Bölüm 8: Öncelikli Kuyruk

Veri Yapıları

Sercan KÜLCÜ, Tüm hakları saklıdır.

Öncelikli Kuyruk (Priority Queue)

- Öğeler öncelik sırasına göre saklanır.
- Öncelikli işlemlerin yönetiminde kullanılır.
- Kuyruktan en yüksek öncelikli öğeyi çıkarmak için O(1) zaman yeterlidir.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Temel Kavramlar

- **Öncelik Kuyruğu**: Öğelerin saklandığı yapı.
- **Öncelik**: Her öğeye atanan öncelik değeri.
- **En Yüksek Öncelik**: Kuyruğun başında bulunan düğümün öncelik değeri.
- **FIFO İlkesi**: Eşit öncelikteki öğeler arasındaki sıra.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Kullanım Alanları

- **İşletim Sistemleri**: Görev sıralamasında kullanılır.
- **Çizge Algoritmaları**: Dijkstra ve A* algoritmaları gibi.
- **Acil Durum Yönetimi**: Hasta sıralaması ve olay yönetimi.
- **Veri Sıkıştırma**: Huffman kodlaması.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Temel İşlemler

- **Ekleme (Insertion):** Öğe eklenirken konumu önceliğine göre bulunur.
- **Çıkarma (Extraction):** En yüksek öncelikli öğe çıkarılır.
- **Sorgulama (Peek):** Öncelikli öğeyi döndürür, kuyruktan çıkarmaz.
- **Boş mu (isEmpty):** Kuyruğun boş olup olmadığını söyler.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Dizi Temsili

- Öğeler basit bir şekilde dizide tutulur.
- Öncelikli öğe dizinin başında saklanır.
- Öğe ekleme ve çıkarma işlemlerinden sonra sıralama bozulabilir.
- Dizinin her işlemden sonra sıralı kalması zor ve karmaşık olabilir.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Bağlı Liste Temsili

- Öğeler bağlı liste yapısında saklanır.
- Öğeler önceliklerine göre bağlı listede uygun konuma eklenir.
- Öncelikli öğe listenin başında saklanır.

Sercan KÜLCÜ, Tüm hakları saklıdır.

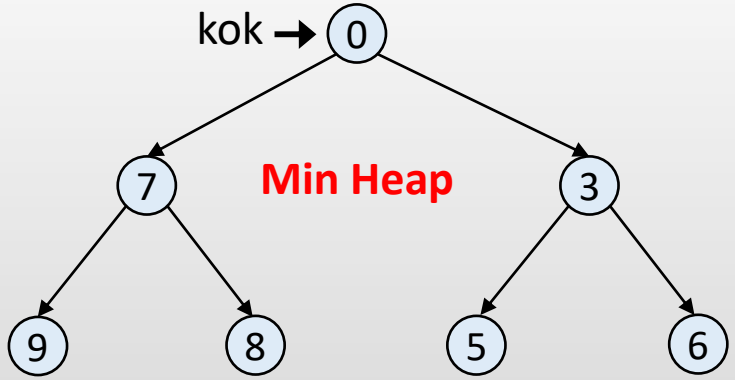
İkili Heap

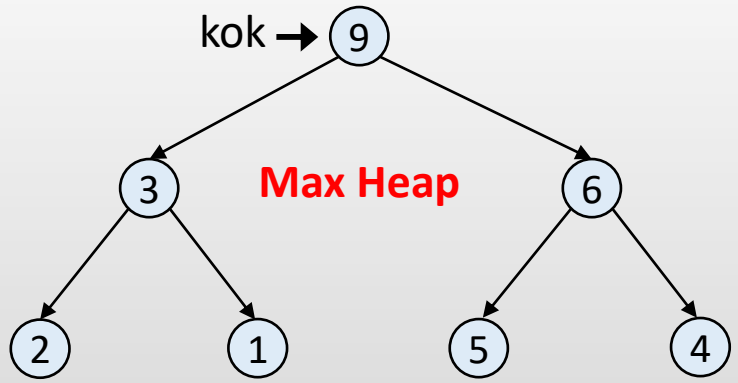
- İkili heap (min-heap veya max-heap) yaygın kullanılan bir veri yapısıdır.
- En yüksek öncelikli öğe kök düğümde bulunur.
- Öğe ekleme ve çıkarma işlemleri O(log n) zaman karmaşıklığına sahiptir.
- Thread-safe değildir.

Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap

- İkili Heap, özel bir ikili ağaç yapısıdır.
- Min-Heap ve Max-Heap olmak üzere iki türü vardır.
- **Min-Heap:** Kök düğümde en düşük öncelik değerine sahip öğe bulunur.
- **Max-Heap:** Kök düğümde en yüksek öncelik değerine sahip öğe bulunur.





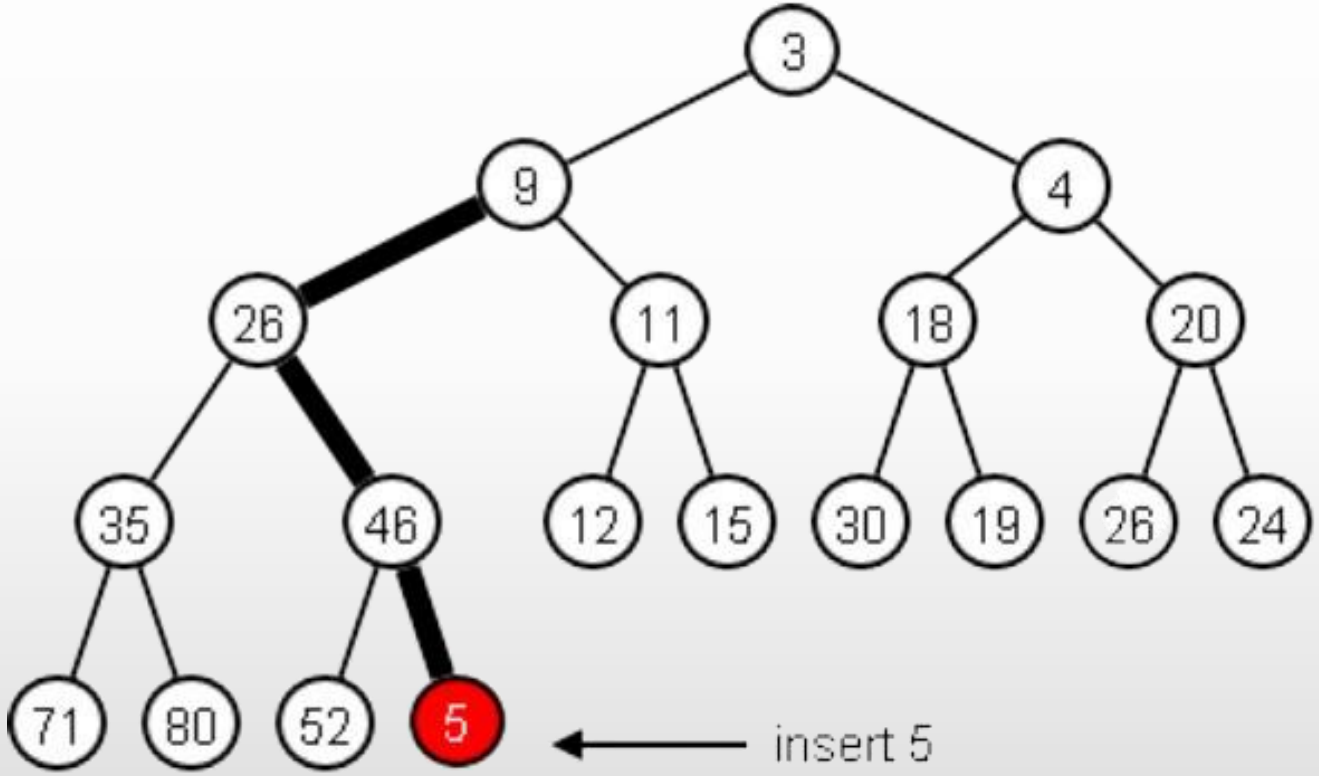
Sercan KÜLCÜ, Tüm hakları saklıdır.

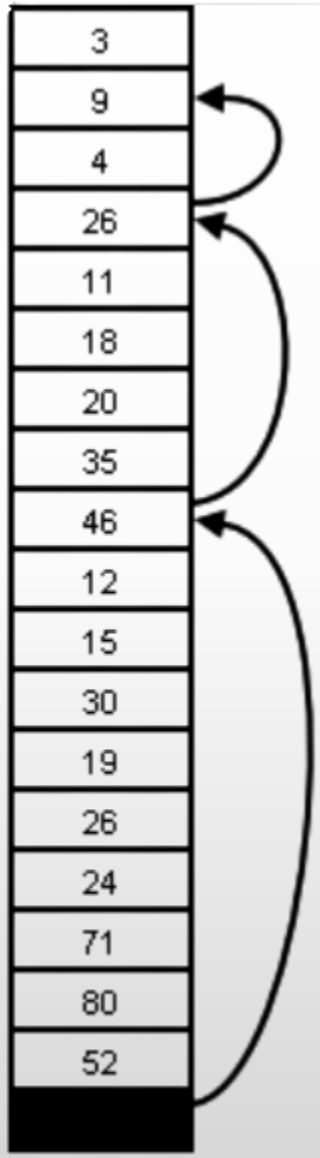
Öğe Ekleme

- Ağacın boşta olan ilk yaprak düğümüne öğe eklenir.
- Öğe ekledikten sonra, ağacın yapısı bozulabilir.
- Max-heap yapısında ata düğüm çocuklarından yüksek değere sahiptir.
- Ağacın tekrar dengelenmesi için "heapify" adı verilen bir işlem yapılır.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Öğe Ekleme





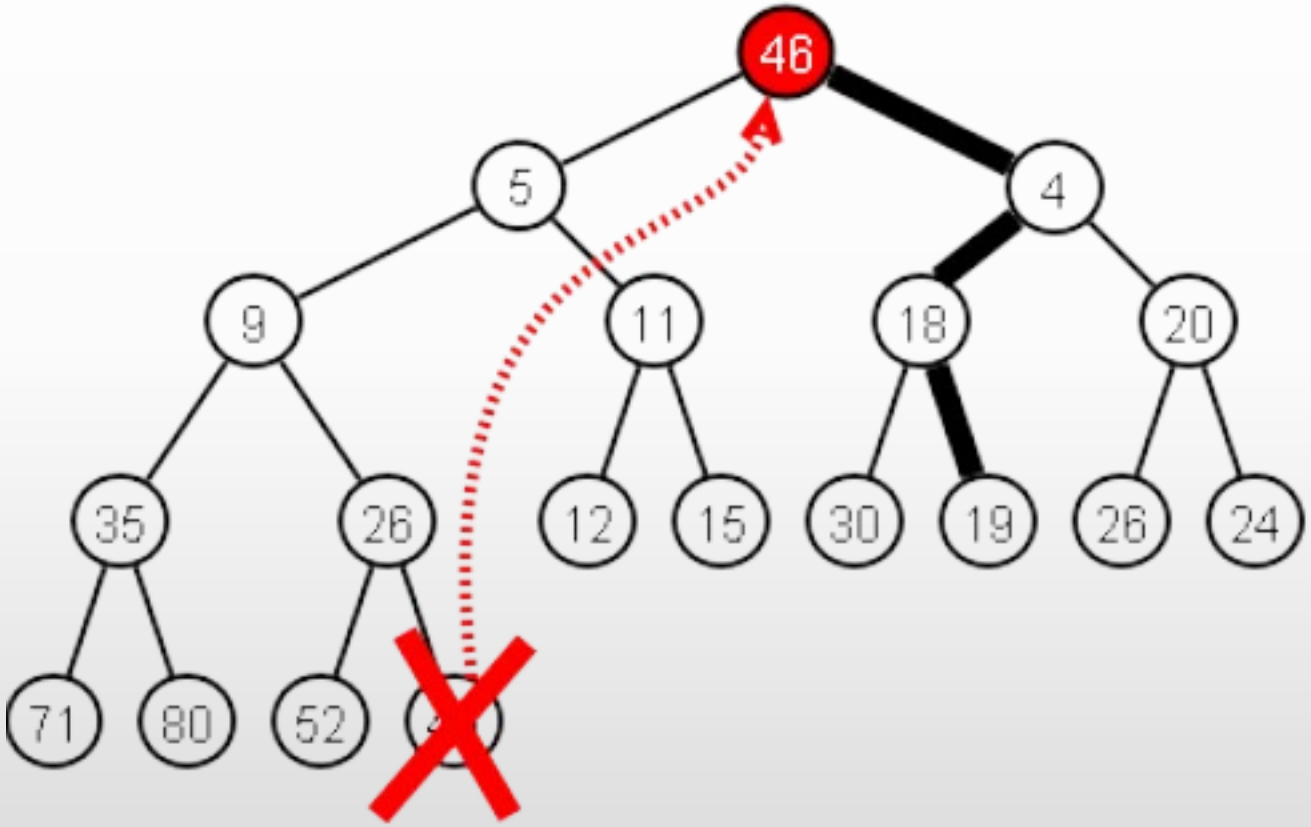
Sercan KÜLCÜ, Tüm hakları saklıdır.

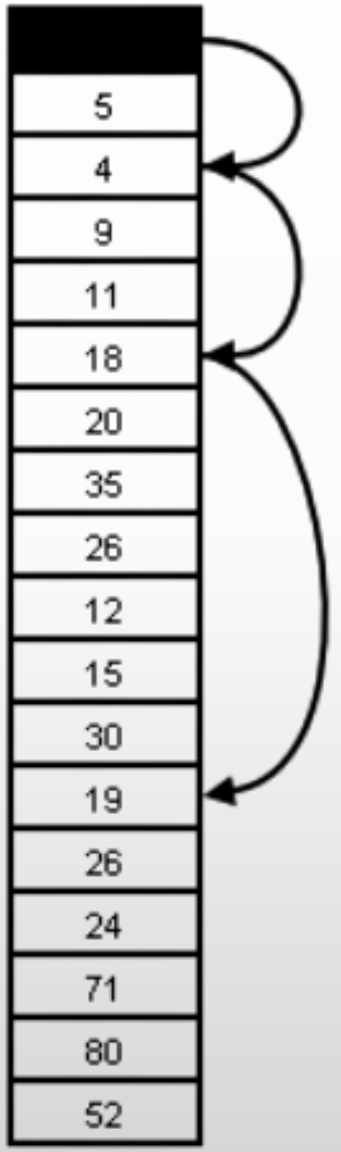
Öğe Çıkarma

- Kök düğümde bulunan öğe çıkarılır.
- Ağacın boş olmayan son yaprak düğümü kök'e taşınır.
- Bu işlemden sonra ağacın yapısı bozulabilir.
- Max-heap yapısında ata düğüm çocuklardan yüksek değere sahiptir.
- Ağacın tekrar dengelenmesi için "heapify" adı verilen bir işlem yapılır.
- heapify işlemi O(log n) zaman karmaşıklığına sahiptir.

Sercan KÜLCÜ, Tüm hakları saklıdır.

Öğe Çıkarma



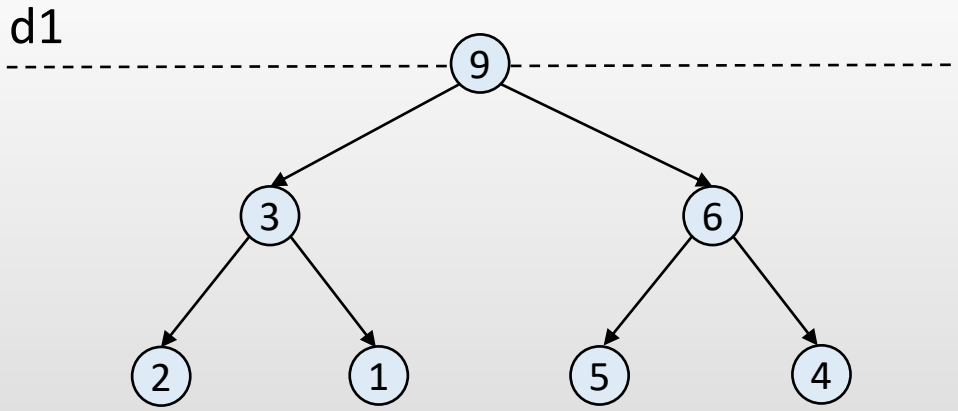


Sercan KÜLCÜ, Tüm hakları saklıdır.

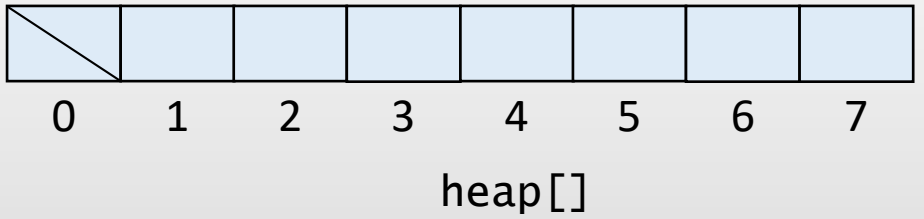
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



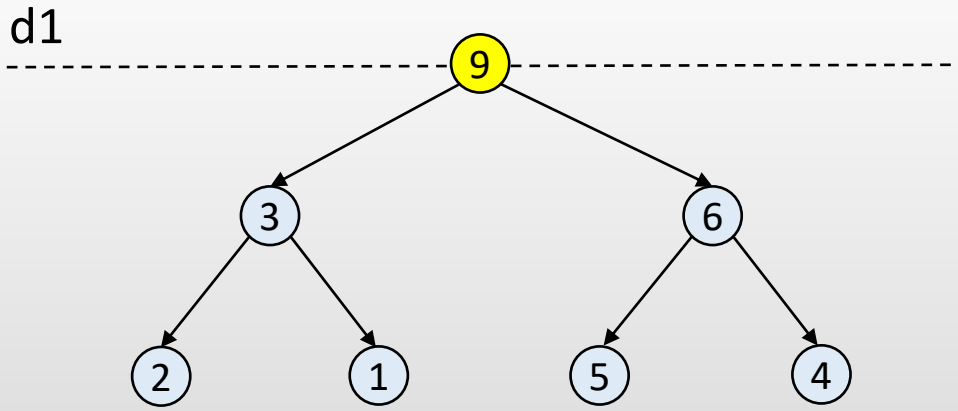
Max Heap



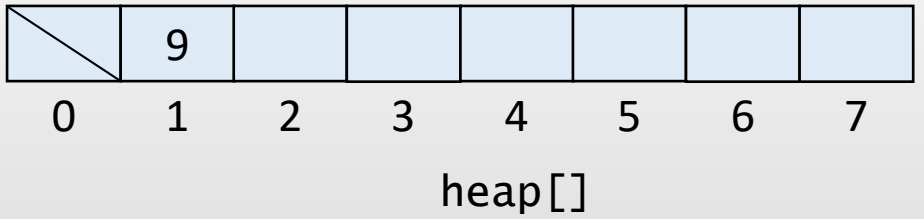
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



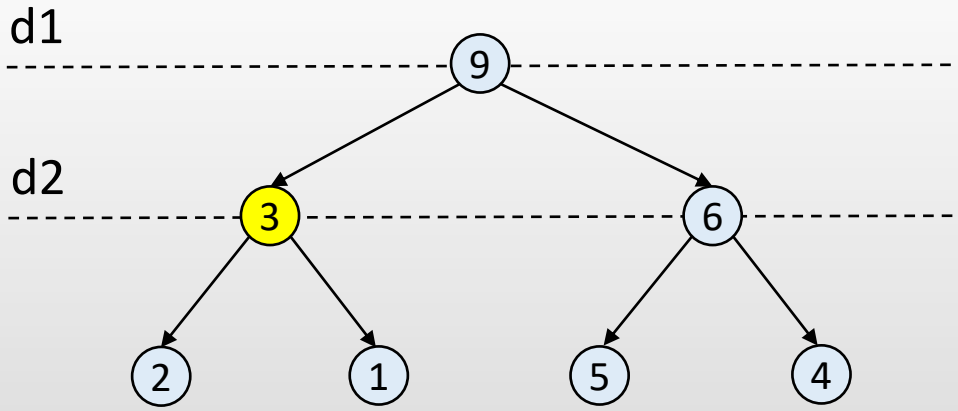
Max Heap



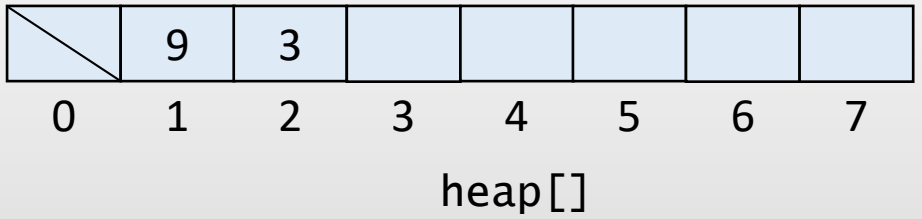
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



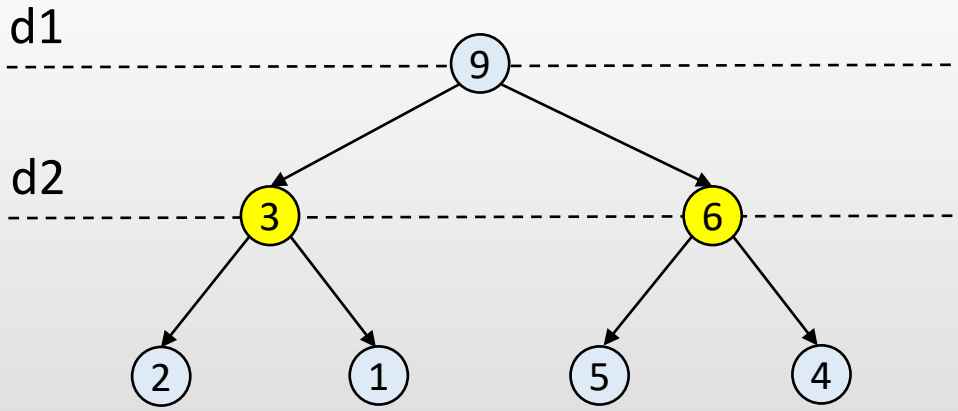
Max Heap



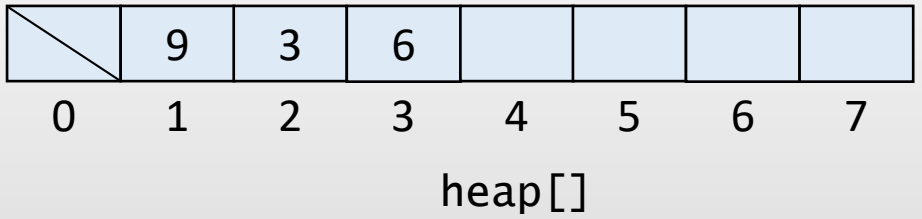
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



