VERİ YAPILARI PART-2

**1)** **Main fonksiyonumuzdan gelen A dizisi sonucu sıralamamız yapılacaktır.**

insertionSort( array A)

for i 1 den length[A-1]’e kadar döner. // 1+n+1+n

Değer A[i] //n

j i-1 //n

while j>=0 ve A[j]>Değer // (n+1)\*n

A[j+1] A[j] // n\*n

j j-1 // n\*n

A[j+1] Değer //n

O(n) notasyonu sırasında algoritmasının zaman karmaşıklığı hesaplanırken, yapılan karşılaştırmalar ve yer değiştirmeler dikkate alınır. En fazla (n-1) karşılaştırma (n-1) yer değiştirme yapar ve bunun sonucunda (an^2+bn+c) şeklinde hesaplanan fonksiyonda Yapılan hesaplama sonucunda n(n-1) değerinin asimptotik üst sınırı O(N^2) değerini vermektedir.

**En kötü başarım:**Insertion algoritmasının en kötü olduğu durum listenin tersten sıralı olduğu durumdur. Bu durumda O(N^2) değerinin verir.

**En iyi başarım:**Insertion algoritması en iyi olduğu durum listenin sıralı olduğu durumdur. Bu durumda n-1 kere karşılaştırma yapar ve O(N) karmaşıklıkla çalışır.

**Ortalama başarım:** Eklemeli sıralama algoritması ortalama olarak O(N^2) karmaşıklıkla çalışır

**2)** **FOO FONKSİYONU**

|  |  |  |  |
| --- | --- | --- | --- |
| Temel Hesap Birimi | Birim Zaman | Frekans | Toplam |
| foo(n){ | - | - | - |
| for(int i=0;i<n;++i) | 1,1,1 | 1,n+1,n | 2n+2 |
| print(); | 1 | n | n |
| } | - | - | - |
|  |  |  | 3n+2 |

**CHECKFUN(N) FONKSİYONU**

|  |  |  |  |
| --- | --- | --- | --- |
| Temel Hesap Birimi | Birim Zaman | Frekans | Toplam |
| checkFunc(n){ | - | - | - |
| foo(n); | 3n+2 | 1 | 3n+2 |
| if(n==1) | 1 | 1 | 1 |
| return 1; | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| return checkFunc(n/2) | 1 | Log2 n | Log2n |
| } | - | - | - |
|  |  |  | (3n+5)+log2n |

**TEST FONKSİYONU**

|  |  |  |  |
| --- | --- | --- | --- |
| Temel Hesap Birimi | Birim Zaman | Frekans | Toplam |
| Test(n): | - | - | - |
| for(int i=1;i<2n;i+=4i) | 1,1,1 | 1,log52n,log5 2n | (Log5(2n)^2)+1 |
| for(int j=n;j>0;j--) | 1,1,1 | Log5(2n), Log5(2n)\*(n+1),n\* Log5(2n) | (2n+2)\* Log5(2n) |
| if(i\*j==target) | 1 | n\* Log5(2n) | n\* Log5(2n) |
| target=checkFunc(n); | (3n+6)+ log2n | n\* Log5(2n) | 3n^2+…=N^2O(Log) |
| else | 1 | n\* Log5(2n) | n\* Log5(2n) |
| print(); | 1 | n\* Log5(2n) | n\* Log5(2n) |
|  |  |  | O(N^2) tüm hepsi sığmazdı (O) notasyonu ile en sade halini yazdım |

İfademiz = (N^2O(logn) + aN+c …..) ifadesi olduğu için O notasyonu (N^2)\*logn olacaktır çünkü (N^2)\*logn fonksiyonun artış hızı diğerlerine göre daha yüksektir. Toplam satırında yazan tüm ifadelerin O notasyonu sonucunda (N^2)\*lognsonucunu elde ederiz. Sonuç olarak yazdığımız kodun maliyeti O(N2\*Log(n))’dir.