YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ ALGORİTMA ANALİZİ ÖDEVİ

SEMİH ERKEN 19011503

1.Brute Force

```
45      while(temp != NULL){
46         if(temp->val > max_Val_node){
47              max_Val_node = temp->val;
48              first = temp->first_Index;
49              last = temp->last_Index;
50         }
51          temp = temp->next;
52      }
53      printf("First Index : %d Last Index : %d Max Value : %d", first ,last ,max_Val_node);
54      return 0;
55  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Yedekle\c Lab> cd "c:\Yedekle\c Lab\" ; if ($?) { gcc 19011503_1.c -o 19011503_1 } ; if ($?) { .\19011503_1 }
Enter the How Many Elements You Will Enter : 10
1. Element : -5
2. Element : 7
3. Element : -30
4. Element : 21
5. Element : -7
6. Element : 32
7. Element : -15
8. Element : 10
9. Element : -42
10. Element : 25
First Index : 3 Last Index : 5 Max Value : 46
PS C:\Yedekle\c Lab>
```

```
44      max_Val_node = temp->val;
45      while(temp != NULL){
46         if(temp->val > max_Val_node){
47              max_Val_node = temp->val;
48              first = temp->first_Index;
49              last = temp->last_Index;
50         }
51          temp = temp->next;
52      }
53      printf("First Index : %d Last Index : %d Max Value : %d", first ,last ,max_Val_node);
54      return 0;
55  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

PS C:\Yedekle\c Lab> cd "c:\Yedekle\c Lab\" ; if ($?) { gcc 19011503_1.c -o 19011503_1 } ; if ($?) { .\19011503_1 }
Enter the How Many Elements You Will Enter : 12
1. Element : 45
2. Element : 34
3. Element : 25
4. Element : -50
5. Element : 20
6. Element : -60
7. Element : 71
8. Element : 23
9. Element : -20
10. Element : 15
11. Element : 4
12. Element : -42
First Index : 0 Last Index : 7 Max Value : 108
PS C:\Yedekle\c Lab>
```

Yukarıda görüldüğü gibi çıktılarda herhangi bir problem yoktur.

```
    node* values = (node*)malloc(sizeof(node));
    node *head = values;
    int max_Val = 0 , counter = 0;

    int i; for(i = 1; i <= n ; i++){
        int j; for(j = 0; j < n - i + 1 ; j++){
            int k; for(k =  j ; k <= j + i - 1 ; k++){
                max_Val = max_Val + area[k];
            }
            node *temp = (node*)malloc(sizeof(node));
            temp->first_Index = j;
            temp->last_Index = j + i - 1;
            temp->val = max_Val;
            temp->next = NULL;
            values->next = temp;
            values = temp;
            max_Val = 0;
        }
    }
```

Brute force algoritmamızın çalışma mantığı dizi üzerindeki bütün kombinasyonları denemektir.

Örnek verecek olursak 10 elemanlı bir dizi için 1,2,3,4,5,6,7,8,9, ve 10 elemanlı bütün alt dizilerin elemanları toplanır bu yapılırken örneğin 3 elemanlı alt dizilerde 0-2 , 1-3 , 2-4, 3-5 ,4-6 ,5-7 ,6-8 , 7-9 şeklindeki indislerin barındırdığı değerler toplanır ve bu alt dizilerin başlangıç bitiş indisleri ile toplam değerleri linkli listeye kaydedilir.

```
    }
    node*temp = head->next;
    max_Val_node = temp->val;
    while(temp != NULL){
       if(temp->val > max_Val_node){
            max_Val_node = temp->val;
            first = temp->first_Index;
            last = temp->last_Index;
       }
        temp = temp->next;
    }
    printf("First Index : %d Last Index : %d Max Value : %d", first ,last ,max_Val_node);
    return 0;
}
```

Burada da görüldüğü üzere linkli listedeki tüm düğümler gezilir maximum toplam değere sahip alt küme bulunur ve koşul ifadesine girildiğinde linkli listenin o düğümünden başlangıç bitiş indisleri ile toplam değerleri çekilerek ekrana yazdırılır.