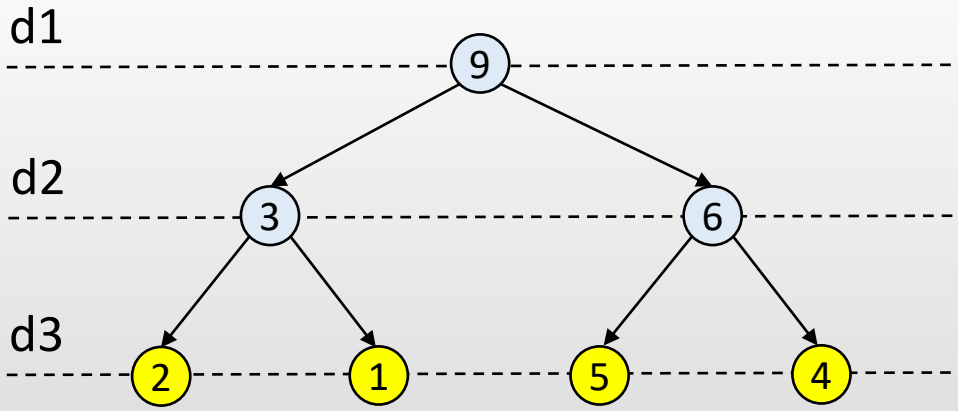
Max Heap



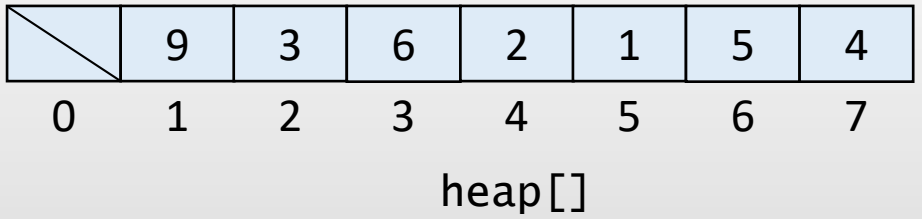
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



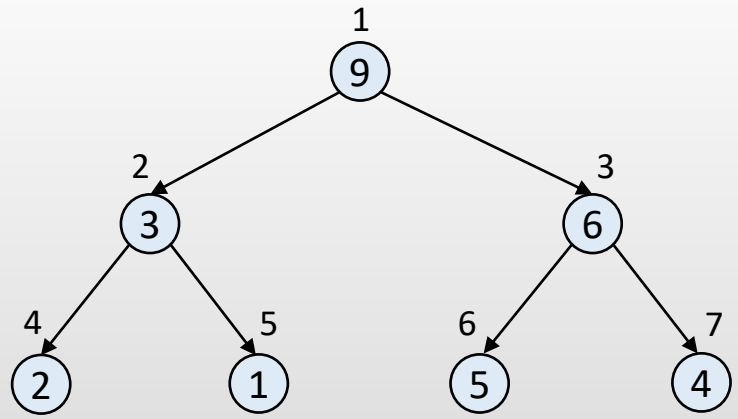
Max Heap



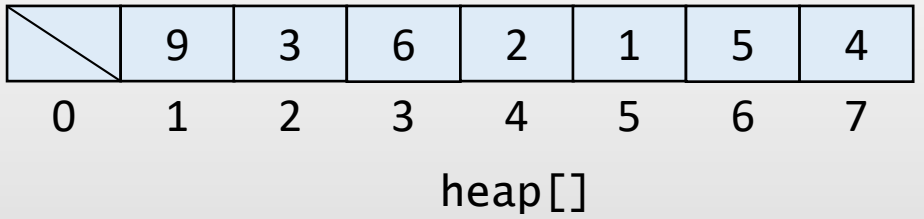
Sercan KÜLCÜ, Tüm hakları saklıdır.

İkili Heap Gösterimi

- Dizinin ilk elemanı boş bırakılır. Heap, tam ikili ağaçtır. Değerler soldan sağa düzey ağaç dolaşımı ile dizi içinde saklanır.



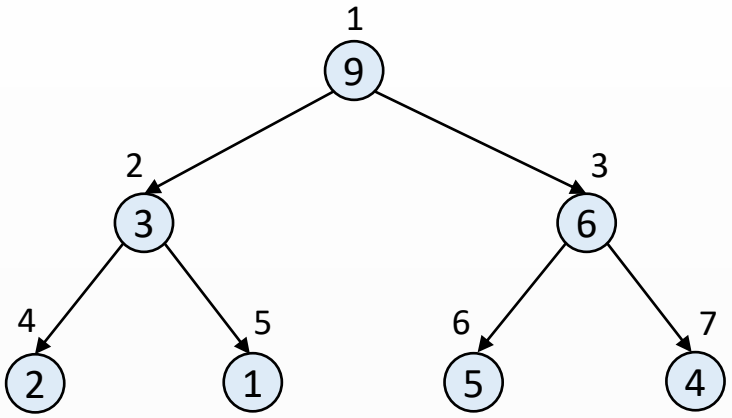
Max Heap



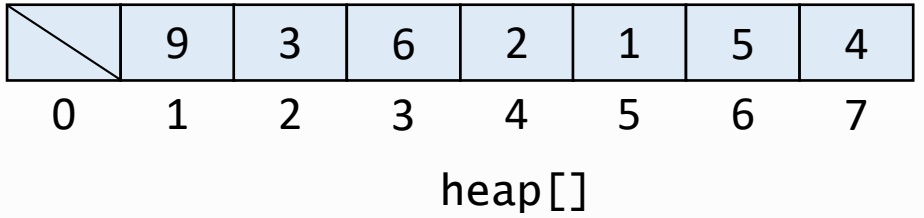
Sercan KÜLCÜ, Tüm hakları saklıdır.

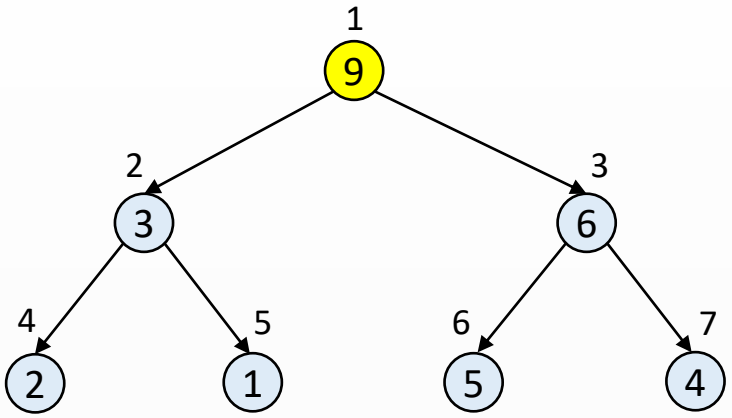
Ata ve Çocuk Hesaplamaları

Sercan KÜLCÜ, Tüm hakları saklıdır.

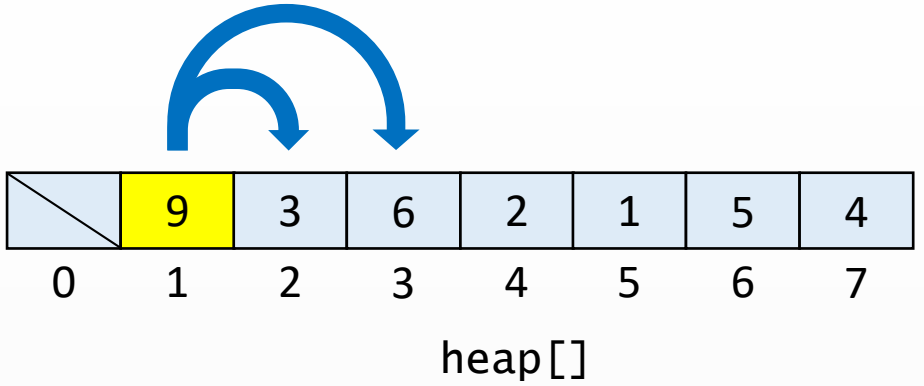


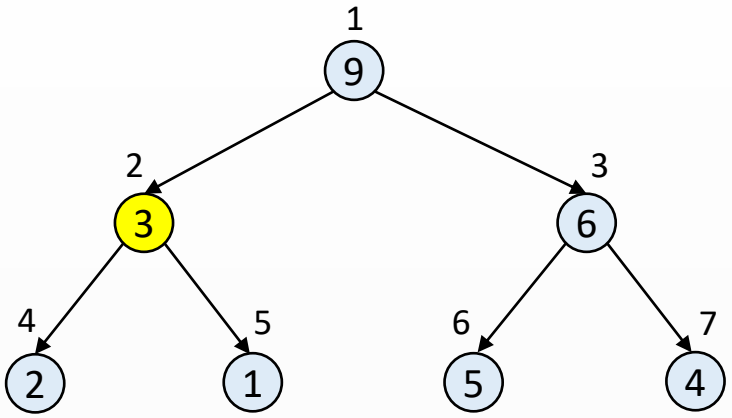
Max Heap



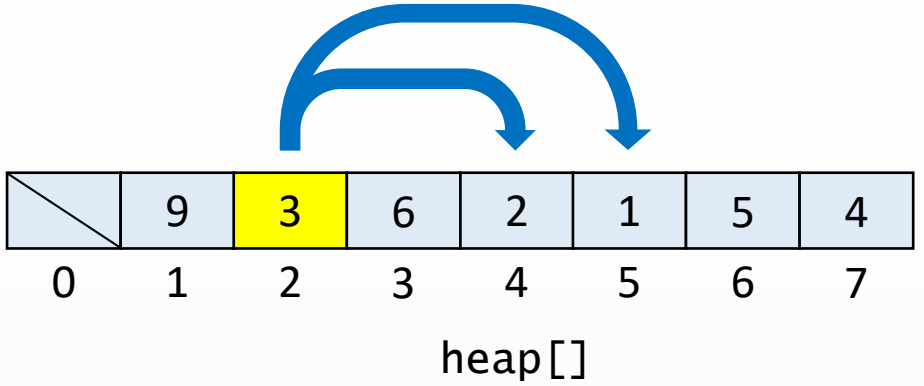


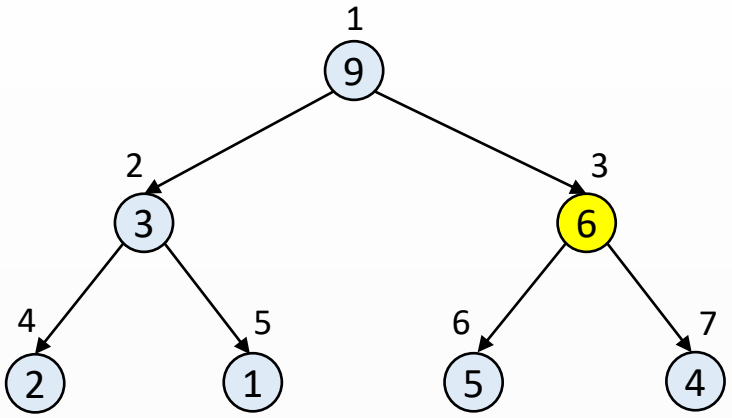
Max Heap



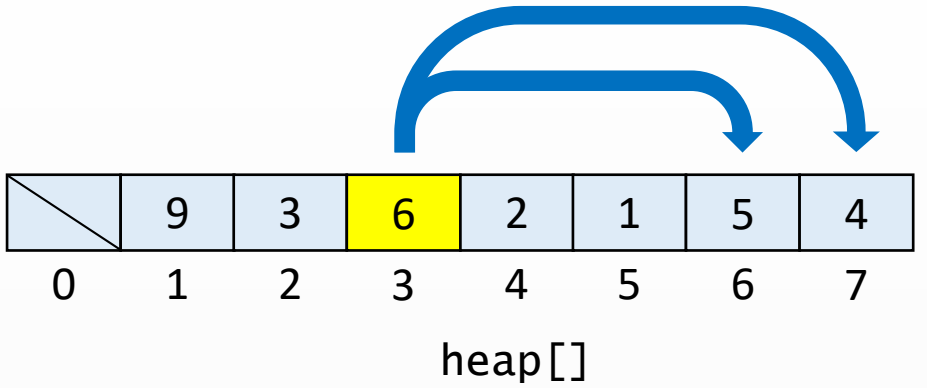


Max Heap





Max Heap



Çocuklar:

indeks 1 → 2, 3

indeks 2 → 4, 5

indeks 3 → 6, 7

indeks k → 2*k, 2*k + 1

Ata:

indeks 7 → ⌊ 7/2 ⌋ = 3

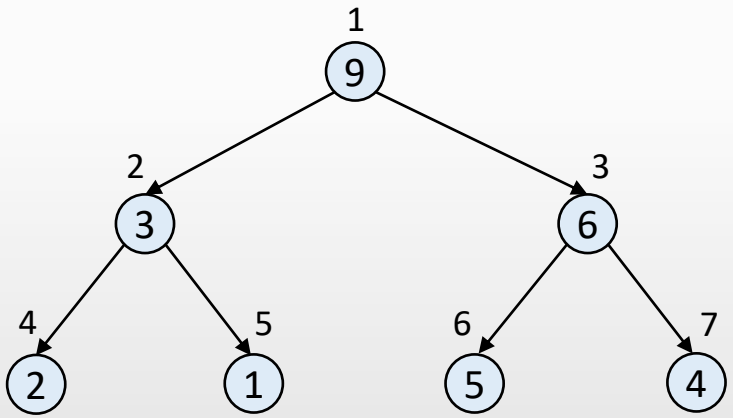
indeks 6 → ⌊ 6/2 ⌋ = 3

indeks 5 → ⌊ 5/2 ⌋ = 2

indeks k → ⌊ k/2 ⌋

İkili Max Heap Ağacı

- Her bir düğümün değeri, çocuklarının değerinden büyüktür.
- En büyük değer kök düğümde bulunur. Kök düğümün indeksi 1'dir.



Max Heap

	9	3	6	2	1	5	4
0	1	2	3	4	5	6	7

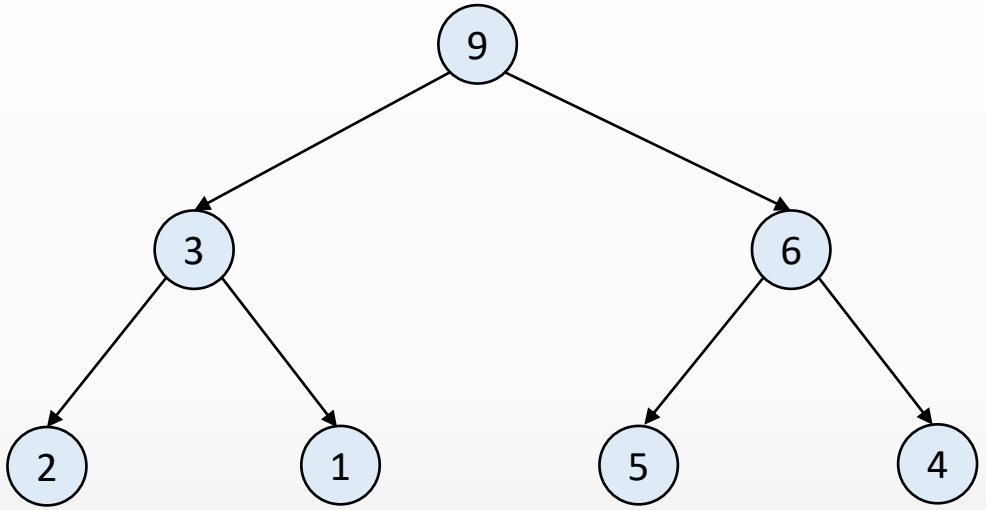
heap[]

Sercan KÜLCÜ, Tüm hakları saklıdır.

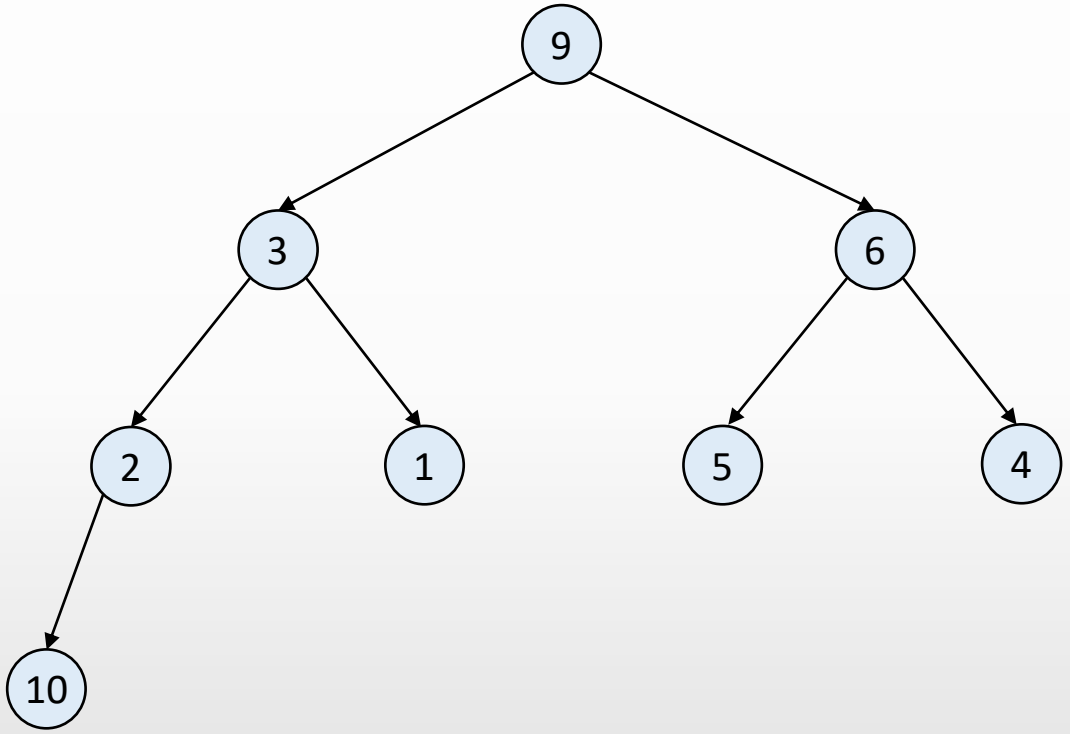
Aşağıdan Yukarıya Heap Ağacına Dönüştürme

- Max heap ikili ağacının her bir düğümünün değeri, çocuklarının değerlerinden büyüktür.
- Heap ağacına bir öğe eklendikten sonra bu özellik bozulabilir.
- Bu nedenle öğelerin yerlerinin değiştirilmesi gerekir.
- Ağaç aşağıdan yukarıya doğru taranarak yeniden heap ağacına dönüştürme işlemi (yüzdür - swim) uygulanır (bottom-up heapify).

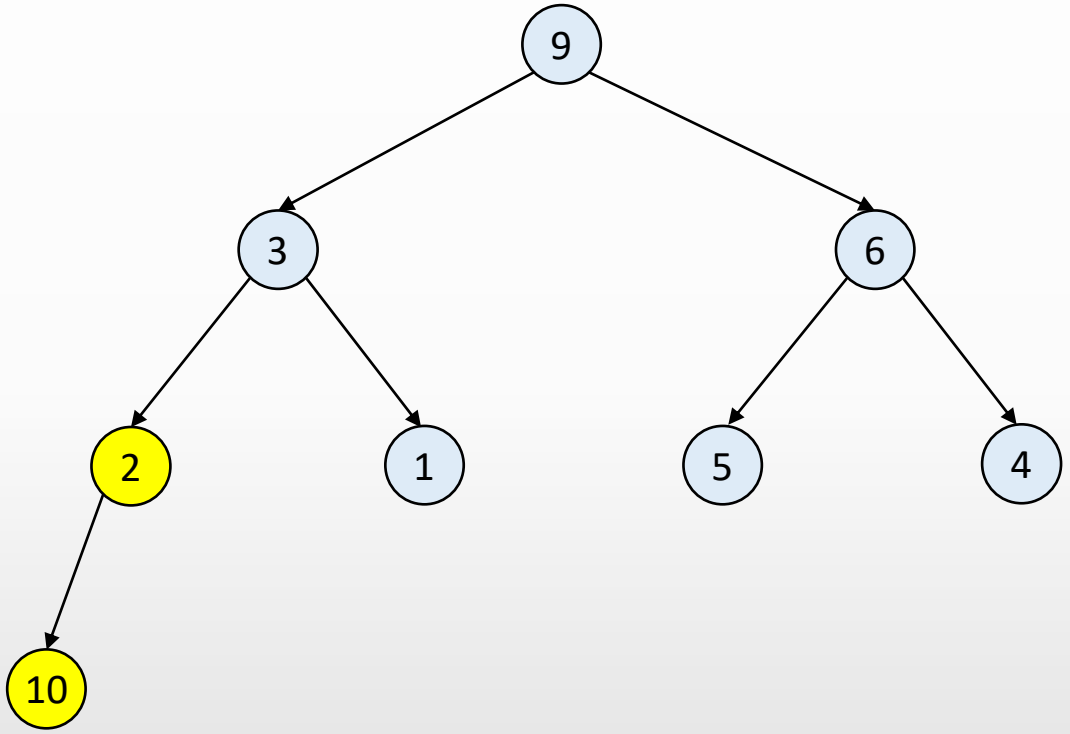
Sercan KÜLCÜ, Tüm hakları saklıdır.



