idx++;
    temp = temp.next;
}

dt1.next = null;
dt2.next = null;

ListNode rh = reverse(dh2.next);

//append
dt1.next = rh;
```



```java
int idx = 0;
ListNode temp = head;

while(temp != null) {
    if(idx % 2 == 0) {
        dt1.next = temp;
        dt1 = dt1.next;
    }
    else {
        dt2.next = temp;
        dt2 = dt2.next;
    }
    idx++;
    temp = temp.next;
}

dt1.next = null;
dt2.next = null;

ListNode rh = reverse(dh2.next);

//append
dt1.next = rh;
```



idx = 0 1 2 3 4 5

a → b → c → d → e

rh