ALGORİTMA ANALİZİ ASSIGNMENT 2 RAPORU

AHMET ENİS ŞİMŞİR

19011077

enis.simsir@std.yildiz.edu.tr

https://youtu.be/J1owEmwdUEA

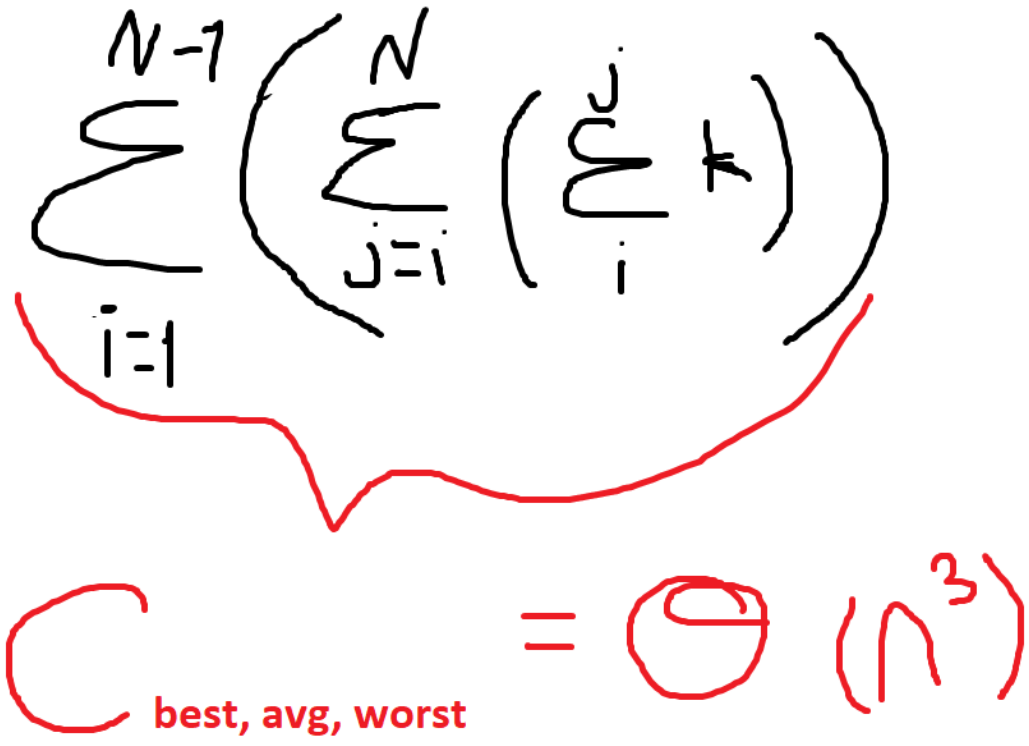
YÖNTEM

Brute Force

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
	int N, i, j, k, max=0;
	printf("maden boyu: ");
	scanf("%d",&N);
	int A[N];
	for(i=0;i<N;i++){
		printf("%d. deger: ",i);
		scanf("%d",&A[i]);
	}
	int sum, max_i, max_j;
// olabilecek her tür i ve j kombinasyonlarina bakiyoruz
	for(i=0;i<N-1;i++){
		for(j=i+1;j<N;j++){
			sum=0;
			// her kombinasyon icin toplam degeri buluyoruz
			for(k=i;k<=j;k++){
				sum+=A[k];
			}
			// en buyun kombinasyonu seciyoruz
```

```
                if(sum>max){
                        max=sum;
                        max_i=i;
                        max_j=j;
                }
            }
        }

    printf("max kazanc: %d\nkazilmasi gerekilen blok butunlugu: %d-%d",max,max_i,max_j);

    return 0;
}
```

Kısaca açıklamak gerekirse Brute Force yöntemi ile her tür ihtimali deniyoruz ve en iyi sonuç vereni seçiyoruz.



$$C_{best,\ avg,\ worst} = \Theta(n^3)$$

zorunda olduğumuz için tek bir case var

Sözde Kod:

```
sum <- 0
max <- 0
i <- 1
Repeat i <- i + 1 till i = N-1:
        j <- i + 1
        Repeat j <- j +1 till j = N:
                sum <- 0
                k <- i
                        Repeat k <- k+1 till k = j:
                                sum <- sum + A(k)
                        if sum>max:
                                max <- sum
                                max_i <- I max_j <-j
```

Divide and Conquer

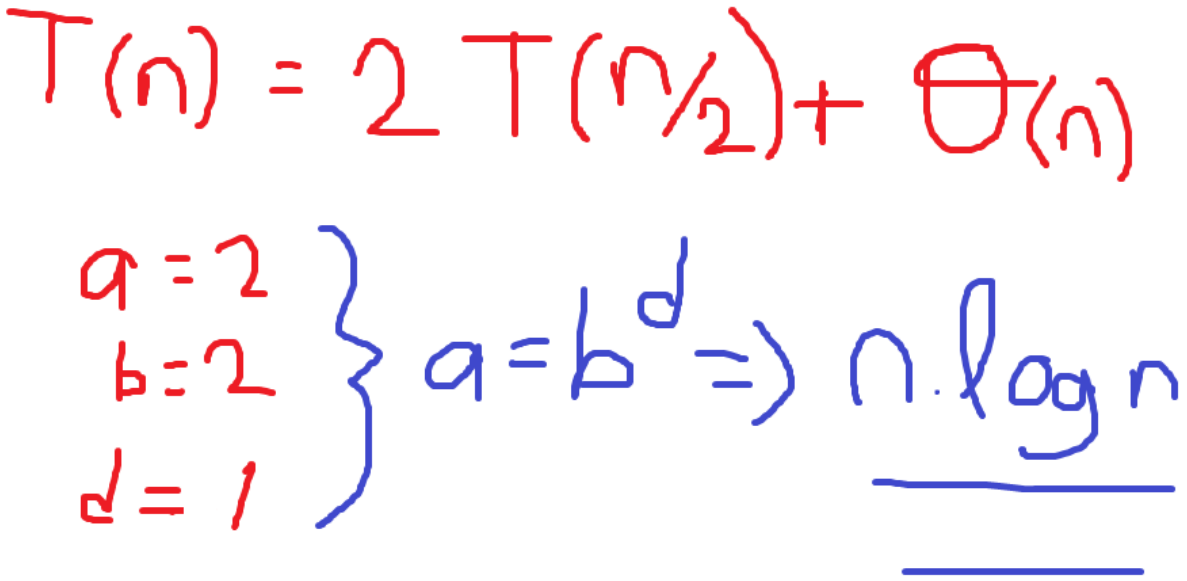
```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

int divide(int A[], int low, int high,int *first, int *last);
int max(int a, int b);

int main(int argc, char *argv[]) {
        int N, k;
        printf("maden boyu: ");
        scanf("%d",&N);
```

```
        int A[N];
        for(k=0;k<N;k++){
                printf("%d. deger: ",k);
                scanf("%d",&A[k]);
        }
        int i, j;
        int sonuc = divide(A, 0, N - 1, &i, &j);
        printf("en kazancli sonuc: %d\n%d ile %d arasi",sonuc,i,j);
        return 0;
}

int divide(int A[], int low, int high,int *first, int *last){
        //bolum kisminda tek bir deger mi kaldi kontrolu
        if (high <= low){
    return A[low];
  }
        int mid = (low + high) / 2;
        int maxLeft = -21000000;
        int maxRight = -21000000;
        int i, tmpFirst, tmpLast;
        int sum=0;
// dizinin soluna bakiyoruz
        for (i = mid; i >= low; i--){
    sum += A[i];
    if (sum > maxLeft) {
      maxLeft = sum;
      tmpFirst = i;
    }
  }
  sum=0;
```

```
// dizinin sagina bakiyoruz
   for (i = mid + 1; i <= high; i++){
      sum += A[i];
      if (sum > maxRight) {
         maxRight = sum;
         tmpLast = i;
      }
   }
   // burada recursive olarak dizinin once soluna sonra sagina bakiyoruz
   // ve fonksiyona her girdiginde ayni kontrol devam ediyor ta ki tek bir eleman kalana kadar
   int maxAll = max(divide(A, low, mid, first, last), divide(A, mid + 1, high, first, last));
   // en buyuk deger saginda ya da solunda ise bu if'e giriyor
   if(maxAll>maxLeft + maxRight){
         return maxAll;
         }
         // degilse buna giriyor. ayrica first ve last pointerlari da indisleri tutmak icin
         // en buyuk deger orta noktanin saginda ya da solunda da olsa dizi surekli bolundugu icin
         // bir noktada dizinin ortasina denk gelecek o yuzden bu kisma illa ki girecek
         else{
               *first = tmpFirst;
               *last = tmpLast;
               return maxLeft + maxRight;
         }
}