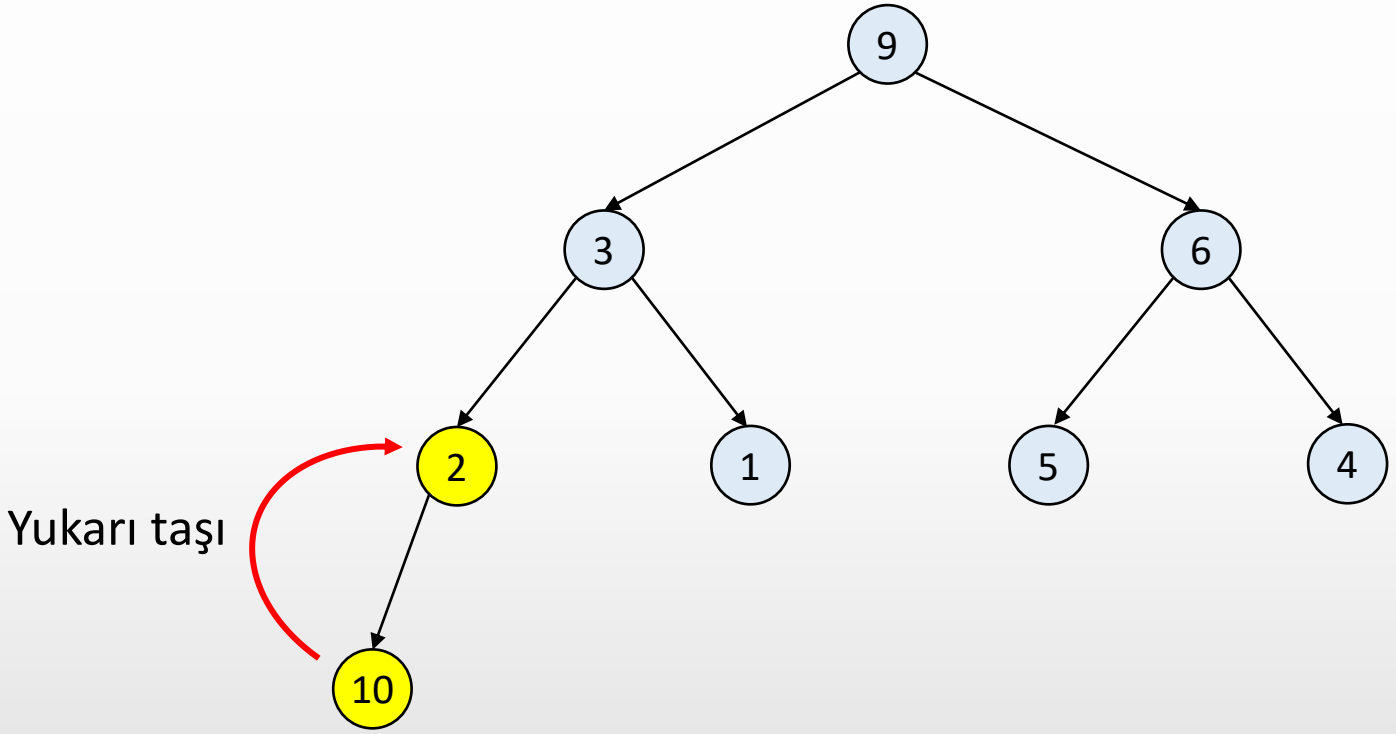

```
ekle(10)
```



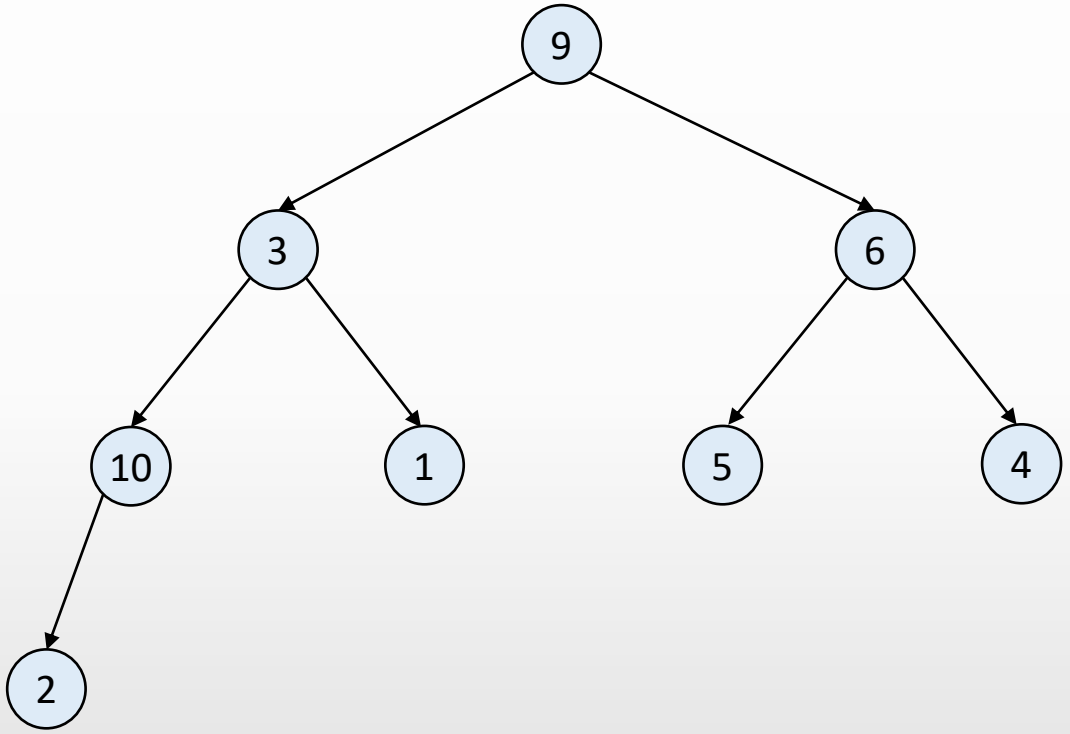

```
ekle(10)
```



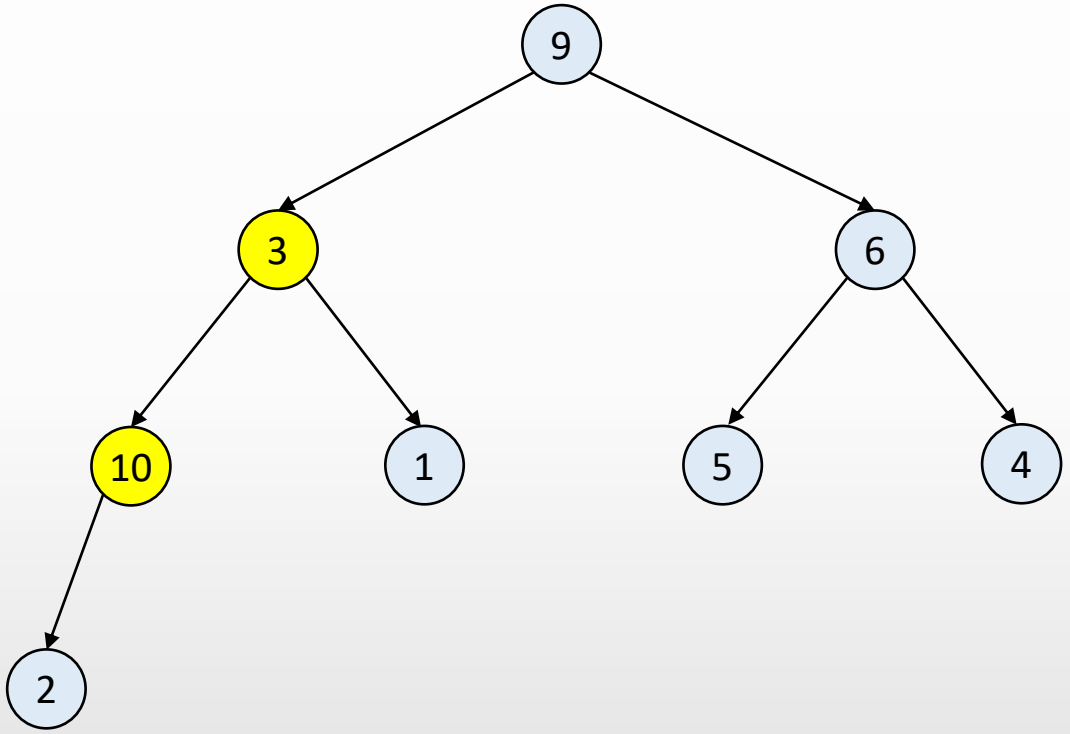

```
ekle(10)
```



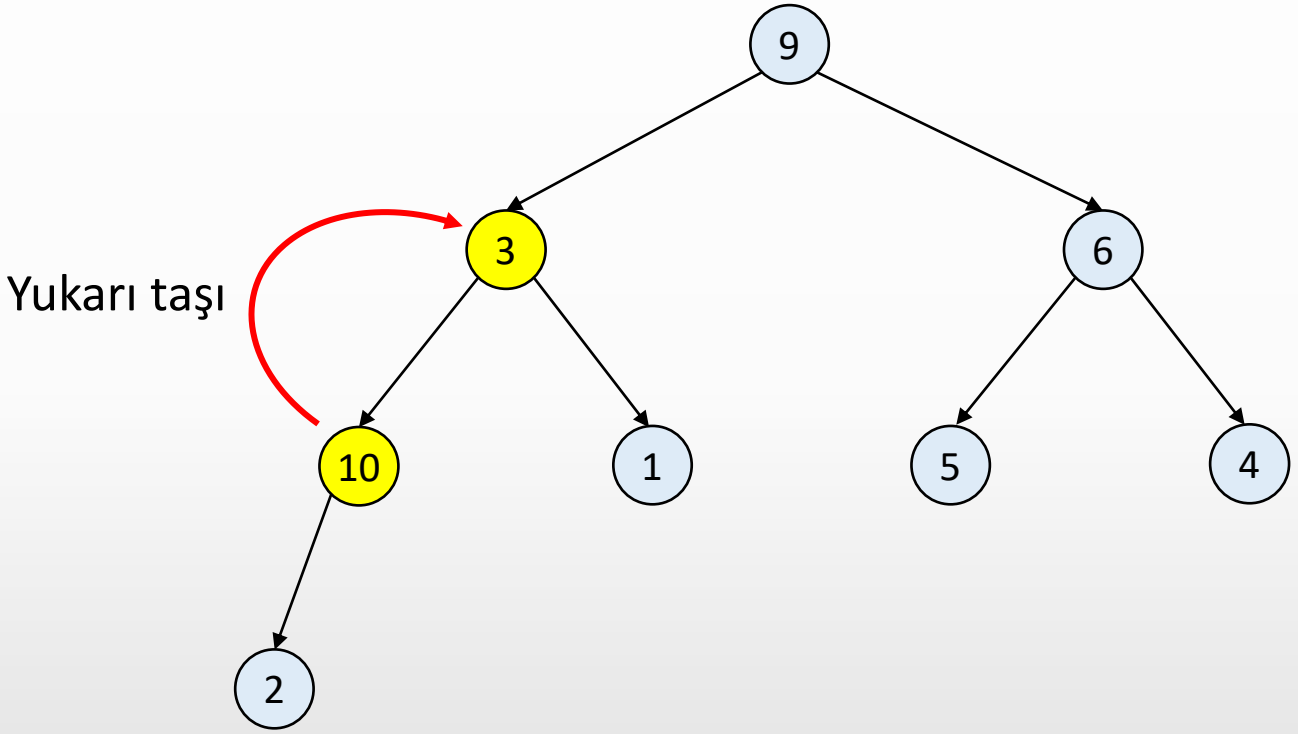

```
ekle(10)
```



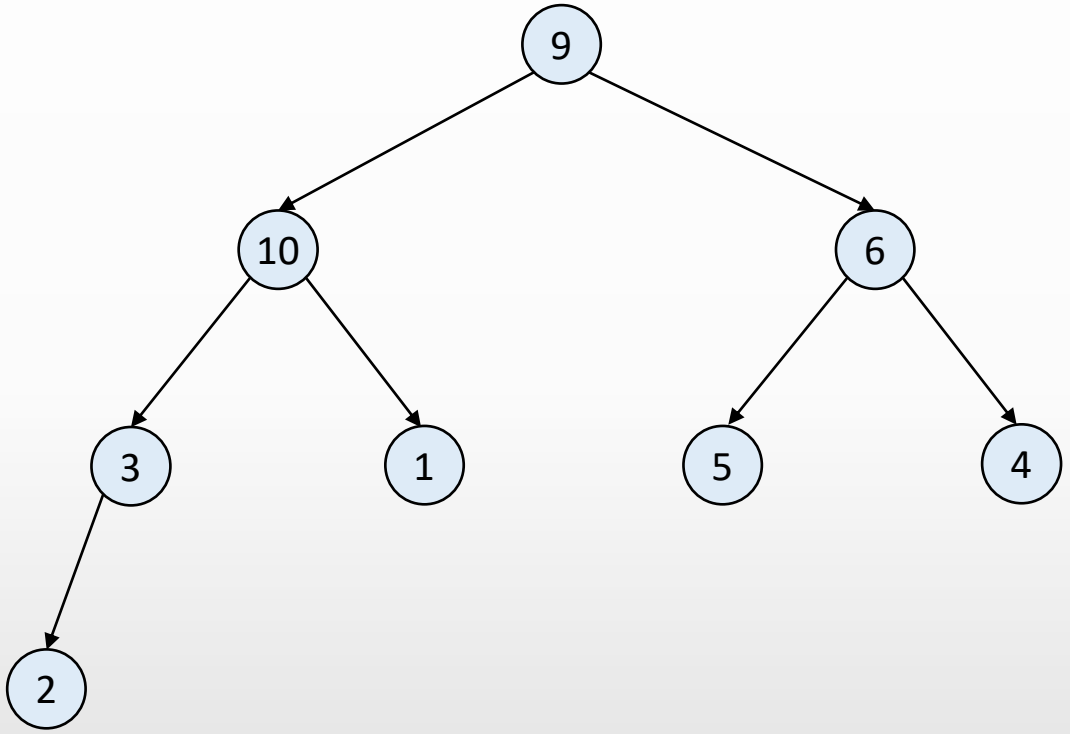

```
ekle(10)
```



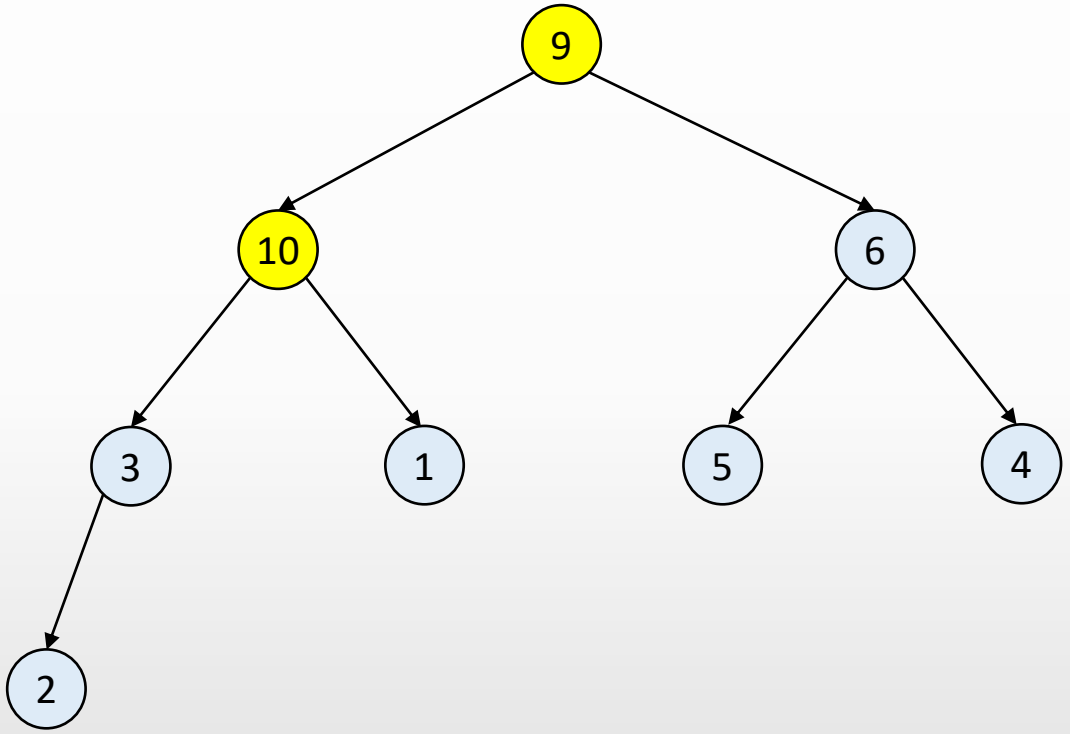

```
ekle(10)
```



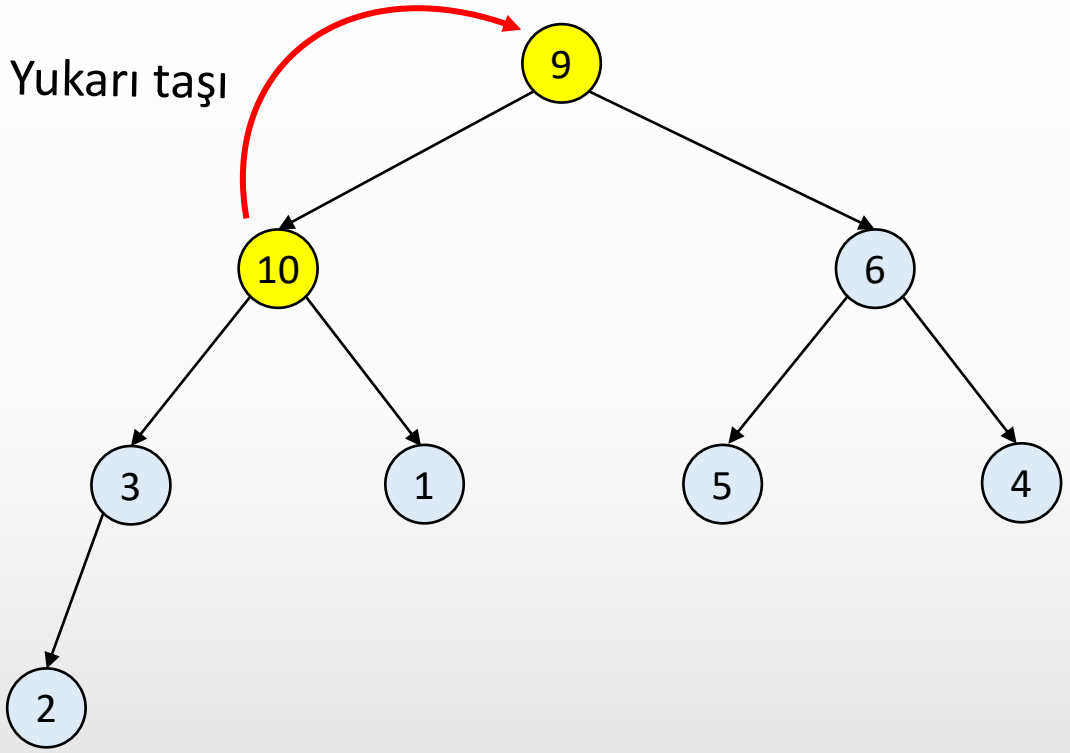

```
ekle(10)
```



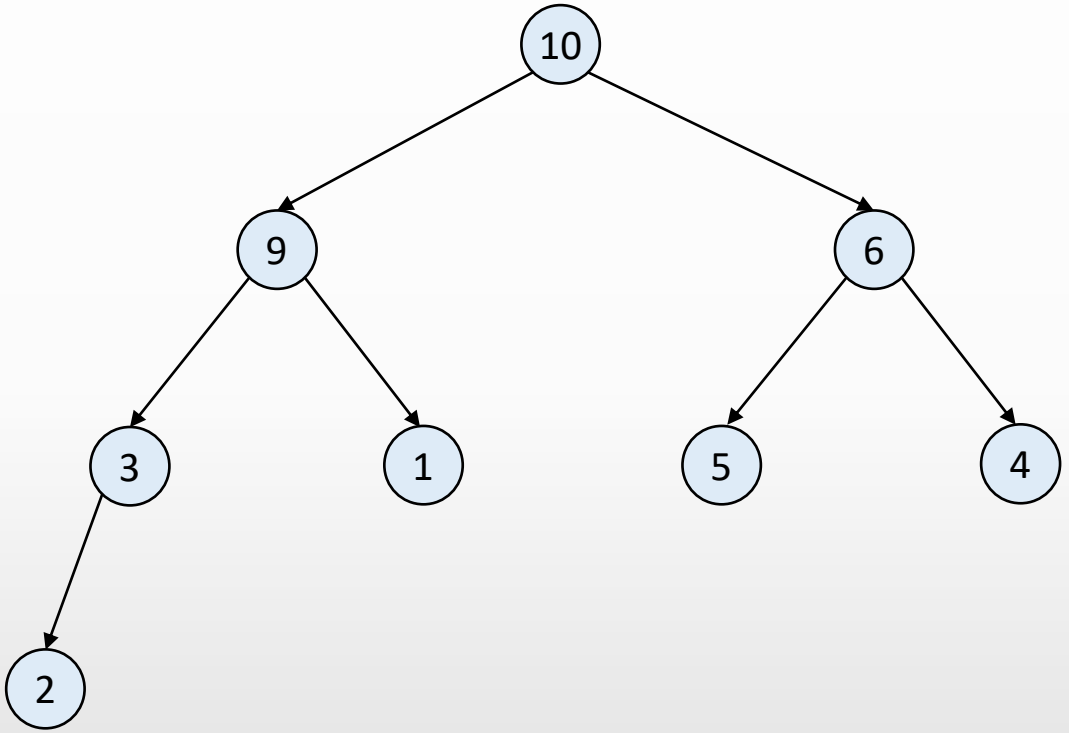

```
ekle(10)
```

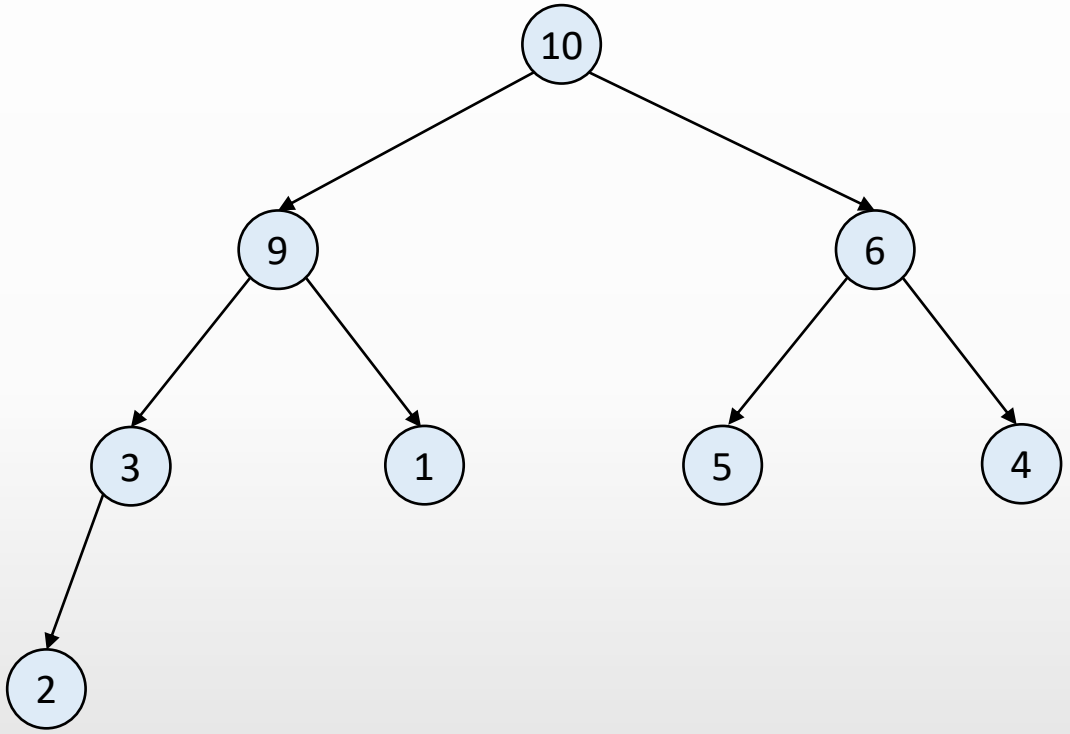



```
ekle(10)
```

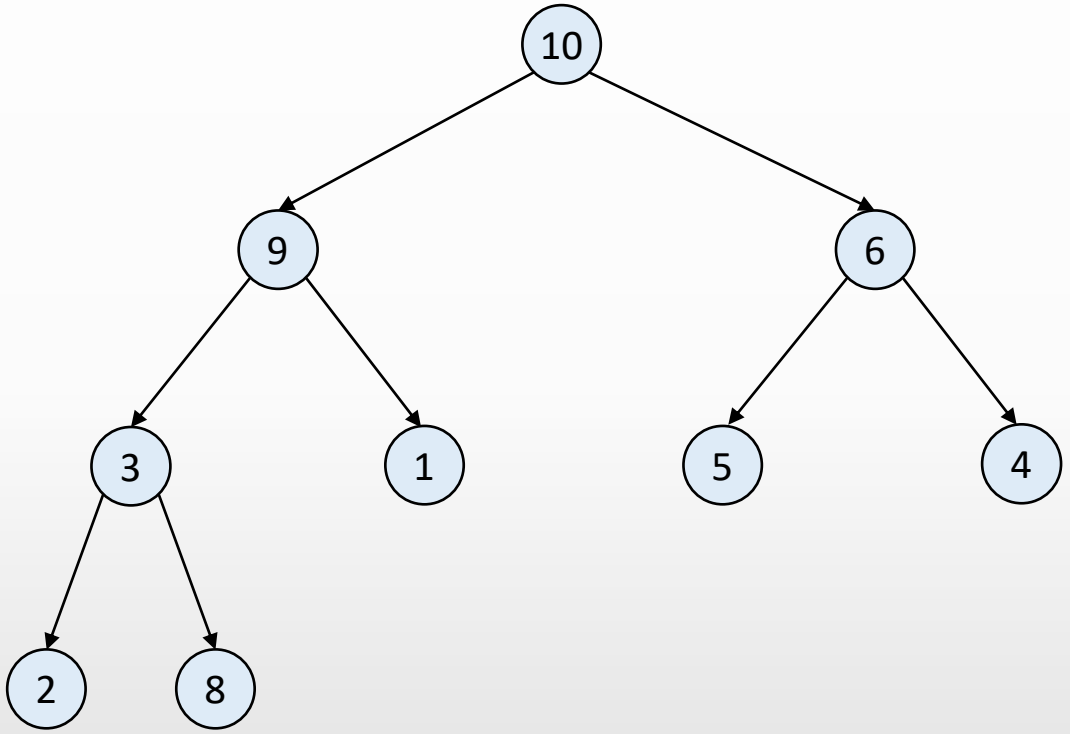



```
ekle(10)
```

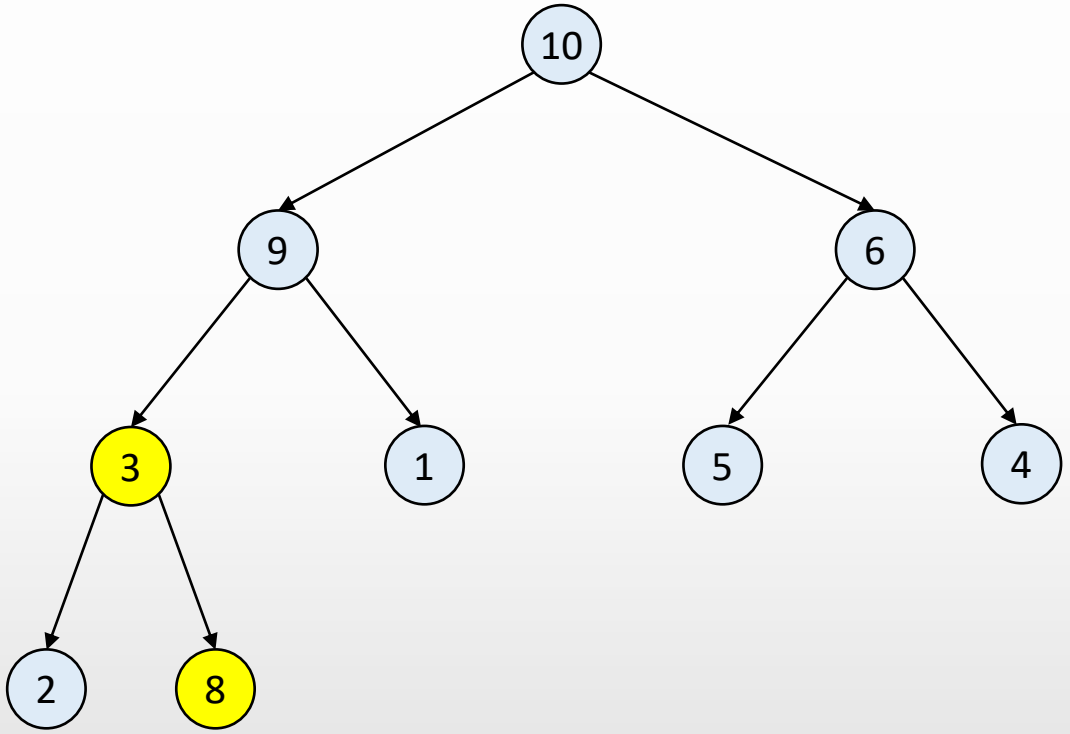

10
9
6
3
1
5
4
2



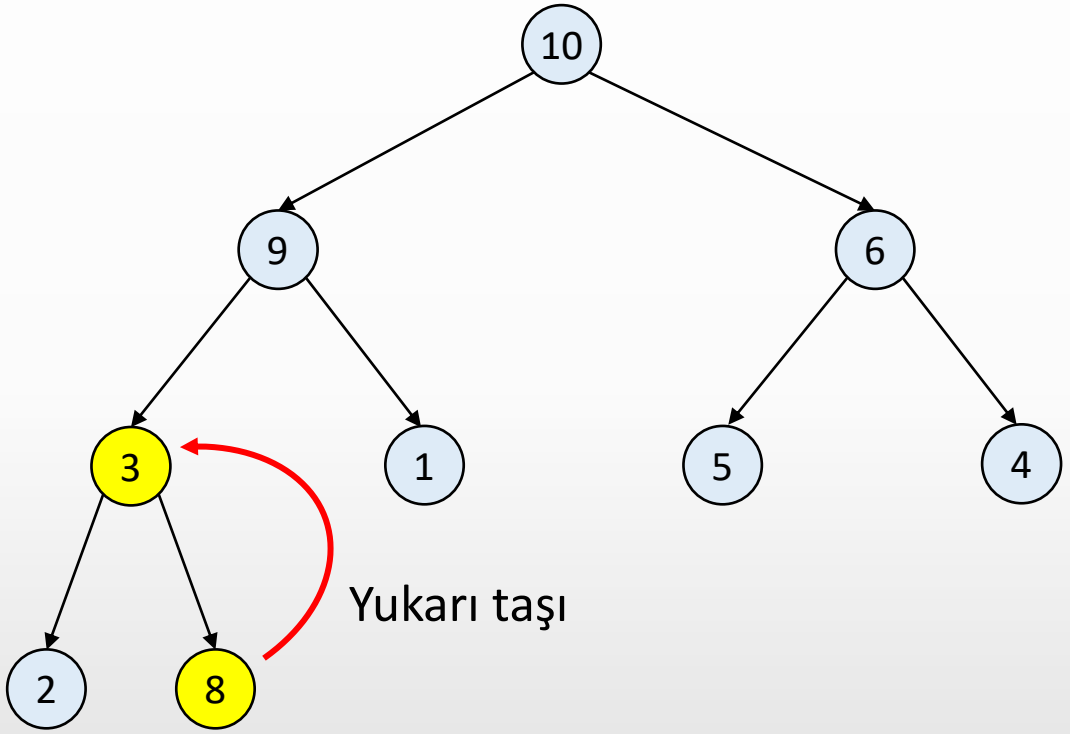

```
ekle(8)
```



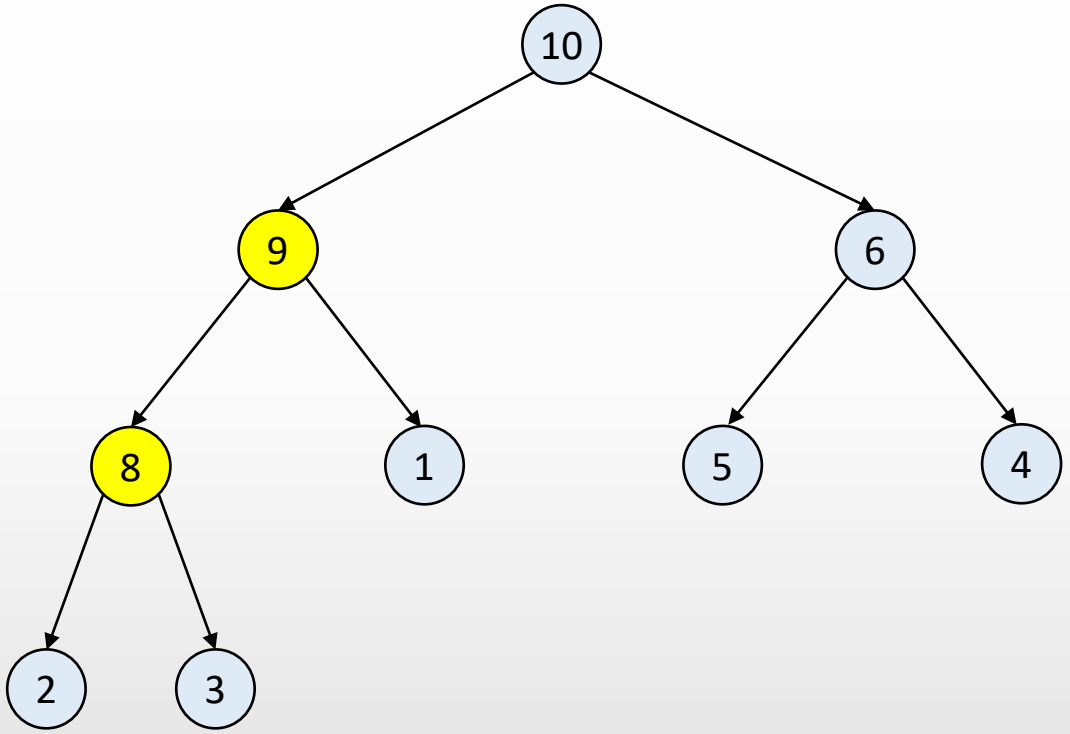

```
ekle(8)
```




```
ekle(8)
```




```
ekle(8)
```




```
ekle(8)
```

Sercan KÜLCÜ, Tüm hakları saklıdır.