2. DIVIDE AND CONQUER

```
63
64  int main(void)
65  {
66      int area[] = { 8, -30, 36, 2, -6, 52, 8, -1, -11, 10, 4};
67      int n = sizeof(area) / sizeof(area[0]);
68
69      printf("%d to %d Max Value Is : %d",max_range(area , 0 , n-1)->first_Index, max_range(area , 0 , n-1)->last_Index, max_range(area , 0
70
71      return 0;
72  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER                                   Code + ⌄

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Yedekle\c Lab> cd "c:\Yedekle\c Lab\" ; if ($?) { gcc 19011503_2.c -o 19011503_2 } ; if ($?) { .\19011503_2 }
2 to 10 Max Value Is : 94
PS C:\Yedekle\c Lab>
```

```
64  int main(void)
65  {
66      int area[] = { -5, -10, 20, 36, -60, -20, -30, 6, 50, 100, 40, -5};
67      int n = sizeof(area) / sizeof(area[0]);
68
69      printf("%d to %d Max Value Is : %d",max_range(area , 0 , n-1)->first_Index, max_range(area , 0 , n-1)->last_Index, max_range(area ,
70
71      return 0;
72  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER                                   Code +

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Yedekle\c Lab> cd "c:\Yedekle\c Lab\" ; if ($?) { gcc 19011503_2.c -o 19011503_2 } ; if ($?) { .\19011503_2 }
7 to 10 Max Value Is : 196
PS C:\Yedekle\c Lab>
```

Görüldüğü üzere çıktılarda herhangi bir problem yoktur.

Buradaki algoritmamızın ana mantığı elimizdeki diziyi ikiye böle böle en küçük parçalara ayırıp ardından bu parçalardan büyük olanı seçip birleştirip, birleşmiş parçalardan büyük olanı seçip alttan gelen parçayla kıyaslayıp küçük ise bırakmak büyük ise o parçayı kullanmaktır alttan gelenleri kıyasladığımız bu yöntem sayesinde en büyük parça bütünü elde edilirken ard arda gelen 3 ya da 4 sayının toplamından daha büyük olduğu halde tek parça olan sayılarda göz önünden kaçmayarak listenin düzgün bir şekilde güncellenmesini sağlamışlardır

Kodun çalışma prensibi aşağıdaki gibidir

```
node* max_range(int area[], int low, int high)
{
    if (high <= low) {
        node* newNode = (node*)malloc(sizeof(node));
        newNode->val = area[low];
        newNode->first_Index = low;
        newNode->first_Index = high;
        return newNode;
    }
    int mid = (low + high) / 2;
    node *left_half_max = (node*)malloc(sizeof(node));
    left_half_max->val = INT_MIN;
    int sum = 0;
    for (int i = mid; i >= low; i--)
    {
        sum += area[i];
        if (sum > left_half_max->val) {
            left_half_max->val = sum;
            left_half_max->first_Index = i;
            left_half_max->last_Index = mid;
        }
    }
```

Dizinin başı ve sonu alınır ve ortası bulunur sonra bu orta noktadan başa doğru gidilirken önce ilk sayı max ilan edilir sonra bu ilerleme esnasında toplanarak gidilirken eğer bu sayıdan daha büyük bir değer elde edilir ise struct yapısına bu değer ve bu değeri kapsayan aralığın başlangıç ve bitiş indisleri kaydedilir.

```
node *right_half_max = (node*)malloc(sizeof(node));
right_half_max->val = INT_MIN;
sum = 0;
for (int i = mid + 1; i <= high; i++)
{
    sum += area[i];
    if (sum > right_half_max->val) {
        right_half_max->val = sum;
        right_half_max->first_Index = mid+1;
        right_half_max->last_Index = i;
    }
}

node* max_left_right_decide = (node*)malloc(sizeof(node));
max_left_right_decide = find_max(max_range(area, low, mid),max_range(area, mid + 1, high));

node *sum_of_right_left_halfs = (node*)malloc(sizeof(node));
sum_of_right_left_halfs->val = left_half_max->val + right_half_max->val;
sum_of_right_left_halfs->first_Index = left_half_max->first_Index;
sum_of_right_left_halfs->last_Index = right_half_max->last_Index;
return find_max(max_left_right_decide, sum_of_right_left_halfs);
}
```

Sol yarım halledildikten sonra orta noktadan dizinin sonuna doğru gidilir ve aynı şekilde önce ilk sayı max kabul edilir sonra ilerleme esnasındaki toplamlarda ilk değerden fazlalaştığı vakit bu aralığın başlangıç bitş ve toplam değeri struct yapısına kaydedilir. Sonrasında tekrardan kendisini çağırır ancak bu yapılırken sağ ve sol yarımlar ayrı ayrı gönderilir bu sayede yukarıda yapılan bölme işlemi en aşağıya kadar yapılır ve yukarıya bölünerek elde edilmiş maksimum değer gelir.

Kritik nokta aşağıdan maksimum olarak gelen değerler direkt olarak kabul edilmez burada öz yinelemeli olarak gelen değerler ile mevcut fonksyion içindeki bölünmüş ve maksimum kabul edilmiş sağdan ve soldan birleştirilmiş olan (structta başlangıç indisi olarak sol yarımın ilk indisi ile bitiş olarak sol yarımın son indisi olarak atanır) aralık kıyaslanır ve son olarak max olan değer belirlenir ve diğer tüm belirleme işlemlerinde olduğu gibi başlangıç ve bitiş parametreleri ile berabar aktarılır.