int max(int a, int b){
         if(a>b)   return a;
         return b;
}
```

Kısaca açıklamak gerekirse recursive olarak diziyi her defasında ikiye bölüyoruz ve yeni alt dizileri de ikiye bölerek tek eleman kalana kadar işleme devam ediyoruz. En büyük değer bir noktada alt dizilerden birinin orta noktasına denk geleceği için o noktada indisleri de kaydediyoruz.

Eğer en büyük değer zaten orta noktaya denk geliyorsa bile dizinin iki tarafına bakarak bunu onaylamış oluyoruz.



Sözde Kod:

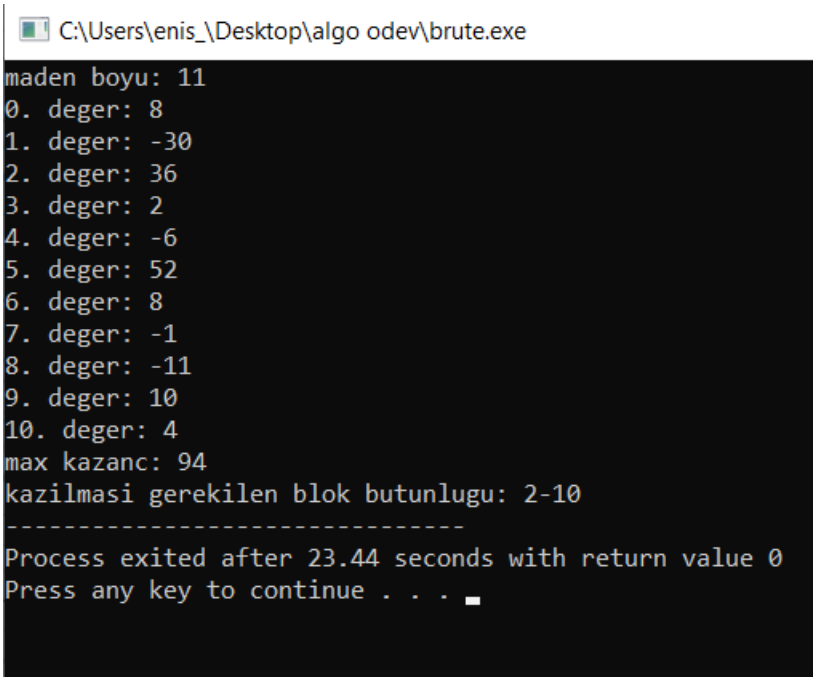
```
main():
        sonuc <- divide(A, 0, N, &i, &j)


divide(A, low, high, first, last):
        if high <= low:
                return A(low)

        mid <- (low+high)/2
        maxLeft <- en küçük sayı
        maxRight <- maxLeft
        sum <- 0
        I <- mid
```

```
repeat i <- i-1 till i = low:
        sum <- sum + A(i)
        if sum > maxLeft:
                maxLeft <- sum
                tmpFirst <- i
sum <- 0

repeat  i <- i+1 till i = high:
        sum <- sum + A(i)
        if sum > maxRight:
                maxRight <- sum
                tmpLast <- i

maxAll <- max(divide(A,low,mid,first,last),divide(A,mid+1,high,first,last)
if maxAll > maxLeft + maxRight:
        return maxAll