Aşağıdan Yukarıya Heap Ağacına Dönüştürme

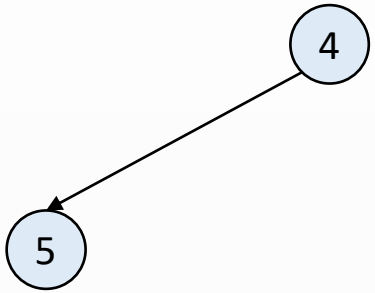
```
ekle(4)
```

4

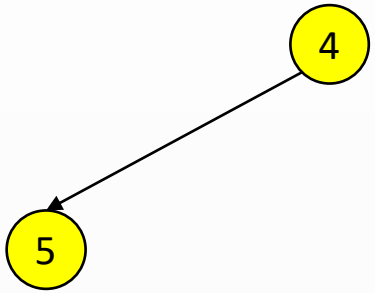
```
ekle(4)
```

4

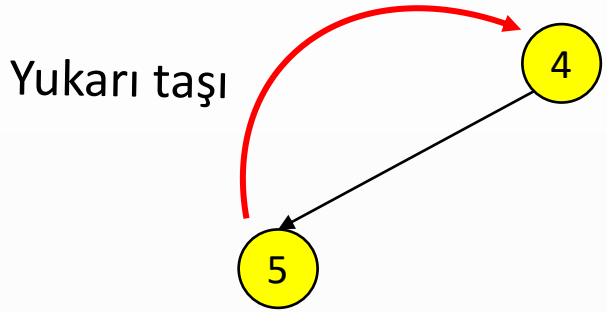
```
ekle(5)
```



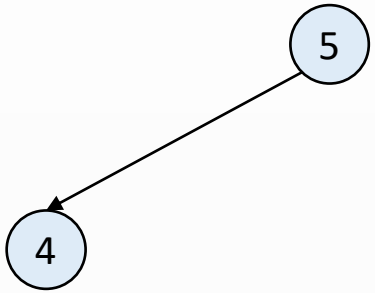

```
ekle(5)
```



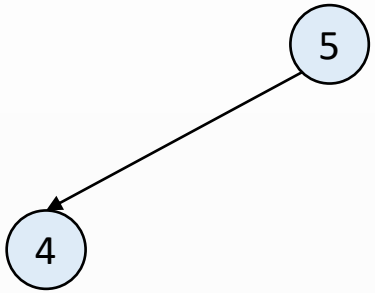

```
ekle(5)
```



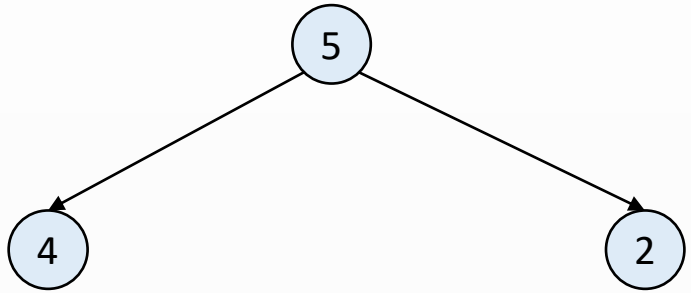

```
ekle(5)
```



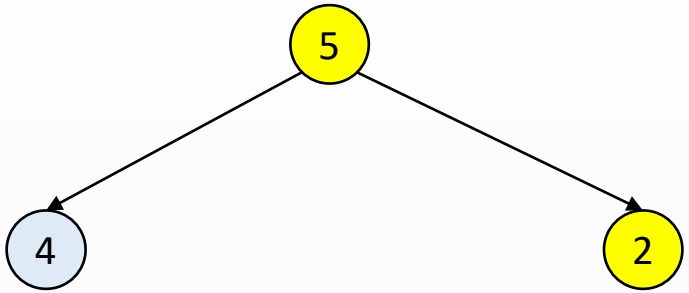

```
ekle(5)
```



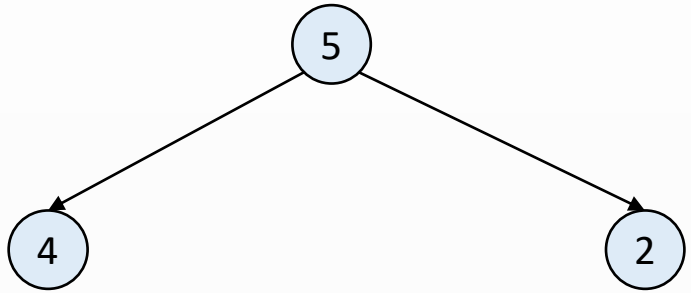

```
ekle(2)
```



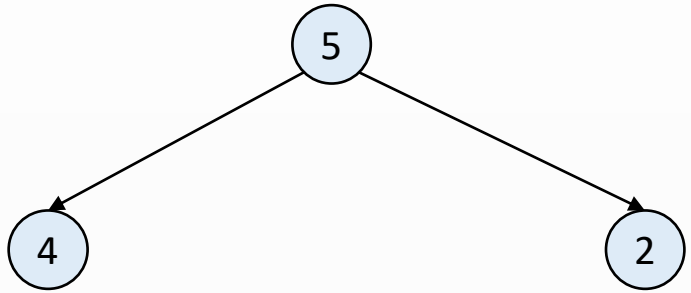

```
ekle(2)
```



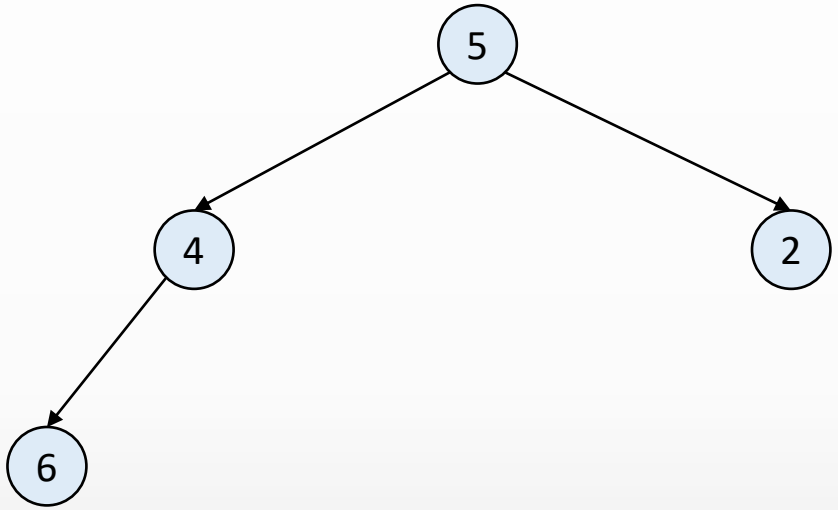

```
ekle(2)
```



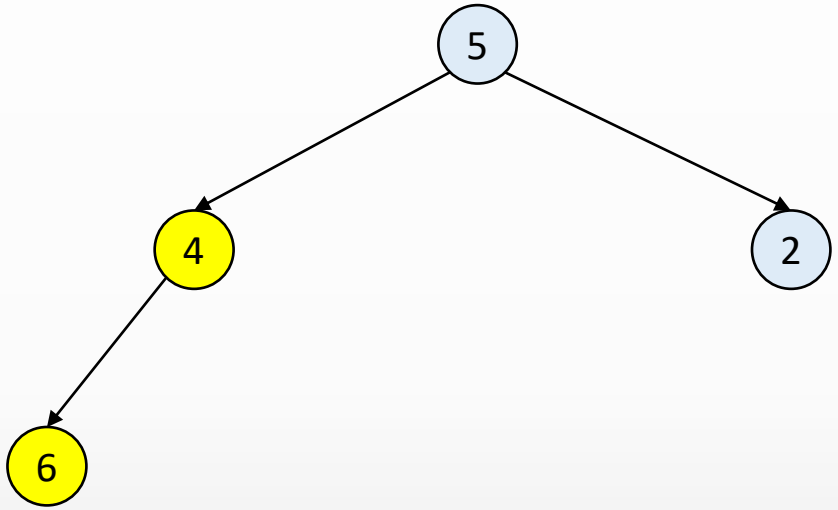

```
ekle(2)
```



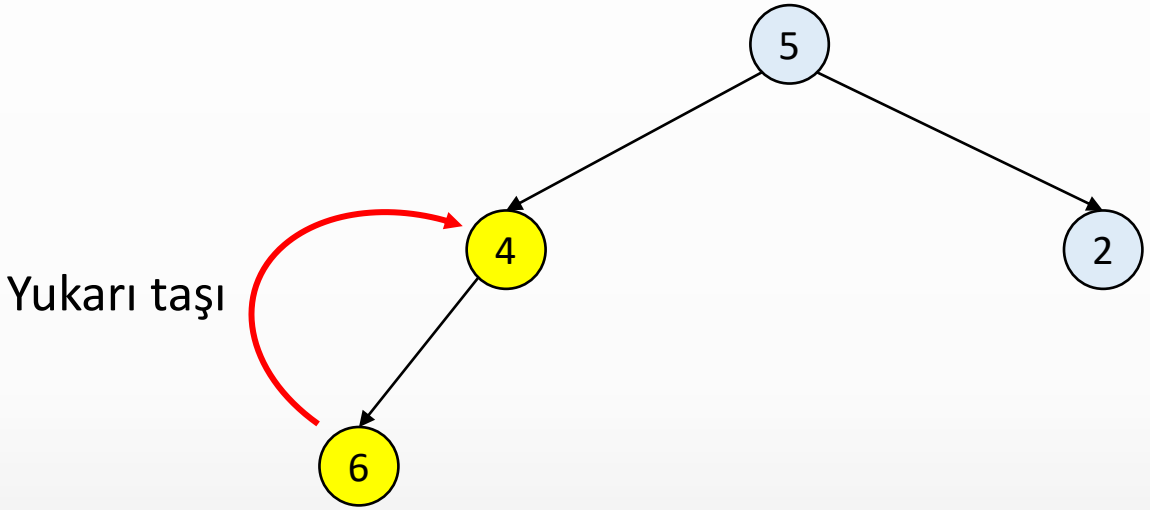

```
ekle(6)
```



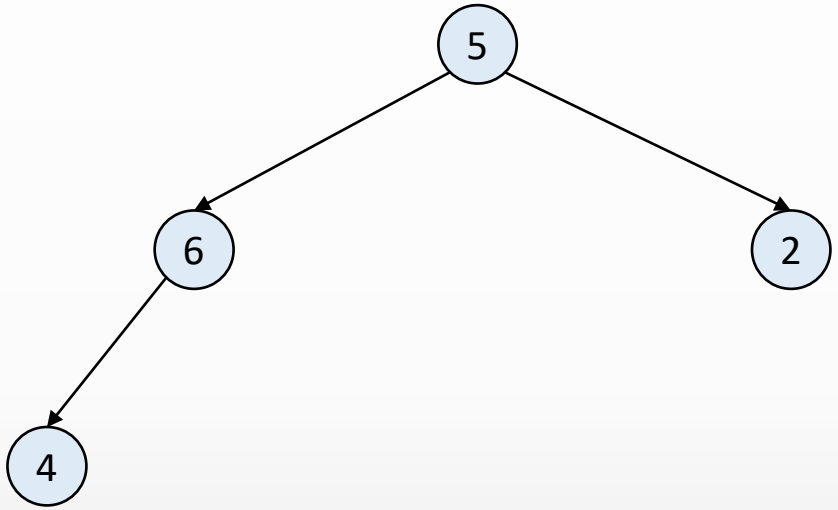

```
ekle(6)
```



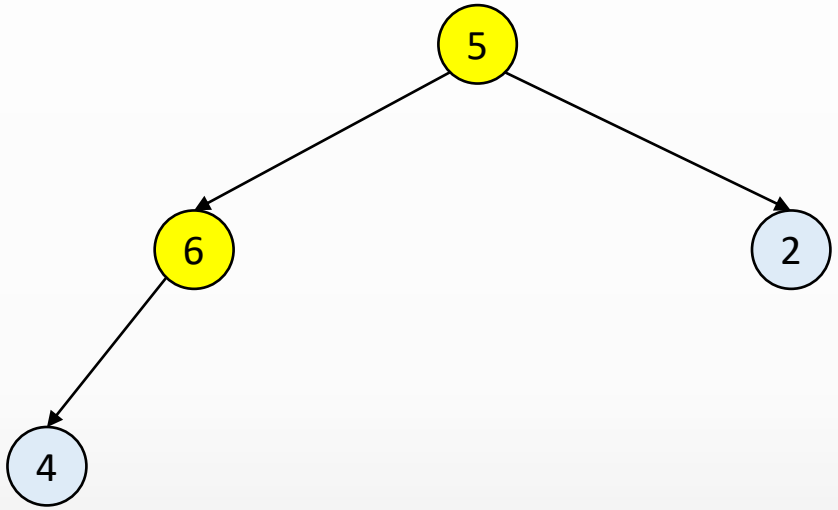

```
ekle(6)
```



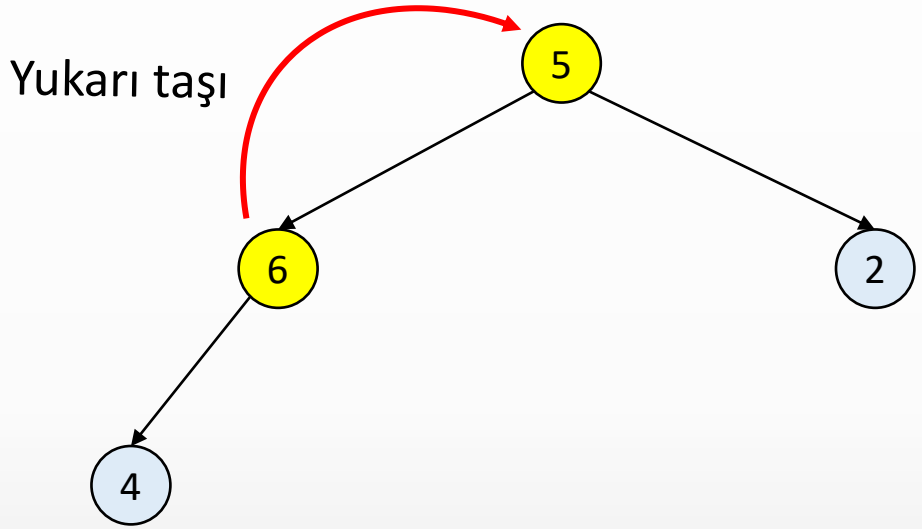

```
ekle(6)
```



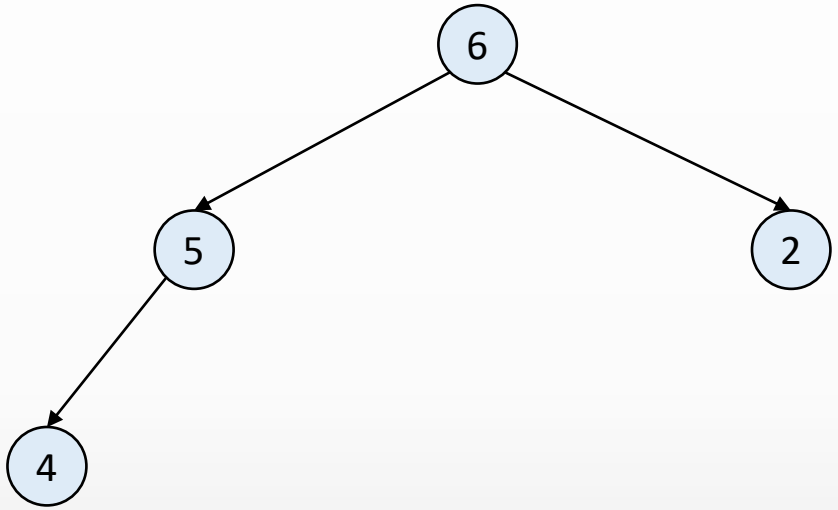

```
ekle(6)
```



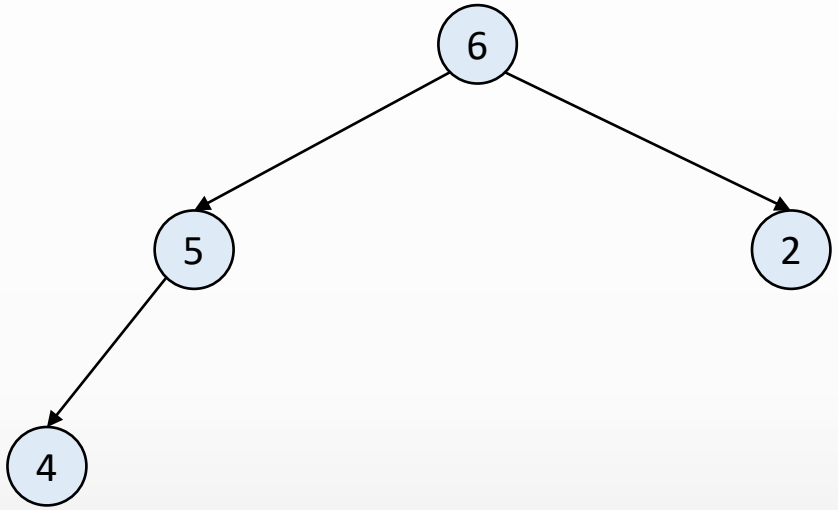

```
ekle(6)
```



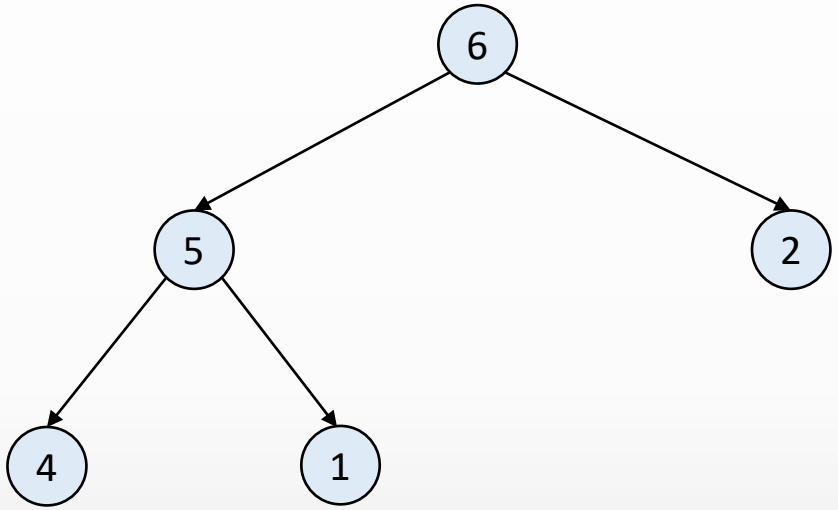

```
ekle(6)
```



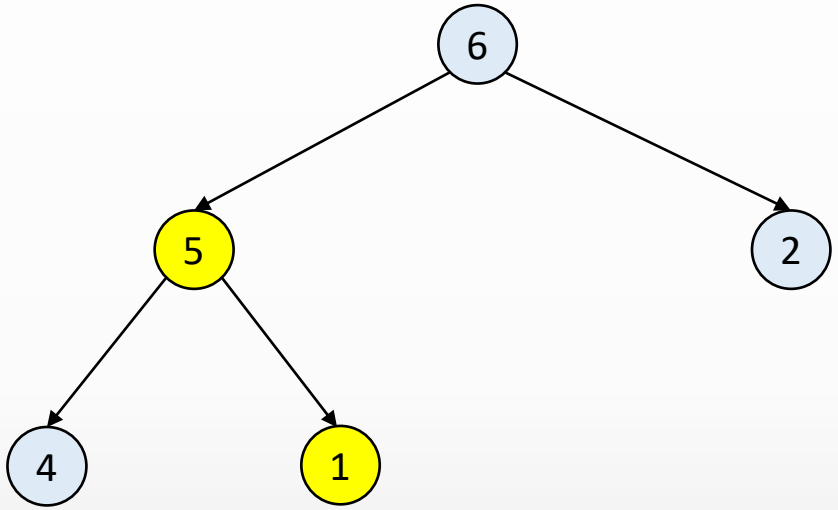

```
ekle(6)
```



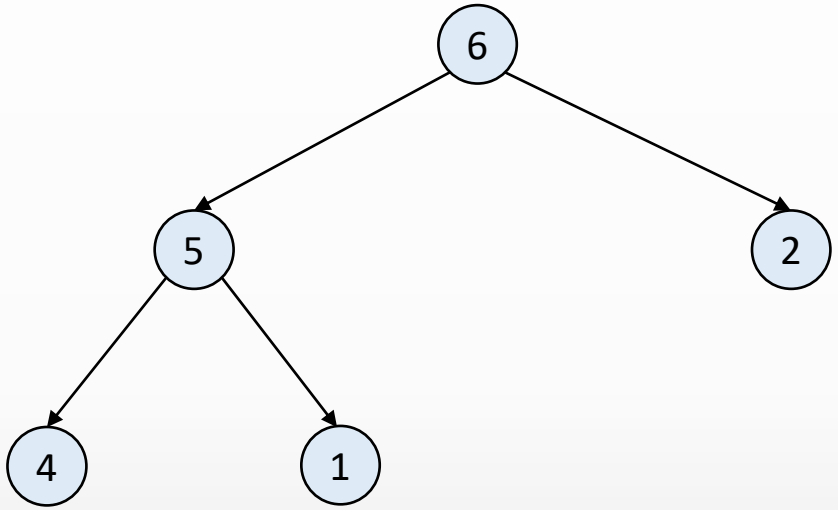

```
ekle(1)
```



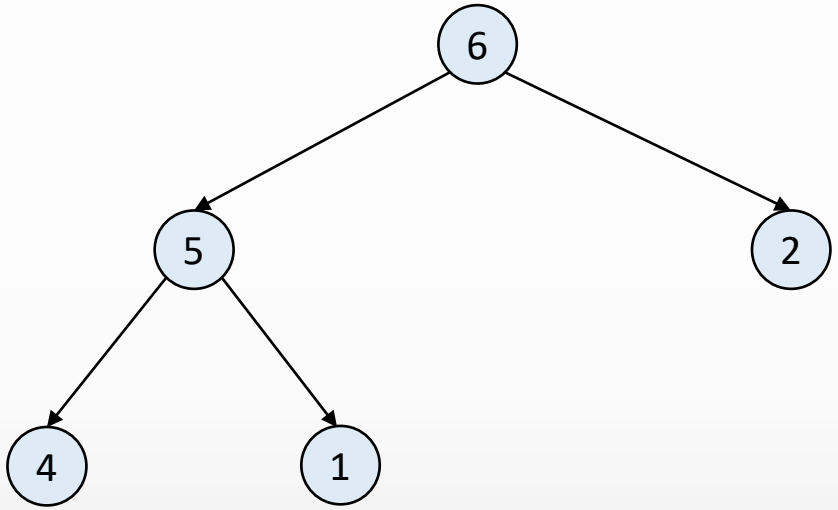

```
ekle(1)
```



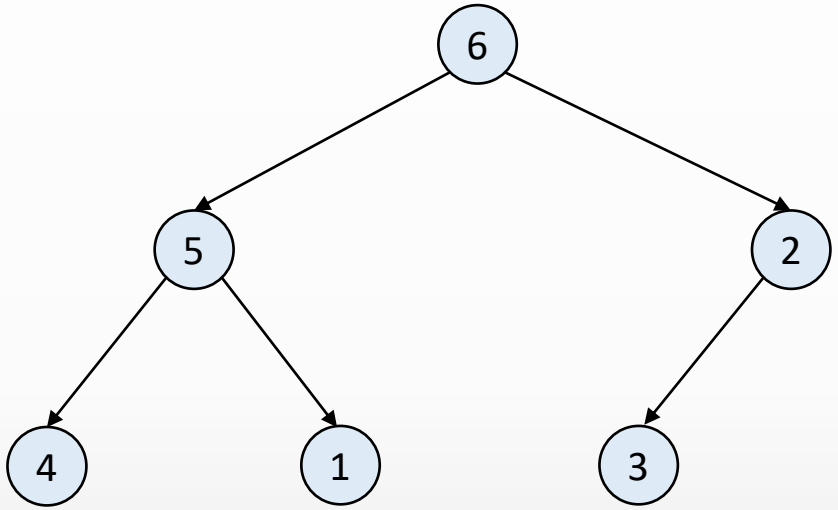

```
ekle(1)
```



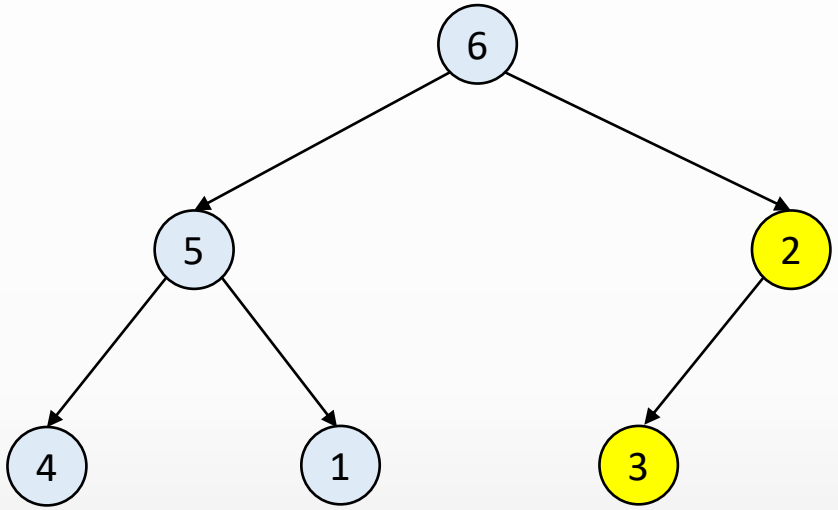

```
ekle(1)
```



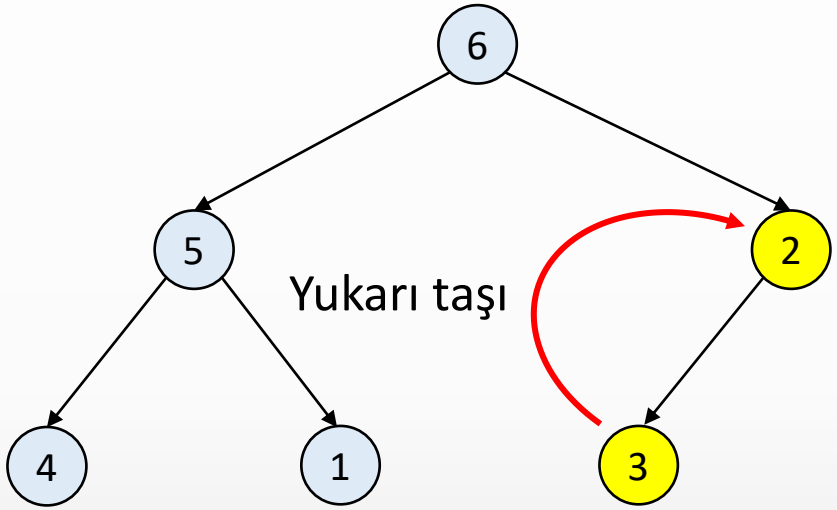

```
ekle(3)
```



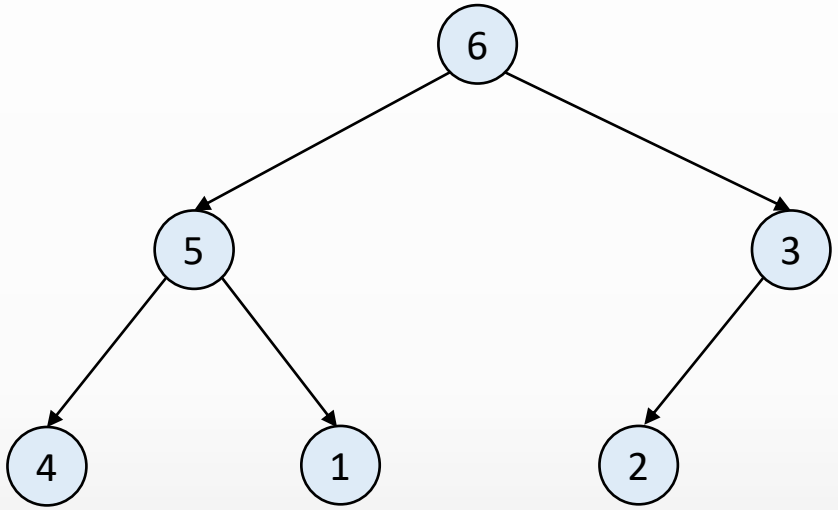

```
ekle(3)
```



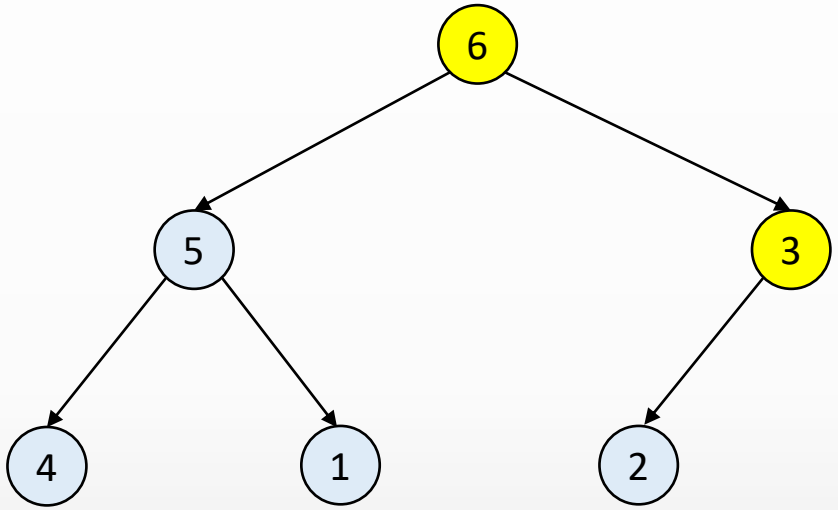

```
ekle(3)
```



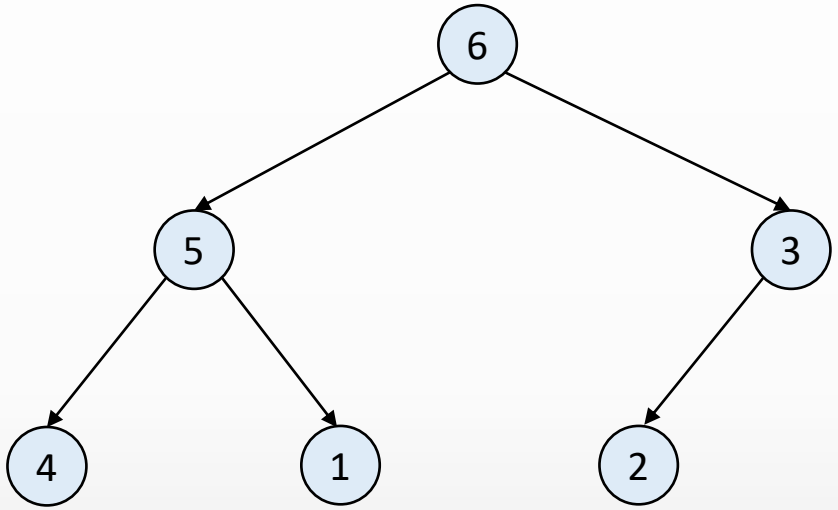

```
ekle(3)
```




```
ekle(3)
```




```
ekle(3)
```




```
ekle(3)
```

Sercan KÜLCÜ, Tüm hakları saklıdır.

Max Heap Ağacına Eleman Ekleme

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

Max Heap Ağacına Eleman Ekleme

```
MaxOK ok = new MaxOK(3);
```

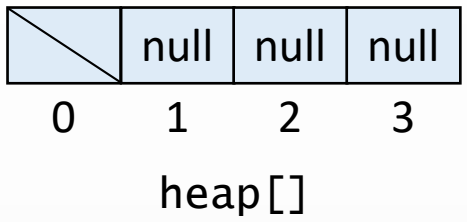
