

DS WEEK-8

```
Start here X d.c X doubly.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct node
4  {
5      int data;
6      struct node* prev;
7      struct node* next;
8  };
9  struct node* head=NULL;
10
11 void insertatEnd(int value)
12 {
13     struct node* newnode=(struct node*)malloc(sizeof(struct node));
14     newnode->data=value;
15     newnode->prev=NULL;
16     newnode->next=NULL;
17
18     if (head == NULL) {
19         head = newnode;
20         return;
21     }
22
23     struct node* temp= head;
24     while(temp->next!=NULL)
25         temp=temp->next;
26
27     temp->next=newnode;
28     newnode->prev=temp;
29 }
30 void insertatLeft(int key, int value)
31 {
32     struct node* temp=head;
33     while (temp!=NULL && temp->data!=key)
34         temp=temp->next;
35
36     if(temp==NULL)
37     {
38         printf("key not found\n");
39         return;
40     }
41     struct node* newnode=(struct node*)malloc(sizeof(struct node));
42     newnode->data=value;
43
44     newnode->next=temp;
45     newnode->prev=temp->prev;
```

```
47     if(temp->prev!=NULL)
48         temp->prev->next=newnode;
49
50     else
51         head=newnode;
52     printf("inserted %d to the left of %d\n", value, key);
53 }
54
55 void deleteValue(int key)
56 {
57     struct node* temp=head;
58     while(temp!=NULL && temp->data!=key)
59         temp=temp->next;
60
61     if(temp==NULL)
62     {
63         printf("value not found\n");
64         return;
65     }
66     if (temp->prev!=NULL)
67         temp->prev->next=temp->next;
68     else
69         head=temp->next;
70
71     if (temp->next!=NULL)
72         temp->next->prev=temp->prev;
73
74     free(temp);
75     printf("deleted node with value %d\n, key");
76 }
77 void display()
78 {
79     struct node* temp=head;
80
81     if(temp==NULL)
82     {
83         printf("list is empty\n");
84         return;
85     }
86     printf("doubly linked list:");
87     while(temp!=NULL)
88     {
89         printf("%d<->", temp->data);
90         temp=temp->next;
91     }
92     printf("NULL\n");
93 }
```

```
94 int main()
95 {
96     int choice, value, key;
97     while(1)
98     {
99         printf("\n-----menu-----\n");
100         printf("1.insert node at end(create list)\n");
101         printf("2.insert left to node\n");
102         printf("3.delete by value\n");
103         printf("4. display list\n");
104         printf("5.exit\n");
105         printf("enter choice:");
106         scanf("%d", &choice);
107
108         switch(choice)
109         {
110             case 1: printf("enter value to insert:");
111                     scanf("%d", &value);
112                     insertatEnd(value);
113                     break;
114
115             case 2: printf("enter existing node value(key):");|
116                     scanf("%d", &key);
117                     printf("enter new value to insert left:");
118                     scanf("%d", &value);
119                     insertatLeft(key,value);
120                     break;
121
122             case 3: printf("enter value to delete:");
123                     scanf("%d", &key);
124                     deleteValue(key);
125                     break;
126
127             case 4:
128                 display();
129                 break;
130
131             case 5:
132                 exit(0);
133
134             default:
135                 printf("invalid choice\n");
136         }
137     }
138     return 0;
139 }
140
```



"C:\Users\BMSCE\Documents" X



```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:10
```

```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:20
```

```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:60
```

```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:80
```

```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:3
enter value to delete:20
deleted node with value 65536
, key
```

```
-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
```

```

C:\Users\BMSCE\Documents' X + v
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:4
doubly linked list:10<-->60<-->80<-->NULL

-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:2
enter existing node value(key):
60
enter new value to insert left:100
inserted 100 to the left of 60

-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:2
enter existing node value(key):80
enter new value to insert left:200
inserted 200 to the left of 80

-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:4
doubly linked list:10<-->100<-->60<-->200<-->80<-->NULL

-----menu-----
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:5

Process returned 0 (0x0)   execution time : 76.650 s
Press any key to continue.

