[22,27,16,2,18,6] -> Insertion Sort

* Yukarıda verilen dizinin sort türüne göre aşamaları yazınız.
  + Eleman sıralamaya ilk elemandan başlanır. Ve sonraki elemanlarla büyük sayı mı yoksa küçük sayı mı diye karşılaştırılır. Eğer, listedeki en küçük sayı ilk eleman değil ise, bulduğu en küçük eleman ile ilk elemanın yerleri değiştirilir.
    - [22,27,16,2,18,6] =>
      * 22 -> karşılaştırdığımız eleman ; 2 -> en küçük eleman
      * 22 > 2 => 2 ile 22’nin yerleri değiştirilir.
  + Sonraki elemana geçilir. Ve karşılaştırmaya en baştan değil, bulunduğumuz yerden sonraki elemanlarla karşılaştırılır. Son 1 eleman kalanada kadara her eleman atladığımızda aynı süreç devam eder.
    - [2,27,16,22,18,6] =>
      * 27 -> karşılaştırdığımız eleman; 6 -> en küçük eleman
      * 27 > 6 => 6 ile 27’nin yerleri değiştirilir.
    - [2,6,16,22,18,27] =>
      * 16 -> karşılaştırdığımız eleman; 16 -> en küçük eleman
      * 16’dan en küçük eleman olduğu için değişiklik olmuyor.
    - [2,6,16,22,18,27] =>
      * 22 -> karşılaştırdığımız eleman; 18 -> en küçük eleman
      * 22 > 18 => 18 ile 22’nin yerleri değiştirilir.
    - [2,6,16,18,22,27] =>
      * 22 -> karşılaştırdığımız eleman; 22-> en küçük eleman
      * 22’den en küçük eleman olduğu için değişiklik olmuyor.
  + Son 1 karşılaştırılacak eleman kaldığı için sortumuzu sonlandırıyoruz. Böylece son sortlanmış halimizi elde ediyoruz.
* Big-O gösterimini yazınız.
  + N tane elemandan, her bir elemanı sortlarken, bir sonraki eleman sort için her defasında N elemanın 1 eksiği kadar sort yapıyoruz. Bu işlemi 1 eleman kalana kadar gerçekleştiriyoruz.
    - Örneğin, 6 elemanlı array’in sort işlemi, ilk eleman için 6 karşılaştırma, sonraki elaman için 5 karşılaştırma, sonrakisi için 4 …… 1 eleman kalana kadar.
  + Formülize edersek : N+(N-1)+(N-2)+…+1 kadar işlem yapıyoruz. Buda (N\*(N+1))/2 => (N2+N)/2 demek oluyor. En yüksek üslü sayı N2 olduğu için => Big-O = O(N2)’dir.
* Time Complexity:
  + Average Case : Aradığımız sayının ortada olması
  + Best Case : Aradığımız sayının başta olması
  + Worst Case : Aradığımız sayının sonda olması
    - Average Case : Aradığımız sayının ortada olması : Eğer, arrayin başından başlıyorsak aramaya, O(n/2) => O(n)’dir. Diğer durumda arrayin uzunluğunun yarısı olacaktır. Yani sabittir => O(1)’dir.
    - Best Case : Aradığımız sayının başta olması : O(1)’dir.
    - Worst Case: Aradığımız sayının sonda olması : Eğer, arrayin başından başlıyorsak aramaya, O(n-1) => O(n)’dir. Diğer durumda sabittir => O(1)’dir.
* Dizi sıralandıktan sonra 18 sayısı hangi case kapsamına girer? Yazınız.
  + [2,6,16,18,22,27] => Sıralı Array, Array uzunluğu = 6;
  + 18 sayısı baktığımızda, 3. index’te yer alıyor. Arrayimizin uzunluğu 6 olduğuna göre, 6’nın yarısı 3’e denk geliyor. Yani 3.indeximiz arrayin ortası olmuş oluyor. Buda, aradığımız sayının ortada olması anlamına geliyor. Yani, Average Case => Eğer başından başlıyorsak O(n), diğer durumda O(3) yani O(1)’e denk geliyor.
* [7,3,5,8,2,9,4,15,6] dizisinin Insertion Sort’a göre ilk 4 adımını yazınız.
  + Sıralamaya ilk elemandan başlanır ve kendisinden sonraki elemanlarla karşılaştırılır ve arrayin en küçük elemanı varsa, onunla yer değiştirir.
    - [7,3,5,8,2,9,4,15,6] =>
      * 7 -> karşılaştırdığımız eleman ; 2 -> en küçük eleman
      * 7 > 2 => 2 ile 7’nin yerleri değiştirilir.
  + Sonraki elemana geçilir ve aynı işlem tekrarlanır. Seçilen elemandan sonraki elemanlarla karşılaştırılır ve arrayin diğer en küçük eleman varsa, onunla yer değiştirir.
    - [2,3,5,8,7,9,4,15,6] =>
      * 3 -> karşılaştırdığımız eleman ; 3 -> en küçük eleman
      * 3’ten en küçük eleman bulunamadığı için, herhangi bir değişiklik olmaz.
    - [2,3,5,8,7,9,4,15,6] =>
      * 5 -> karşılaştırdığımız eleman; 4 -> en küçük eleman
      * 5 > 4 => 2 ile 4’ün yerleri değiştirilir.
    - [2,3,4,8,7,9,5,15,6] =>
      * 8 -> karşılaştırdığımız eleman; 5 -> en küçük eleman
      * 8 > 5 => 5 ile 8’in yerleri değiştirilir.