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

null	null	null	null
0	1	2	3

heap[]

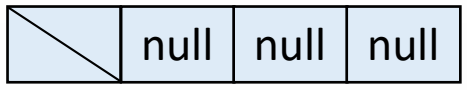
```
MaxOK ok = new MaxOK(3);
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
MaxOK ok = new MaxOK(3);
```

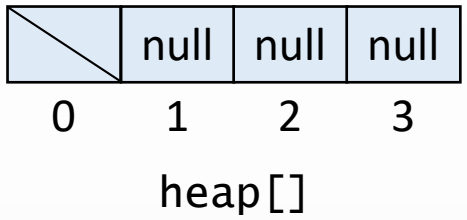
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
heap.length = 4
```

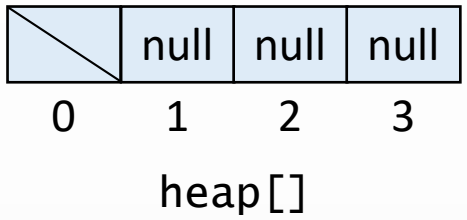
```
MaxOK ok = new MaxOK(3);
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



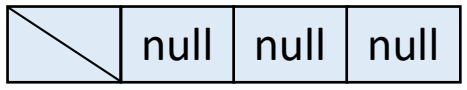
```
n = 0
heap.length = 4


MaxOK ok = new MaxOK(3);
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



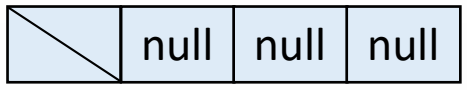
```
n = 0
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



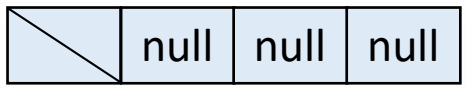
```
n = 0
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



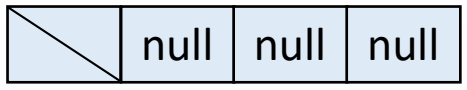
```
x = 4
n = 0
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



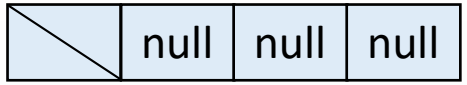
```
x = 4
n = 0
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
x = 4
n = 0
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
x = 4
n = 1
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 4
n = 1
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 4
n = 1
heap.length = 4


ekle(4)
```



```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 4
n = 1
heap.length = 4


ekle(4)
```

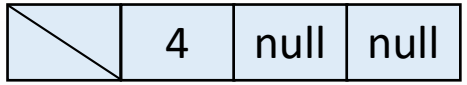
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
k = 1
x = 4
n = 1
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
k = 1
x = 4
n = 1
heap.length = 4


ekle(4)
```

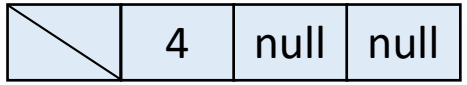
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
k = 1
x = 4
n = 1
heap.length = 4


ekle(4)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
k = 1
x = 4
n = 1
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 4
n = 1
heap.length = 4


ekle(4)
```



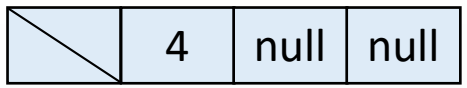
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

