ARALIKTAKI ÖZEL SAYI ÇÖZÜMÜ

Soruda bizden özel bir sayı içeren en büyük altdiziyi istiyor. İlk olarak ekrandan değerleri okumamız gerekiyor.

```
cin >> n;
for (int i = 0; i < n; i++) {
    cin >> d[i];
}
for (int i = 0; i < n; i++) {
    cin >> ozel[i];
}
```

Sayıları ‘d’ dizisine ve her bir sayının özel olup olmadığını tutmak için de ‘ozel’ dizisini kullanabiliriz.

Altgörev - 1:

İlk olarak en düz çözümü düşünelim:

Her bir aralık için aralıkta özel bir sayı geçip geçmediğini bulabiliriz. Ayrıca o aralığın toplamını hesaplayabiliriz.

- Bunun için alt dizimizin sol ve sağ sınırlarını 0’dan n’e kadar olan bir döngü ile seçeriz. Her farklı sol ve sağ değişkenleri için gösterdikleri altdizinin cevap olup olamıyacağına bakarız.
- Her bir altdizi için o aralıktaki sayıların toplamı ve o aralıktaki özel sayı sayısını bulmak için bir döngü ile gerekli sayıları toplarız.
- Eğer aralıktaki özel sayı sayısı tam olarak 1 tane ise, bu alt dizi koşulu sağlıyordur. Bu koşulu sağlayan en büyük toplamı örnekteki gibi ‘cevap’ değişkeni tutarak buluruz.

```
for (int sol = 0; sol < n; sol++)
    for (int sag = sol; sag < n; sag++)
    {
        int toplam = 0, ozel_sayisi = 0;
        for (int i = sol; i < sag; i++)
        {
            toplam += d[i];
            ozel_sayisi += ozel[i];
        }
        if(ozel_sayisi == 1)
            cevap = max(cevap, toplam);
    }
```

Zaman Karmaşıklığı -Time complexity: $O(n^3)$
Hafıza Karmaşıklığı -Memory complexity: O(n)

Altgörev - 2: Bu alt görevde bütün sayılar özel olduğu için en büyük sayıyı ekrana bastırmamız yeterlidir. Çünkü bizden alt dizimizde sadece tek bir özel sayı geçmesini istemişler, o zaman maksimum altdizi uzunluğumuz bir olabilir.

Altgörev - 3: Bu alt görevde bütün sayılar pozitifdir. O zaman, özel sayı kuralını sağladığı sürece mümkün olan en büyük aralıkları seçebiliriz. Çünkü hiçbir sayının negatif etkisi olamaz. Aralıklar, tam olarak bir tane özel sayı içerecek en geniş aralıklar olacak. Bunu yapmak için her özel sayının kendinden bir önceki ile bir sonrakine kadar olan toplamı hesaplamamız yeterli.

Altgörev - 4: Öncelikle her bir altdizimiz tam olarak bir tane özel sayı içermesi gerektiği için, biz de sırayla her bir özel sayı için bulunduğu maksimum toplamlı altdiziyi hesaplamaya çalışacağız.

- Her bir özel sayıyı bul,
- Bu özel sayıdan sola doğru sırasıyla bir özel sayı ile karşılaşana kadar sayıları topla, toplamı maksimum olan aralığı kaydet
- Bu özel sayıdan sağa doğru sırasıyla bir özel sayı ile karşılaşana kadar sayıları topla, toplamı maksimum olan aralığı kaydet
- Bu iki toplamı ve asıl(özel) sayımızın değerlerini topladığımızda bu özel sayıyı içeren en büyük toplamlı altdiziyi bulmuş olacağız.

```
#include <bits/stdc++.h>
#define inf 1000000000LL
#define MAX_N 100005
using namespace std;

int n;
int d[MAX_N],ozel[MAX_N];
int en_iyi_sag(int a) {
   int sum = 0, best = 0;
   a += 1;
   while(a>=0 && a<n) {
       if(ozel[a]) break;
       sum+=d[a];
       best = max(best, sum);
       a += 1;
   }
   return best;
}
int en_iyi_sol(int a) {
   int sum = 0, best = 0;
   a -= 1;
   while(a>=0 && a<n) {
       if(ozel[a]) break;
```

```
        sum+=d[a];
        best = max(best, sum);
        a -= 1;
    }
    return best;
}
int main(){
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> d[i];
    }
    for (int i = 0; i < n; i++) {
        cin >> ozel[i];
    }
    int cevap = -inf;
    for (int i = 0; i < n; i++)
        if(ozel[i] == 1) {
            cevap = max(cevap, d[i] + en_iyi_sag(i) + en_iyi_sol(i));
        }
    cout << cevap << endl;
}
```

Zaman Karmaşıklığı -Time complexity: O(n)
Hafıza Karmaşıklığı - Memory complexity: O(n)

Her bir özel sayı için sağ ve sol tarafına doğru bir kere iterasyon(döngüde işlem) ediyoruz. Böylelikle her bir sayı, o sayının sağ tarafındaki ve sol tarafında bulunan özel sayı için hesaplama yaparken, maksimum 2 işlemde kullanabilir.