else:
        first <- tmpFirst
        last <- tmpLast
        return maxLeft + maxRight
```

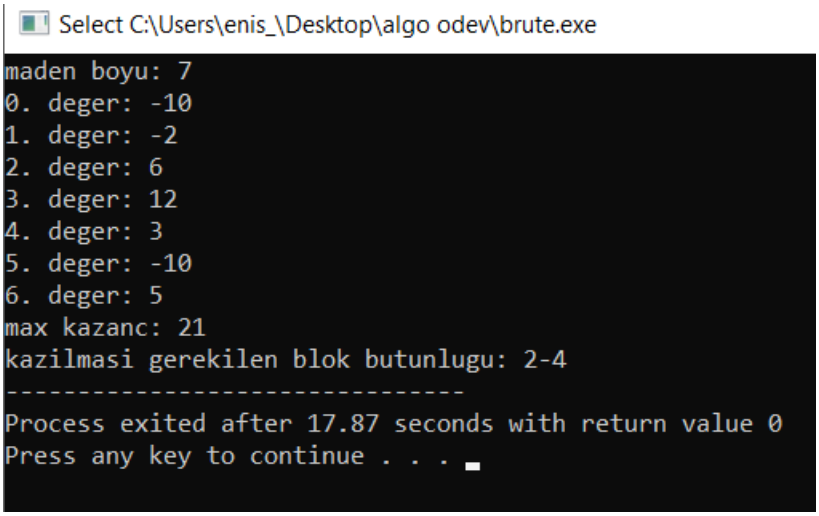
UYGULAMA

Brute Force

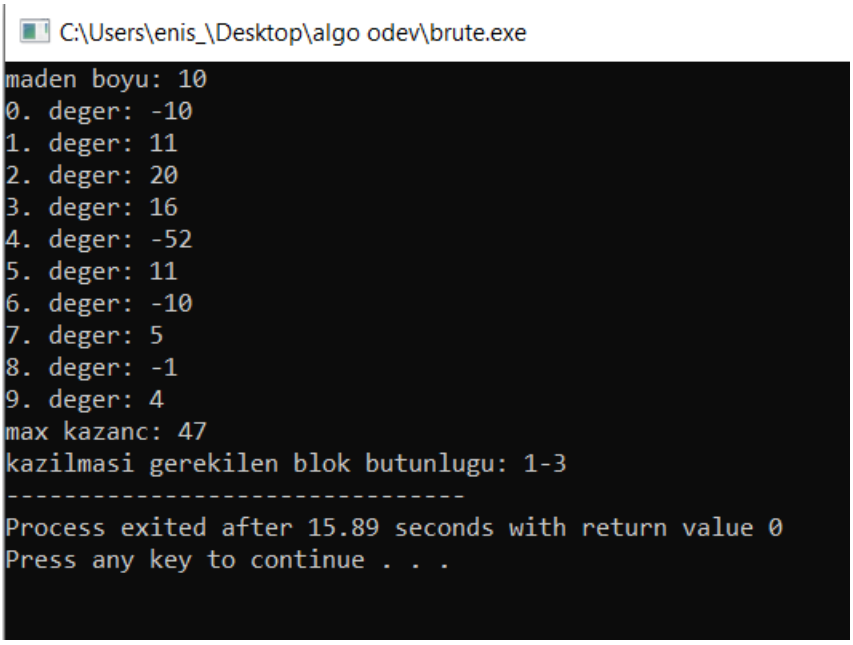
Değerler: 8 -30 36 2 -6 52 8 -1 -11 10 4



Değerler: -10 -2 6 12 3 -10 5

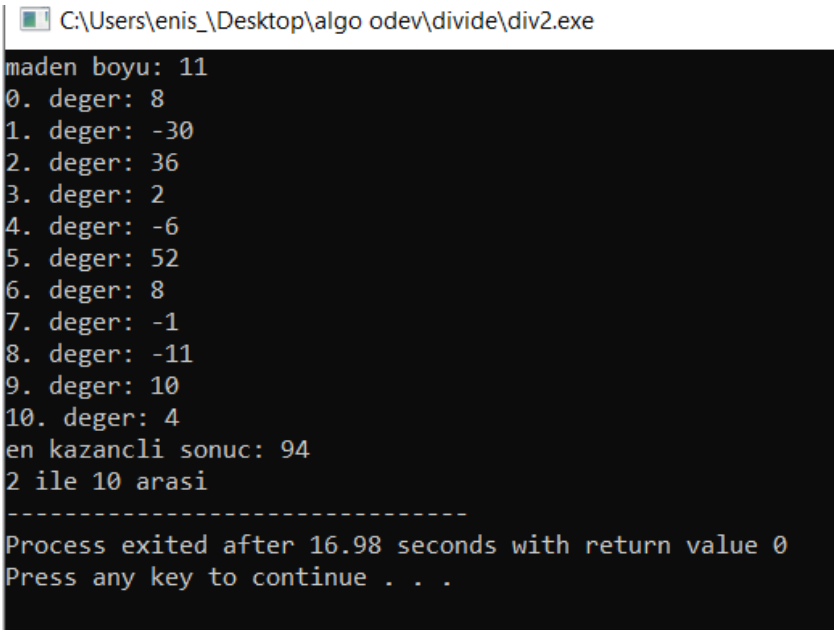


Değerler: -10 11 20 16 -52 11 -10 5 -1 4

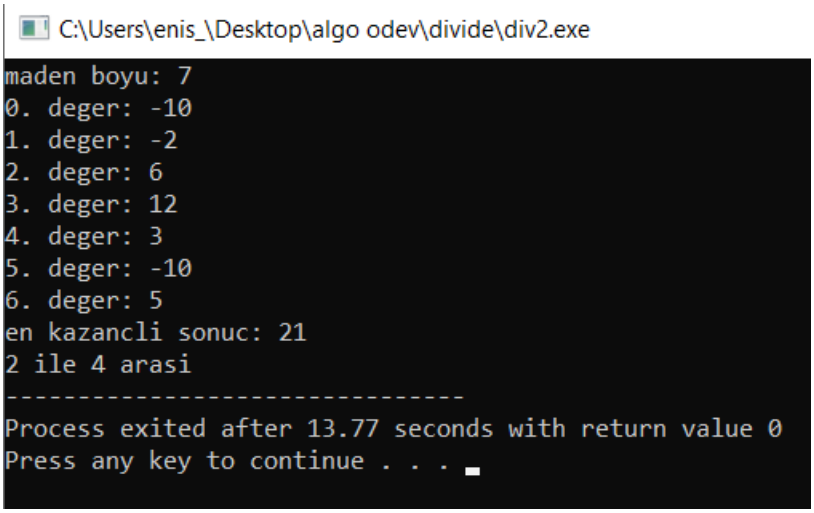


Divide and Conquer

Değerler: 8 -30 36 2 -6 52 8 -1 -11 10 4



Değerler: -10 -2 6 12 3 -10 5



Değerler: -10 11 20 16 -52 11 -10 5 -1 4