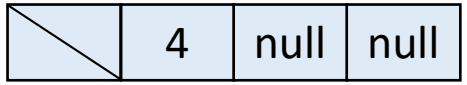
```
x = 4
n = 1
heap.length = 4


ekle(4)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



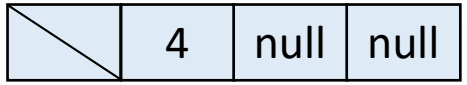
```
n = 1
heap.length = 4
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
n = 1
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
x = 5
n = 1
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 5
n = 1
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 5
n = 2
heap.length = 4


ekle(5)
```

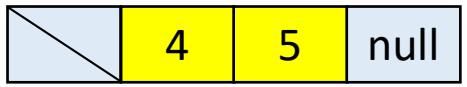
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

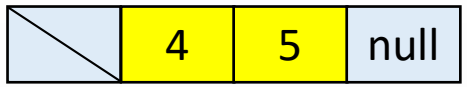
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



heap[]

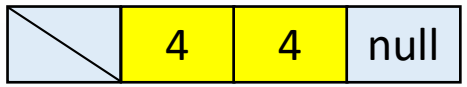
```
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



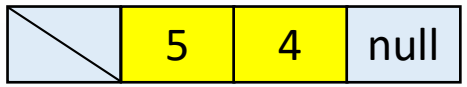
```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



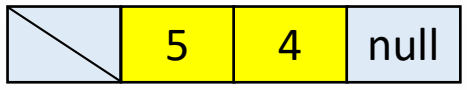
```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



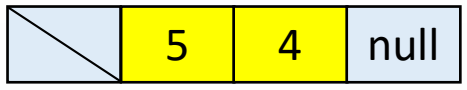
```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k = 1
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k = 1
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

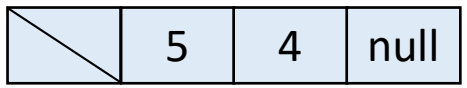
```
x = 5
n = 2
heap.length = 4


ekle(5)
```



```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

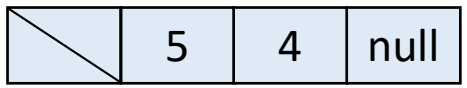
```
x = 5
n = 2
heap.length = 4


ekle(5)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
n = 2
heap.length = 4
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
n = 2
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 2
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 2
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	null
0	1	2	3

heap[]

```
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2
0	1	2	3

heap[]

k = 3
x = 2
n = 3
heap.length = 4

ekle(2)

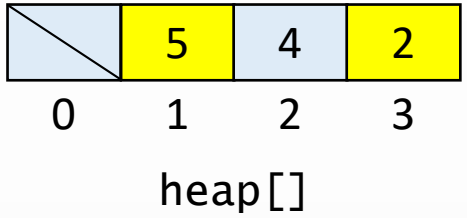
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2
0	1	2	3

heap[]

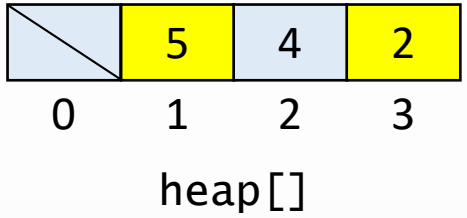
```
k = 3
x = 2
n = 3
heap.length = 4


ekle(2)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k/2 = 1
k = 3
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k/2 = 1
k = 3
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

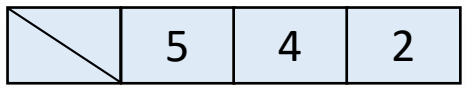
```
x = 2
n = 3
heap.length = 4


ekle(2)
```



```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 2
n = 3
heap.length = 4


ekle(2)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



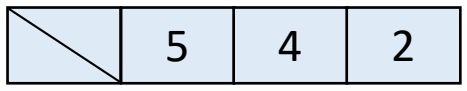
```
n = 3
heap.length = 4
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

n = 3
heap.length = 4

ekle(6)

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
x = 6
n = 3
heap.length = 4


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
x = 6
n = 3
heap.length = 4


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

x = 6
n = 3
heap.length = 4

ekle(6)

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	null	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 3
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	null	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	6	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	6	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	6	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	4	2	6	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 4
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

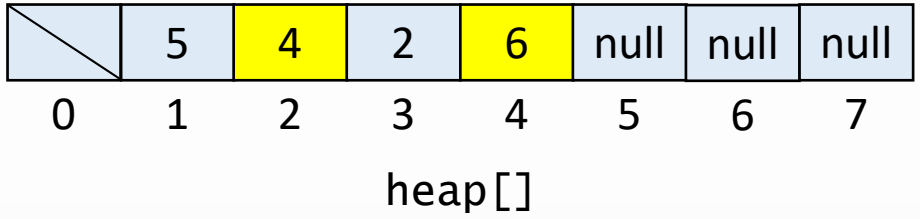
	5	4	2	6	null	null	null
0	1	2	3	4	5	6	7

heap[]

k/2 = 2
k = 4
x = 6
n = 4
heap.length = 8

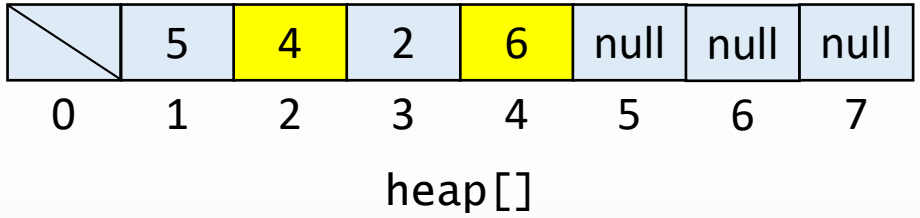
ekle(6)

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



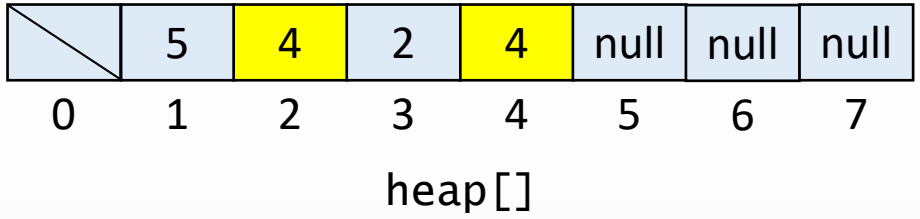

```
k/2 = 2
k = 4
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 6
k/2 = 2
k = 4
x = 6
n = 4
heap.length = 8


ekle(6)
```

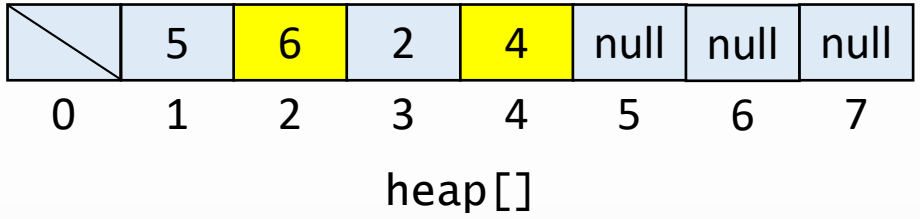
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 6
k/2 = 2
k = 4
x = 6
n = 4
heap.length = 8


ekle(6)
```

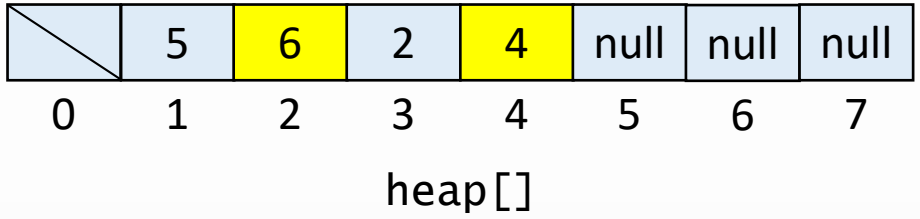


```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 6
k/2 = 2
k = 4
x = 6
n = 4
heap.length = 8


ekle(6)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
gecici = 6
k/2 = 2
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	6	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

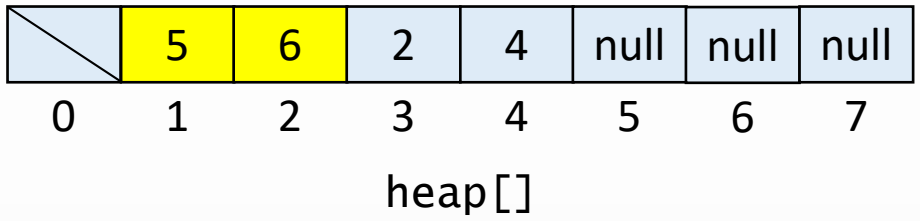
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	5	6	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

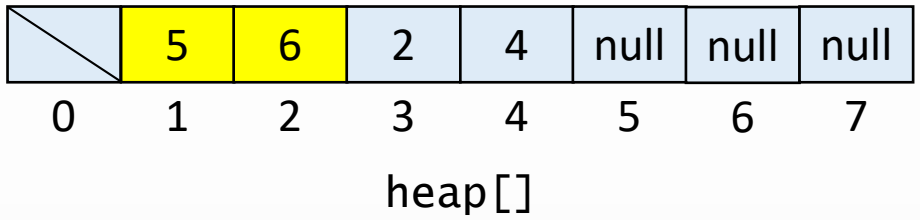
```
k/2 = 1
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



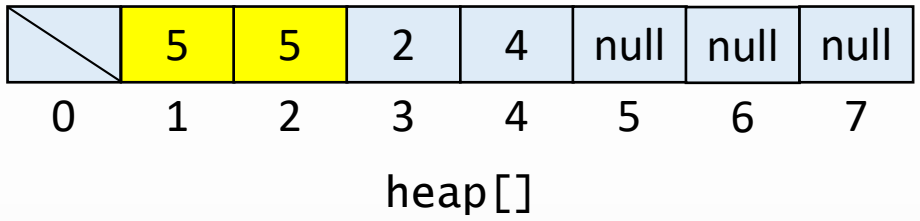
```
k/2 = 1
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 6
k/2 = 1
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

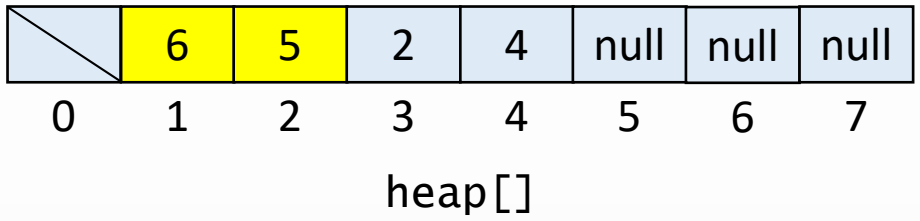
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
gecici = 6
k/2 = 1
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

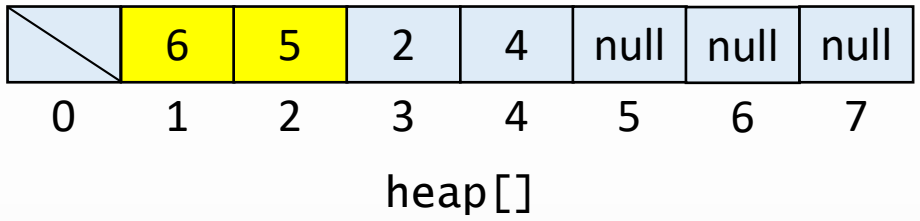


```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
gecici = 6
k/2 = 1
k = 2
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 6
k/2 = 1
k = 1
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 1
x = 6
n = 4
heap.length = 8


ekle(6)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 1
x = 6
n = 4
heap.length = 8


ekle(6)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 6
n = 4
heap.length = 8


ekle(6)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

n = 4
heap.length = 8

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
n = 4
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 4
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 4
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	null	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 5
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
k = 5
x = 1
n = 5
heap.length = 8


ekle(1)
```

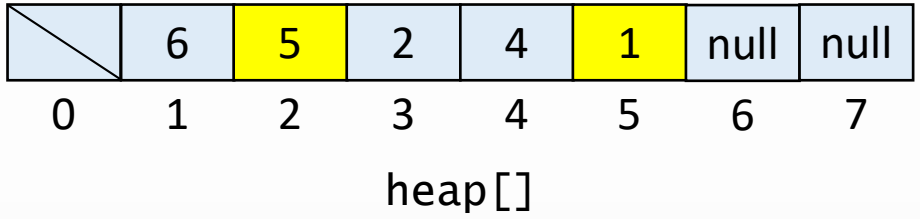
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
k/2 = 2
k = 5
x = 1
n = 5
heap.length = 8


ekle(1)
```

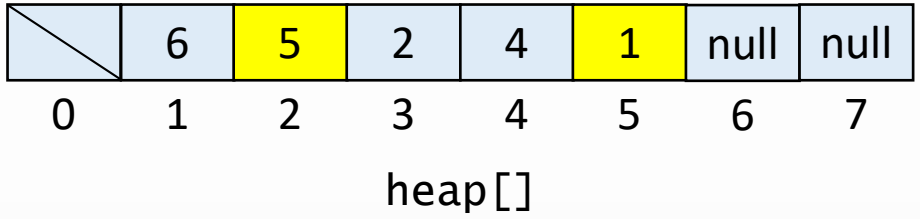
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



k/2 = 2
k = 5
x = 1
n = 5
heap.length = 8

ekle(1)

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
k/2 = 2
k = 5
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 5
heap.length = 8


ekle(1)
```

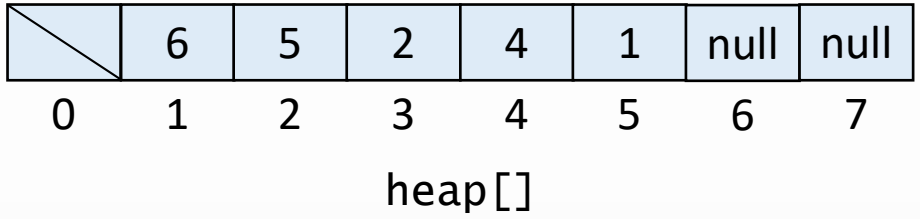
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 1
n = 5
heap.length = 8


ekle(1)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
n = 5
heap.length = 8
```

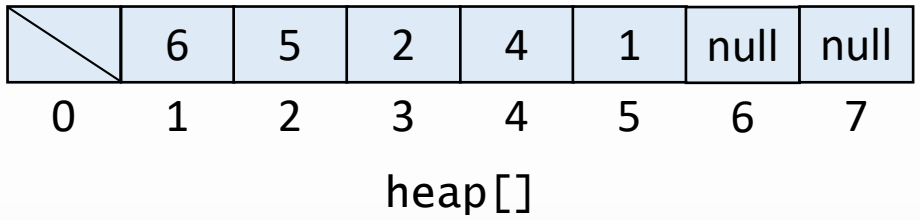
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 3
n = 5
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
x = 3
n = 5
heap.length = 8


ekle(3)
```

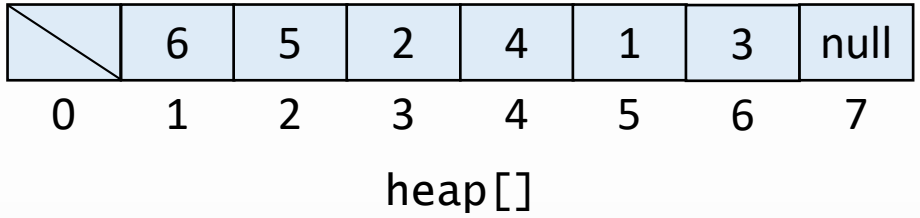
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	null	null
0	1	2	3	4	5	6	7

heap[]

```
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```




```
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	3	null
0	1	2	3	4	5	6	7

heap[]

```
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	3	null
0	1	2	3	4	5	6	7

heap[]

```
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```

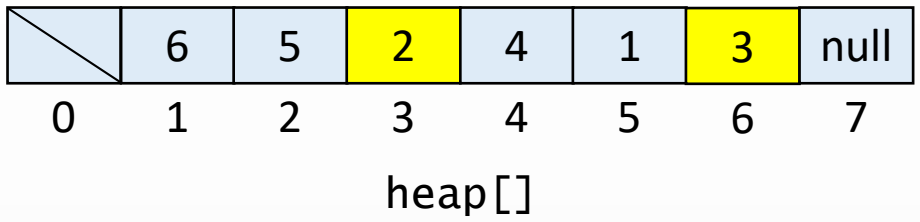
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	2	4	1	3	null
0	1	2	3	4	5	6	7

heap[]

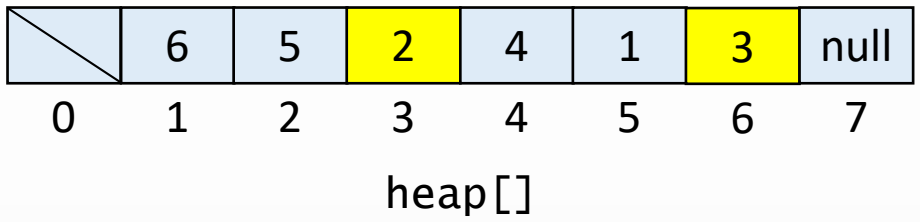
```
k/2 = 3
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



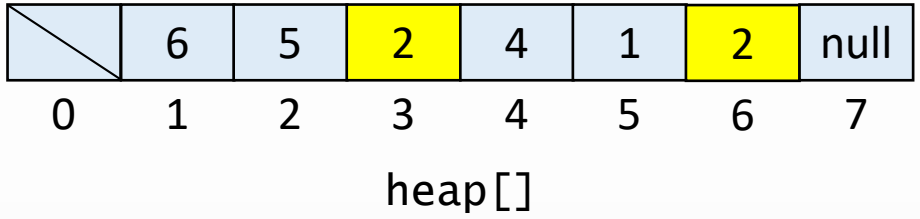
```
k/2 = 3
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 3
k/2 = 3
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```

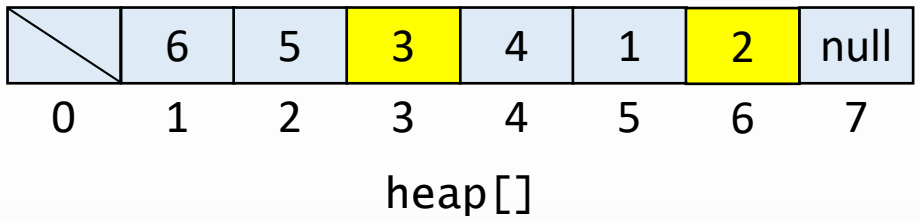
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 3
k/2 = 3
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```



```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
gecici = 3
k/2 = 3
k = 6
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

```
gecici = 3
k/2 = 3
k = 3
x = 3
n = 6
heap.length = 8


ekle(3)
```

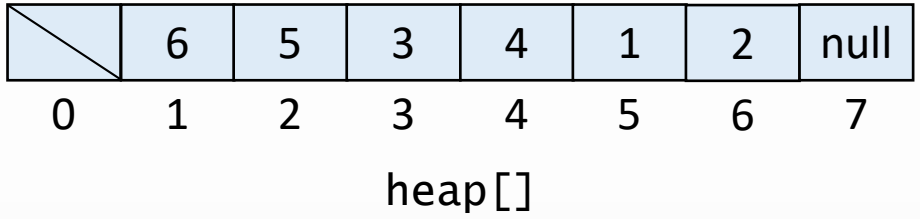
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	3	4	1	2	null
0	1	2	3	4	5	6	7

heap[]

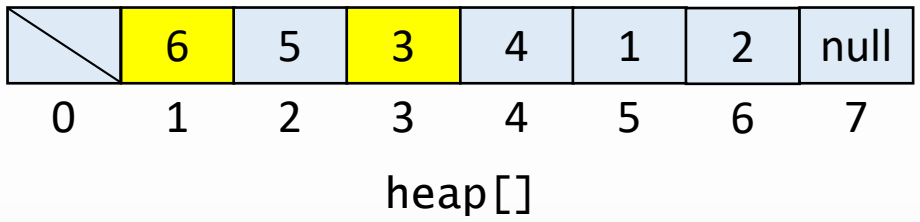
```
k = 3
x = 3
n = 6
heap.length = 8


ekle(3)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



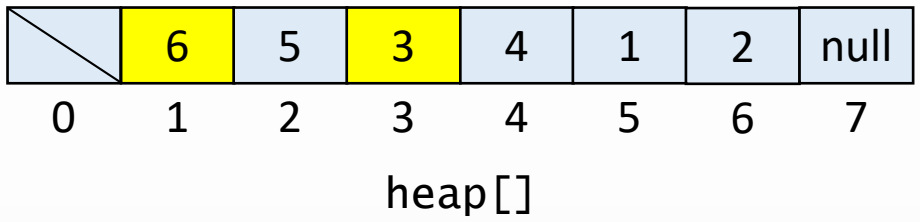

```
k/2 = 1
k = 3
x = 3
n = 6
heap.length = 8


ekle(3)
```

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k/2 = 1
k = 3
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



```
k/2 = 1
k = 3
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	3	4	1	2	null
0	1	2	3	4	5	6	7

heap[]

```
x = 3
n = 6
heap.length = 8


ekle(3)
```

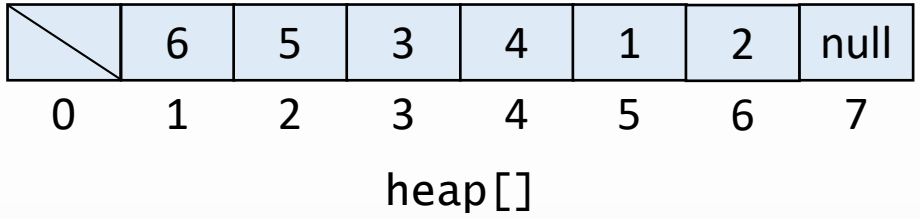
```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

	6	5	3	4	1	2	null
0	1	2	3	4	5	6	7

heap[]

```
x = 3
n = 6
heap.length = 8


ekle(3)
```

```java
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```



n = 6
heap.length = 8

```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

private void yuzdur(int k) {
  while (k > 1 && heap[k / 2] < heap[k]) {
    int gecici = heap[k];
    heap[k] = heap[k / 2];
    heap[k / 2] = gecici;
    k = k / 2;
  }
}
```

Yukarıdan Aşağıya Heap Ağacına Dönüştürme

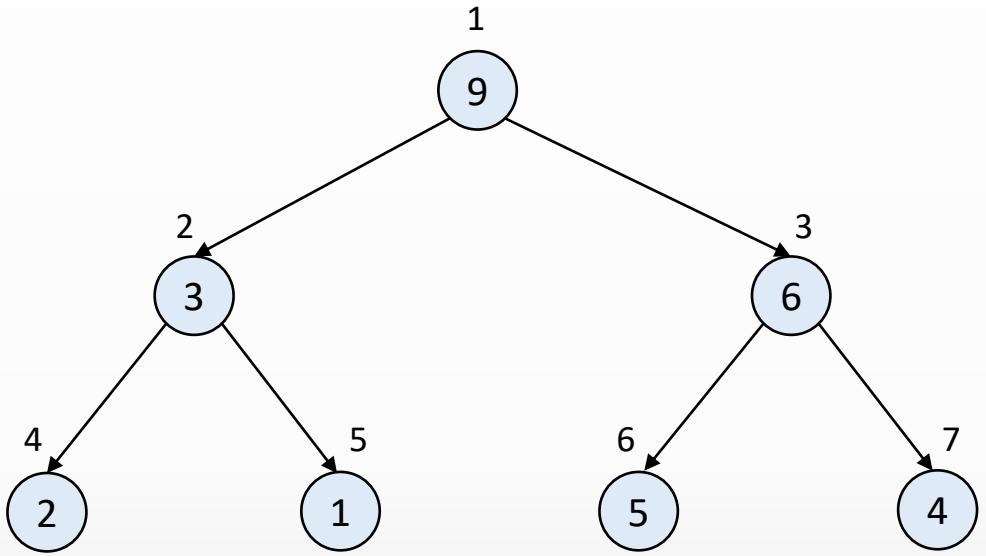
Sercan KÜLCÜ, Tüm hakları saklıdır.