```

← → ↺

leetcode.com/problems/minimum-pair-removal-to-sort-array-/description/?envType=problem-list-v2&envId=doubly-linked-list

☆ ⓘ ⌵

🔥 Doubly-Linked List < > 🔍

🔥 Submit 📄 🏠

🔥 Premium

Description Accepted Editorial Solutions Submissions

3507. Minimum Pair Removal to Sort Array I Solved

Easy Topics Companies Hint

Given an array `nums`, you can perform the following operation any number of times:

- Select the **adjacent** pair with the **minimum** sum in `nums`. If multiple such pairs exist, choose the leftmost one.
- Replace the pair with their sum.

Return the **minimum number of operations** needed to make the array **non-decreasing**.

An array is said to be **non-decreasing** if each element is greater than or equal to its previous element (if it exists).

Example 1:

Input: `nums = [5,2,3,1]`

Output: 2

Explanation:

- The pair `(3,1)` has the minimum sum of 4. After replacement, `nums = [5,2,4]`.
- The pair `(2,4)` has the minimum sum of 6. After replacement, `nums = [5,6]`.

The array `nums` became non-decreasing in two operations.

Example 2:

Input: `nums = [1,2,2]`

Output: 0

Explanation:

The array `nums` is already sorted.

</> Code

C Auto

```
1 #include <stdbool.h>
2 #include <limits.h>
3
4 bool isSorted(int* arr, int n) {
5     for (int i = 1; i < n; i++) {
6         if (arr[i-1] > arr[i]) return false;
7     }
8     return true;
9 }
10
11 int minimumPairRemoval(int* nums, int numsSize) {
12     int operations = 0;
13
14     while (!isSorted(nums, numsSize)) {
15         int minSum = INT_MAX;
16         int idx = 0;
17
18         // Find leftmost adjacent pair with minimum sum
19         for (int i = 0; i < numsSize - 1; i++) {
20             int s = nums[i] + nums[i+1];
21             if (s < minSum) {
22                 minSum = s;
23                 idx = i;
24             }
25         }
26
27         // Replace pair with their sum
28         nums[idx] = minSum;
29         for (int j = idx + 1; j < numsSize - 1; j++) {
30             nums[j] = nums[j+1];
31         }
32         numsSize--; // shrink array
33         operations++;
34     }
35
36     return operations;
37 }
```

Saved Ln 37, Col 2

🧪 Testcase ➤ Test Result

👤 52 Online Accepted Runtime: 0 ms



Submit



</>Code

C   Auto

```
1  #include <stdbool.h>
2  #include <limits.h>
3
4  bool isSorted(int* arr, int n) {
5      for (int i = 1; i < n; i++) {
6          if (arr[i-1] > arr[i]) return false;
7      }
8      return true;
9  }
10 int minimumPairRemoval(int* nums, int numsSize) {
11     int operations = 0;
12
13     while (!isSorted(nums, numsSize)) {
14         int minSum = INT_MAX;
15         int idx = 0;
16
17         // Find leftmost adjacent pair with minimum sum
18         for (int i = 0; i < numsSize - 1; i++) {
19             int s = nums[i] + nums[i+1];
20             if (s < minSum) {
21                 minSum = s;
22                 idx = i;
23             }
24         }
25
26         // Replace pair with their sum
27         nums[idx] = minSum;
28         for (int j = idx + 1; j < numsSize - 1; j++) {
29             nums[j] = nums[j+1];
30         }
31         numsSize--; // shrink array
32         operations++;
33     }
34     return operations;
35 }
```

☒ Testcase | [>_ Test Result](#)

Accepted Runtime: 0 ms

☒ Case 1

☒ Case 2

Input

```
nums =  
[5,2,3,1]
```

Output

```
2
```

Expected

```
2
```

 Contri