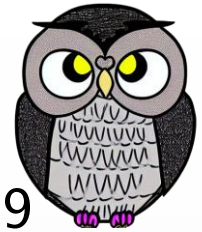
Yukarıdan Aşağıya Heap Ağacına Dönüştürme

- Max heap ikili ağacının her bir düğümünün değeri, çocuklarının değerlerinden büyüktür.
- Heap ağacından bir öğe çıkarıldıktan sonra bu özellik bozulabilir.
- Bu nedenle öğelerin yerlerinin değiştirilmesi gerekir.
- Ağaç yukarıdan aşağıya doğru taranarak yeniden heap ağacına dönüştürme işlemi (batır - sink) uygulanır (top-down heapify).

Sercan KÜLCÜ, Tüm hakları saklıdır.

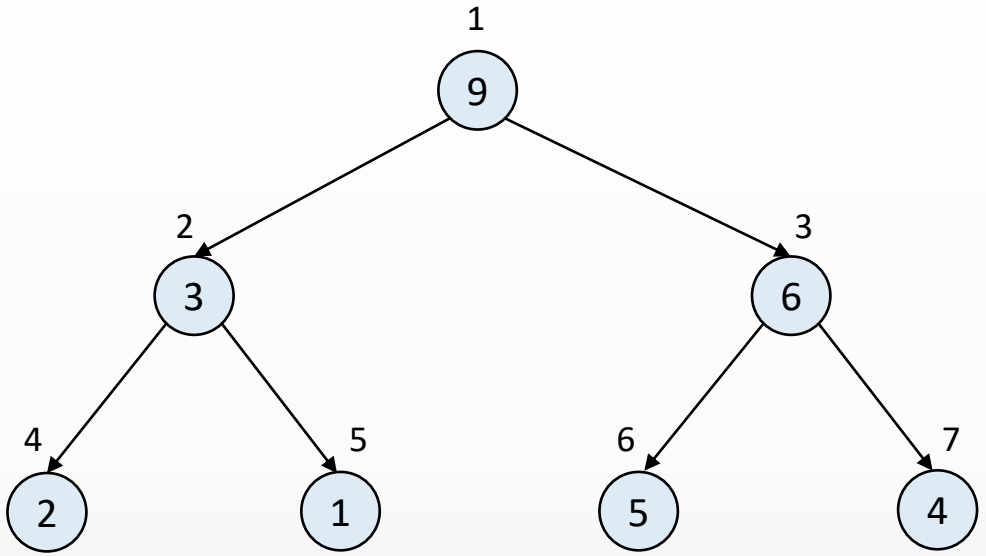
silMax()

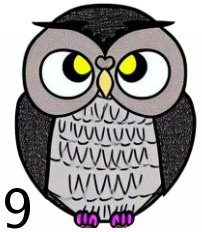




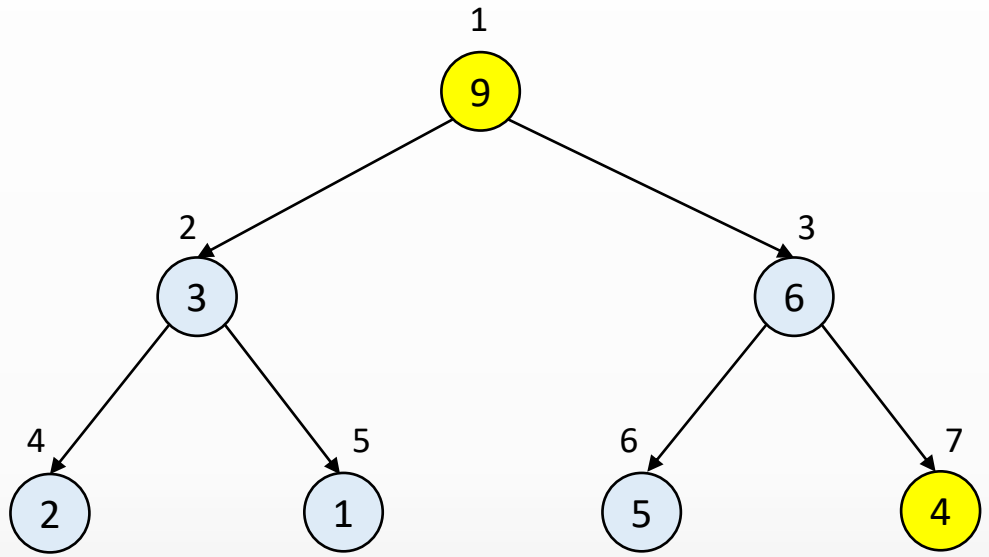
```
silMax()
```

max = 9



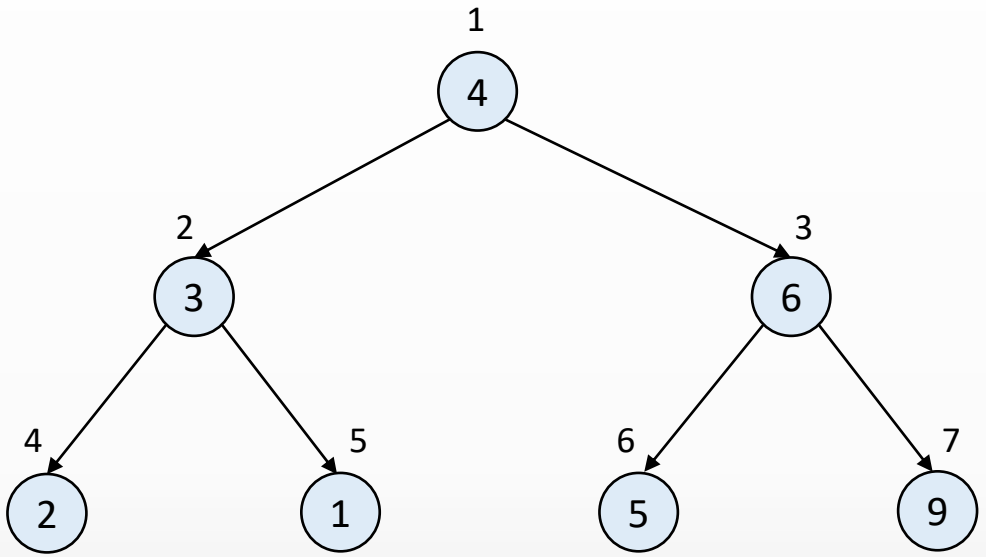


```
silMax()
```

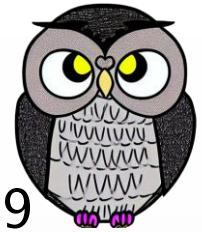



```
max = 9
```

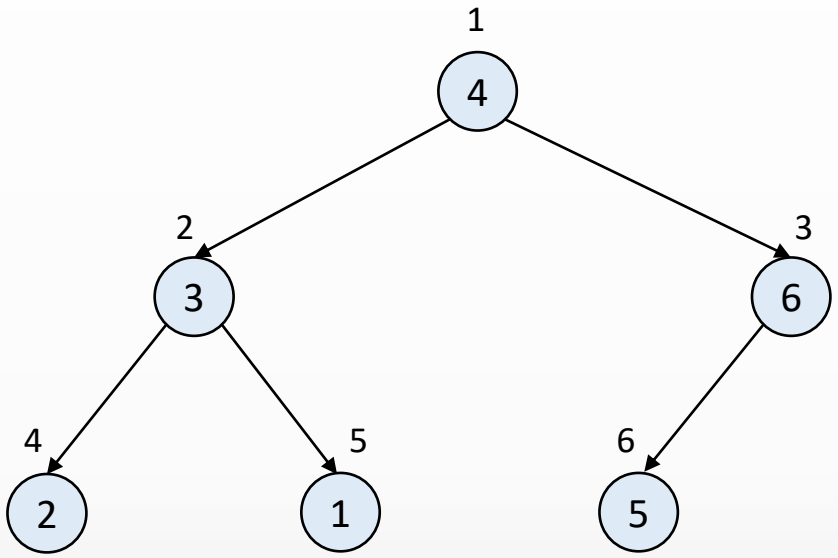
silMax()



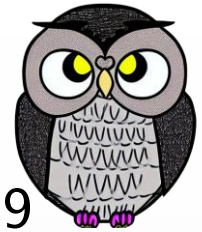
max = 9



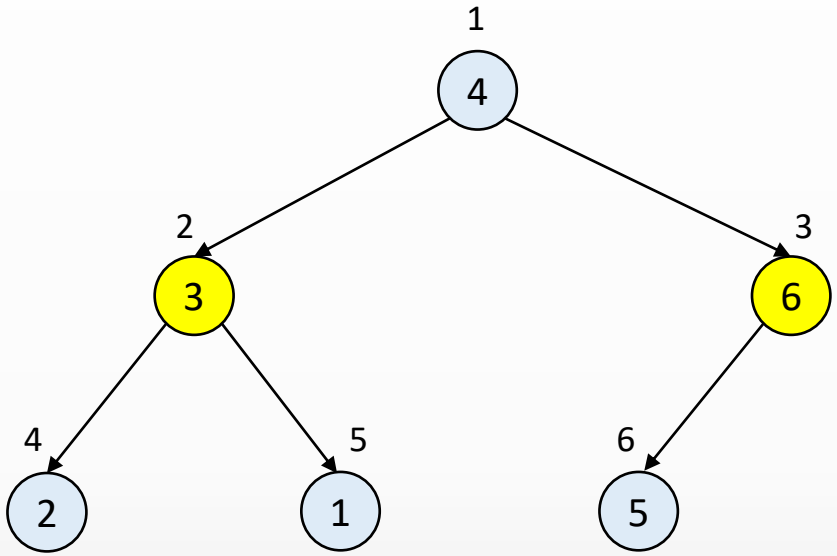
silMax()



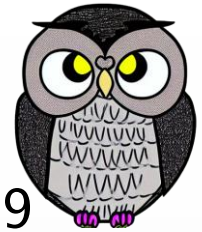
max = 9



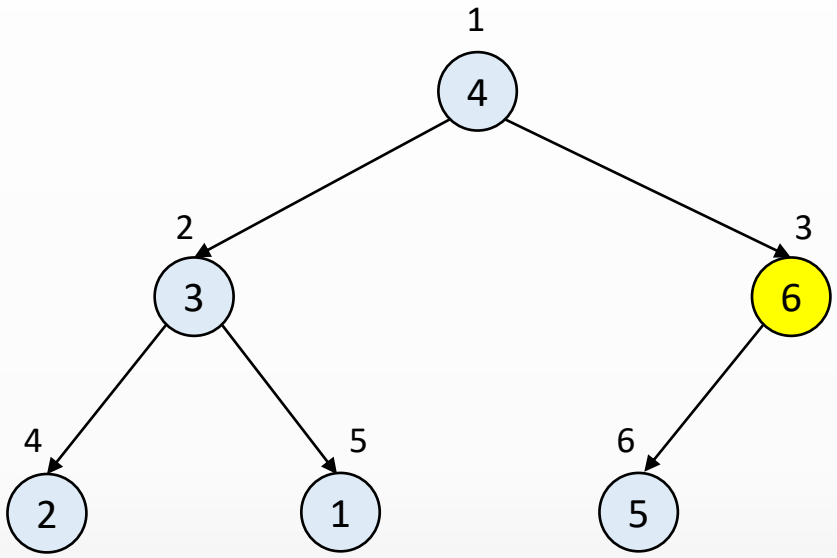
```
silMax()
```



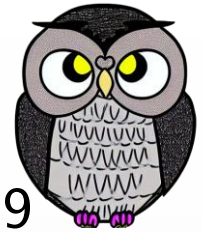

```
max = 9
```



silMax()

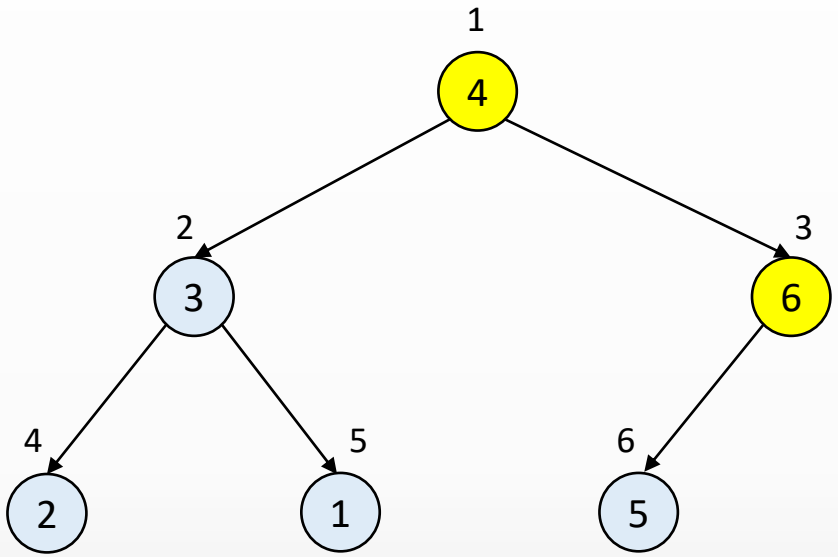


max = 9

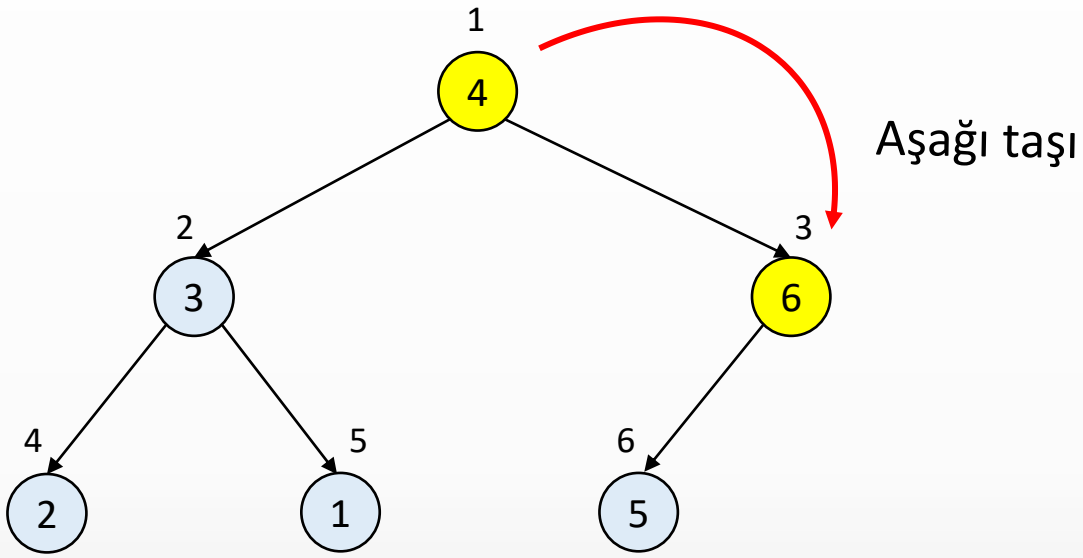


silMax()

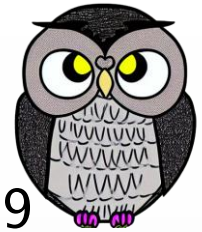
max = 9

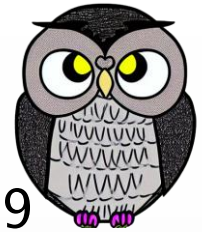


silMax()

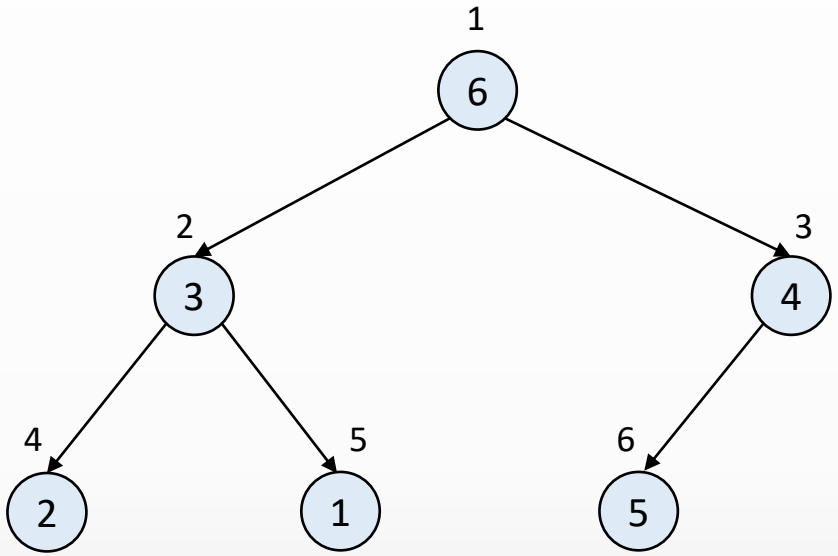


max = 9



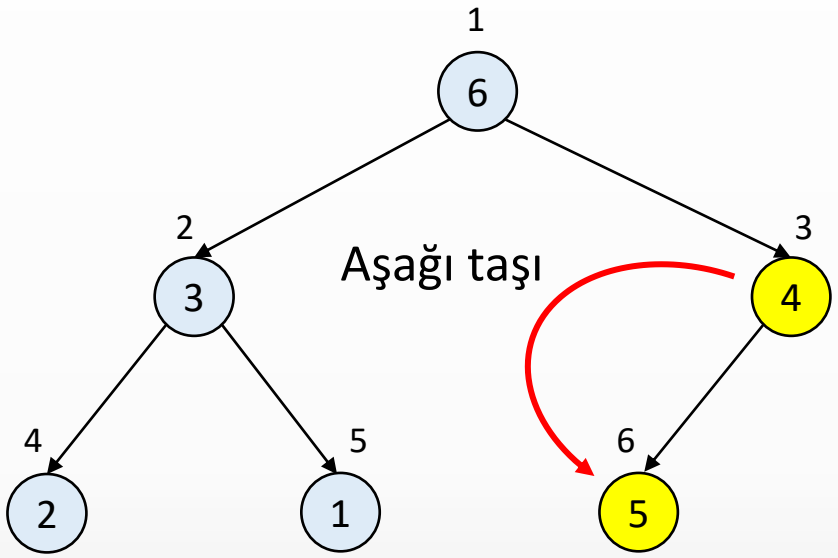


```
silMax()
```

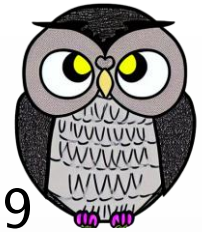


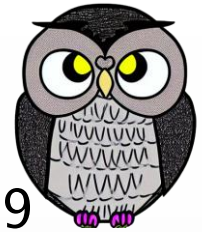

```
max = 9
```

silMax()

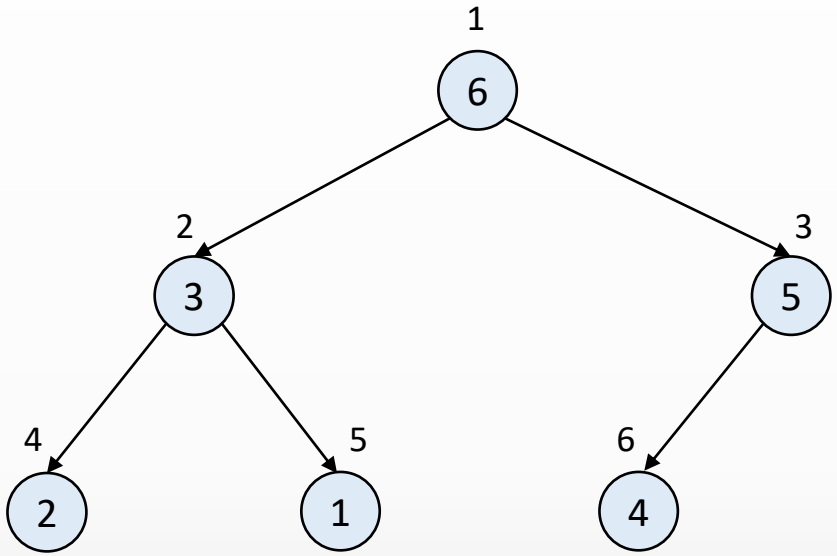


max = 9





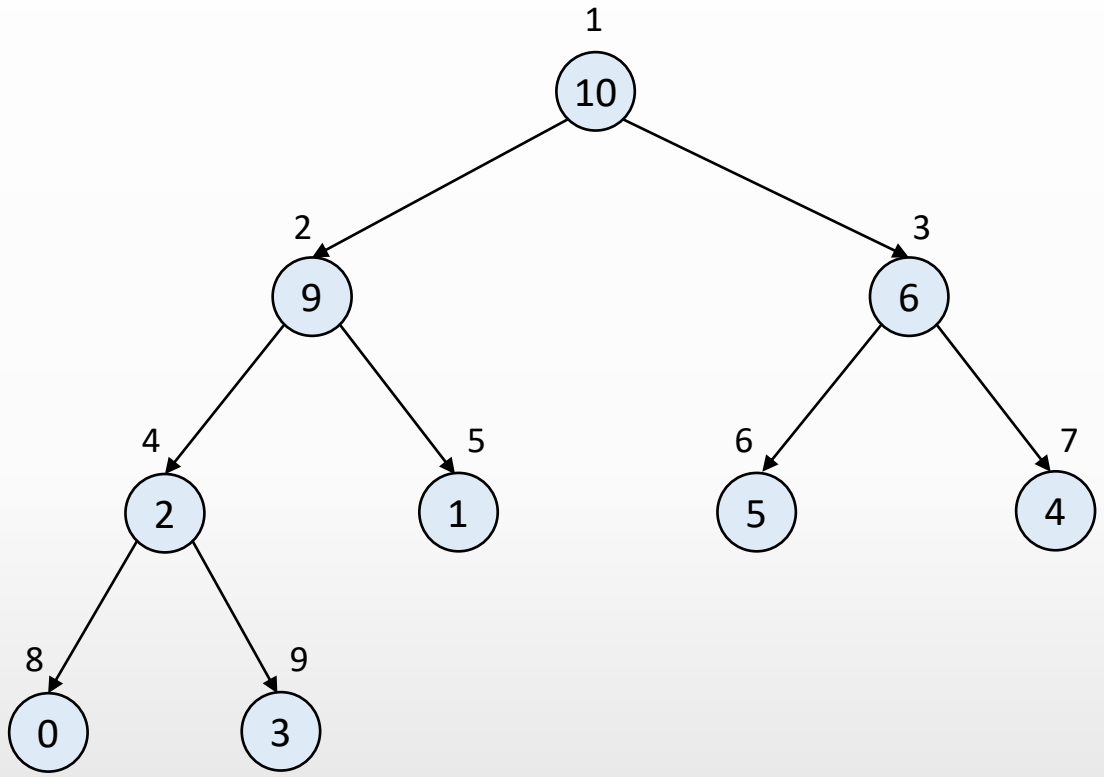
```
silMax()
```



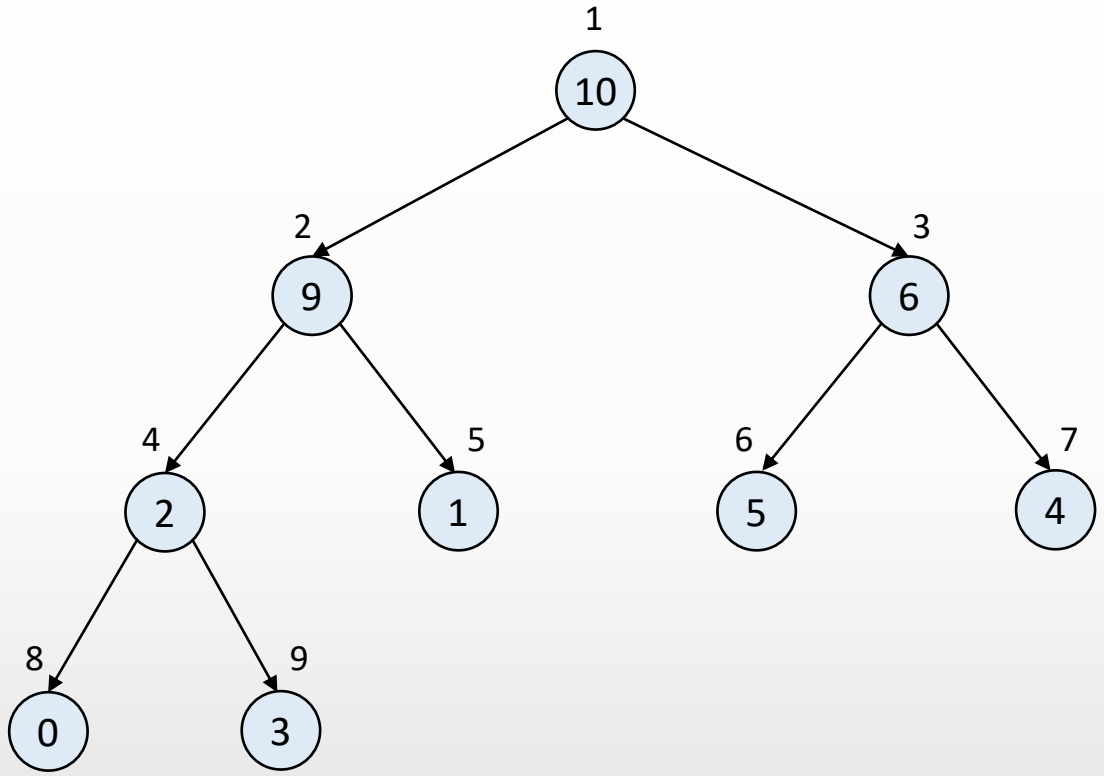

```
max = 9
```

Yukarıdan Aşağıya Heap Ağacına Dönüştürme

Sercan KÜLCÜ, Tüm hakları saklıdır.

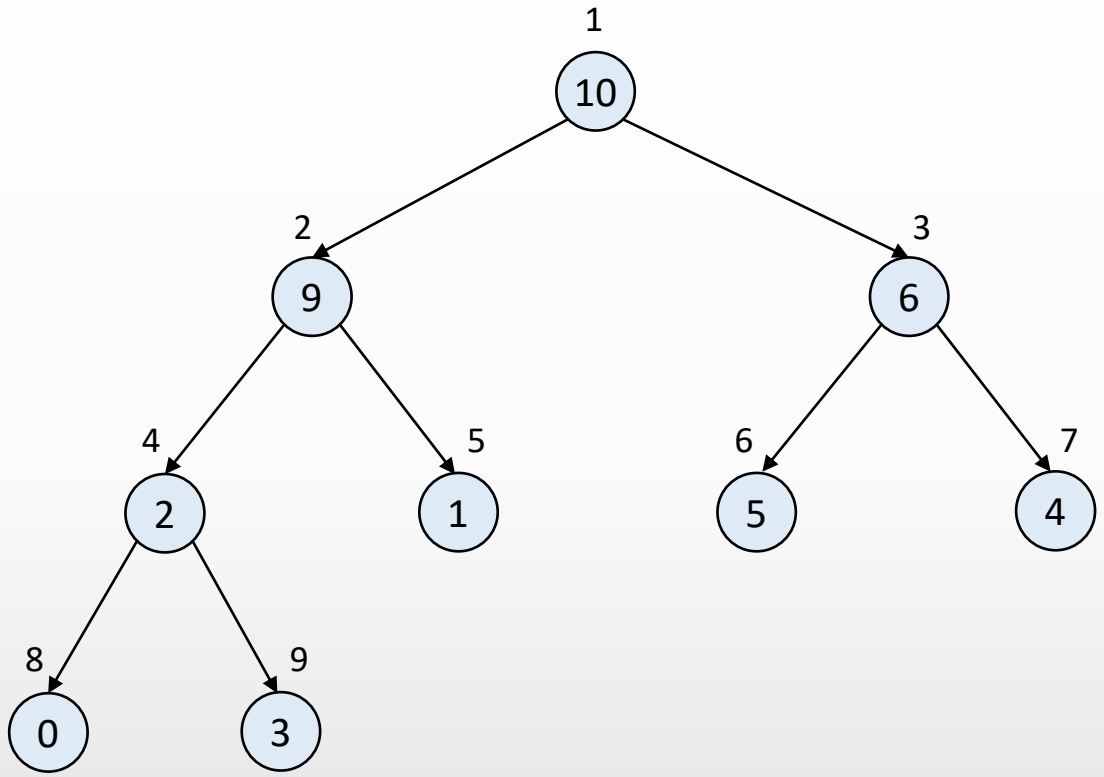

1
10
2
9
3
6
4
2
5
1
6
5
7
4
8
0
9
3

silMax()



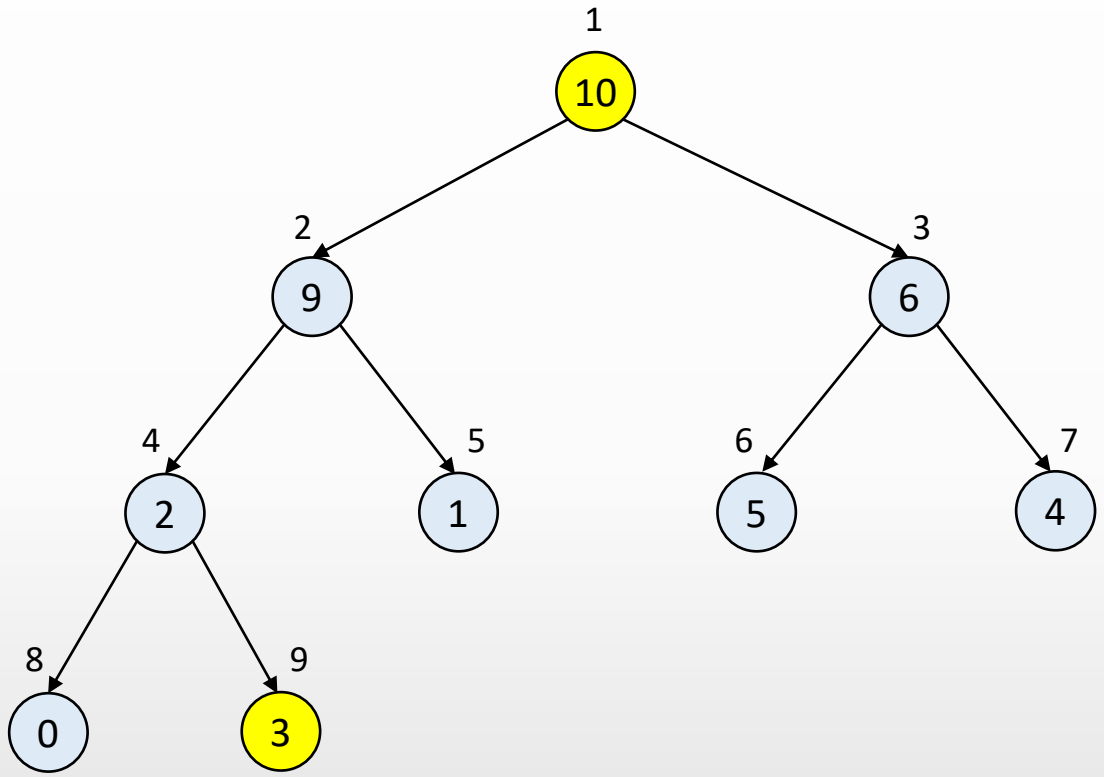
silMax()

max = 10



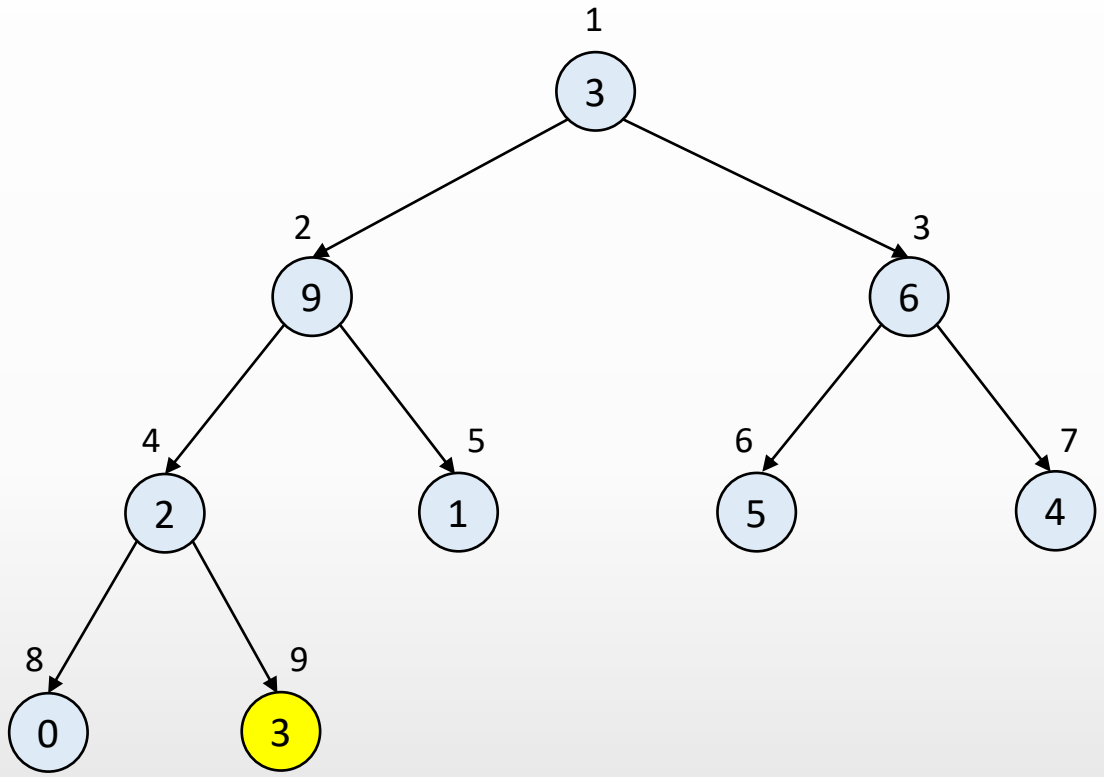
silMax()

max = 10



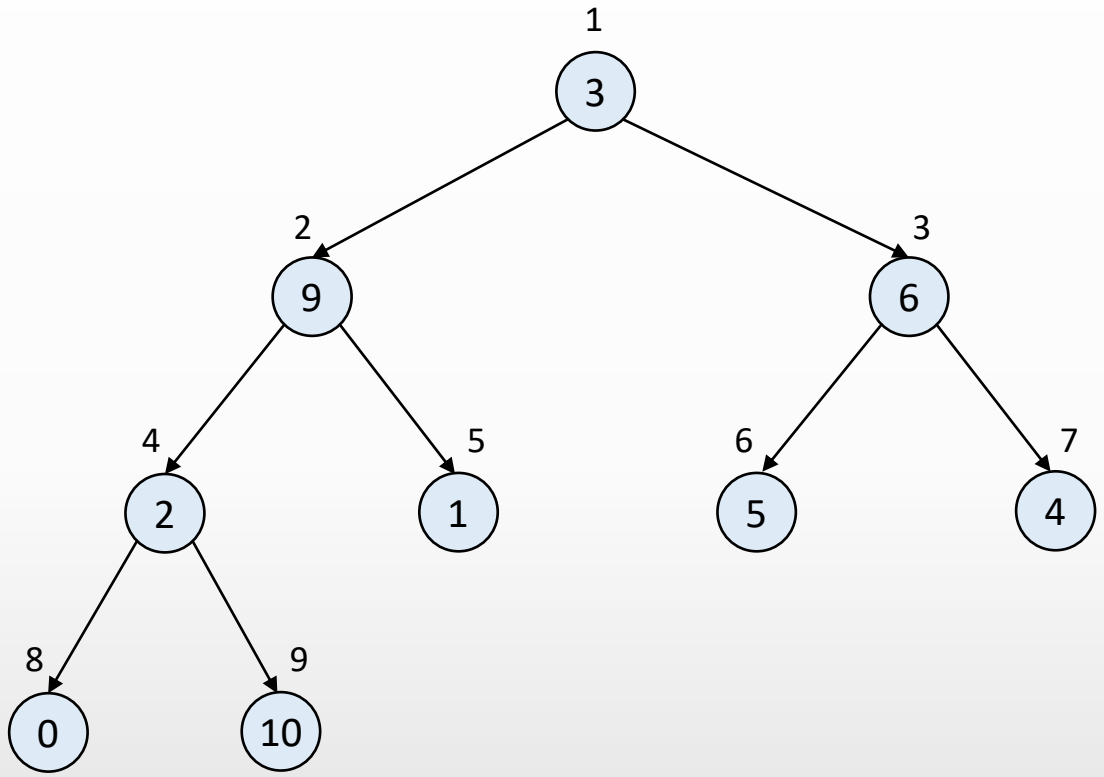
silMax()

max = 10



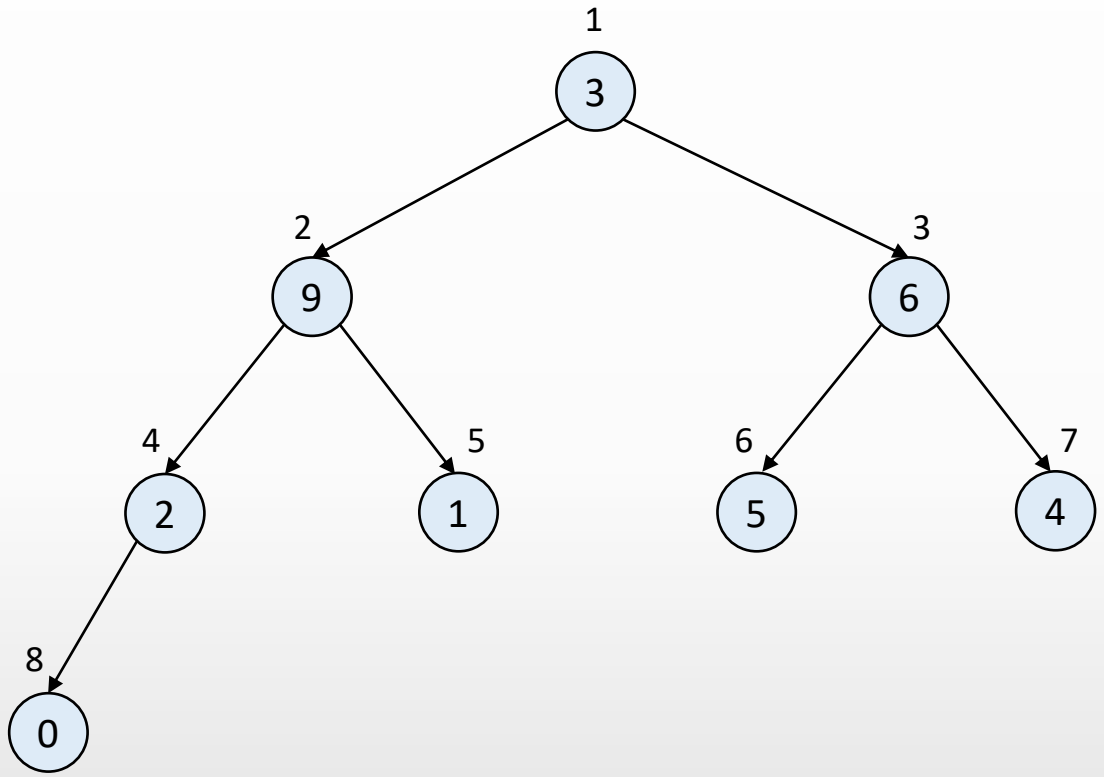
silMax()

max = 10



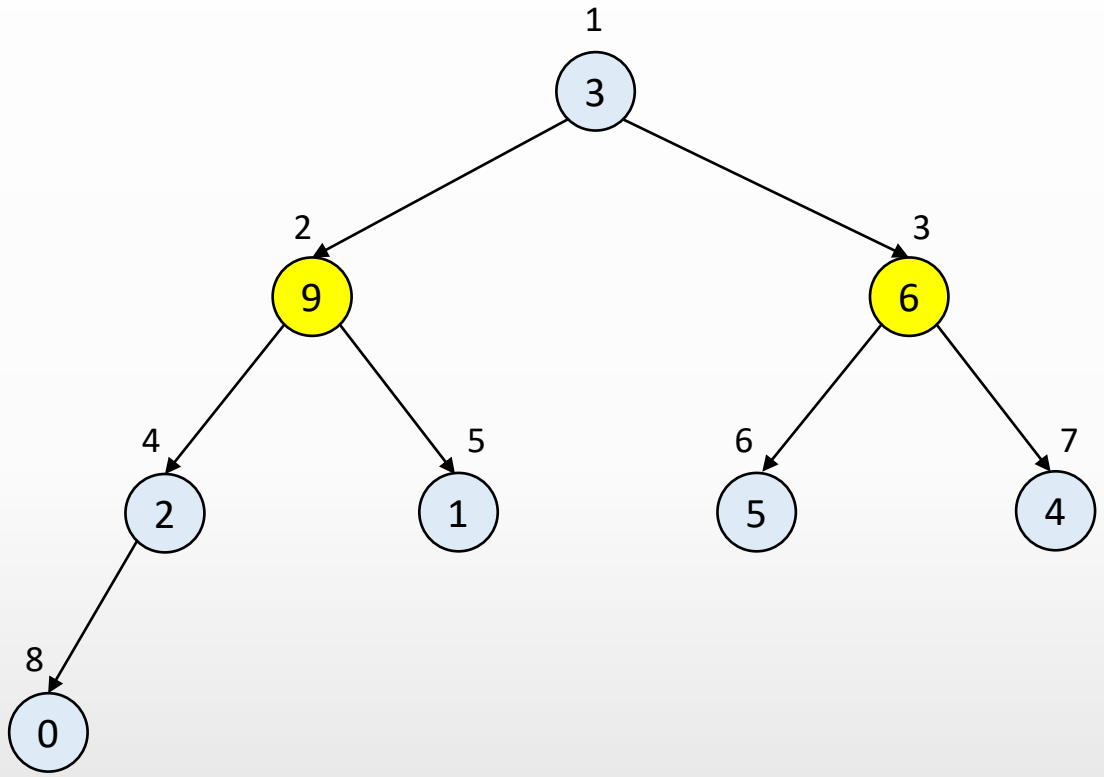
silMax()

max = 10



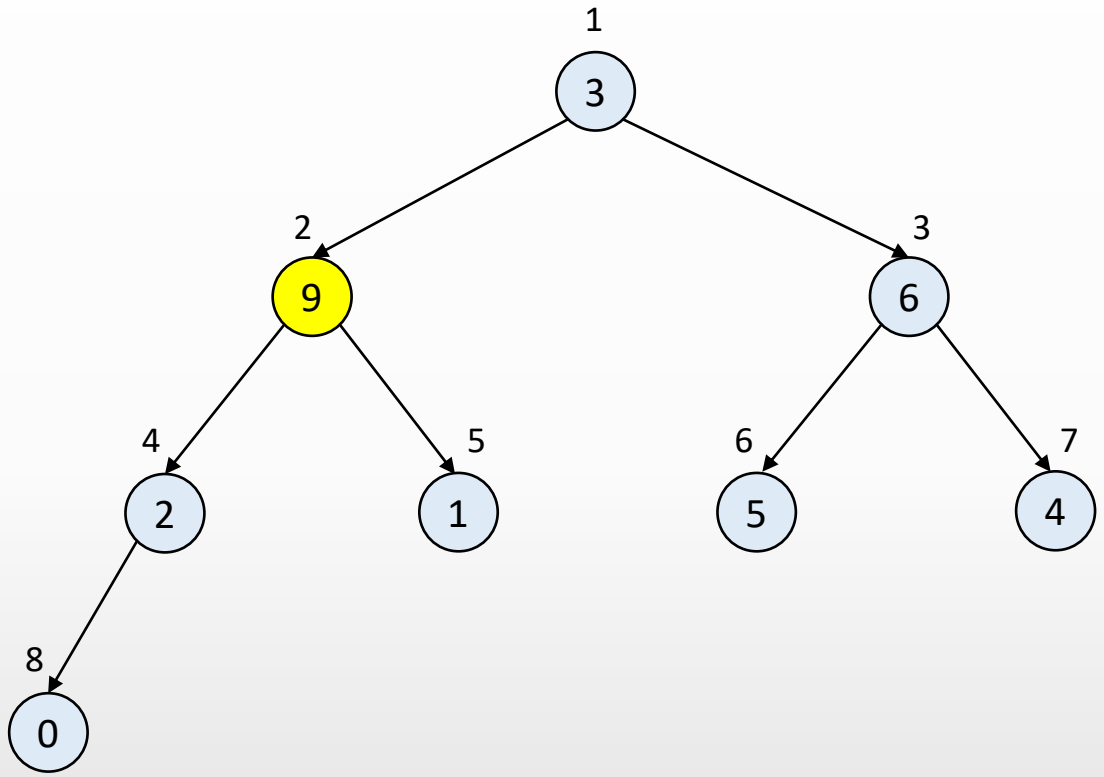
```
silMax()
```

```
max = 10
```



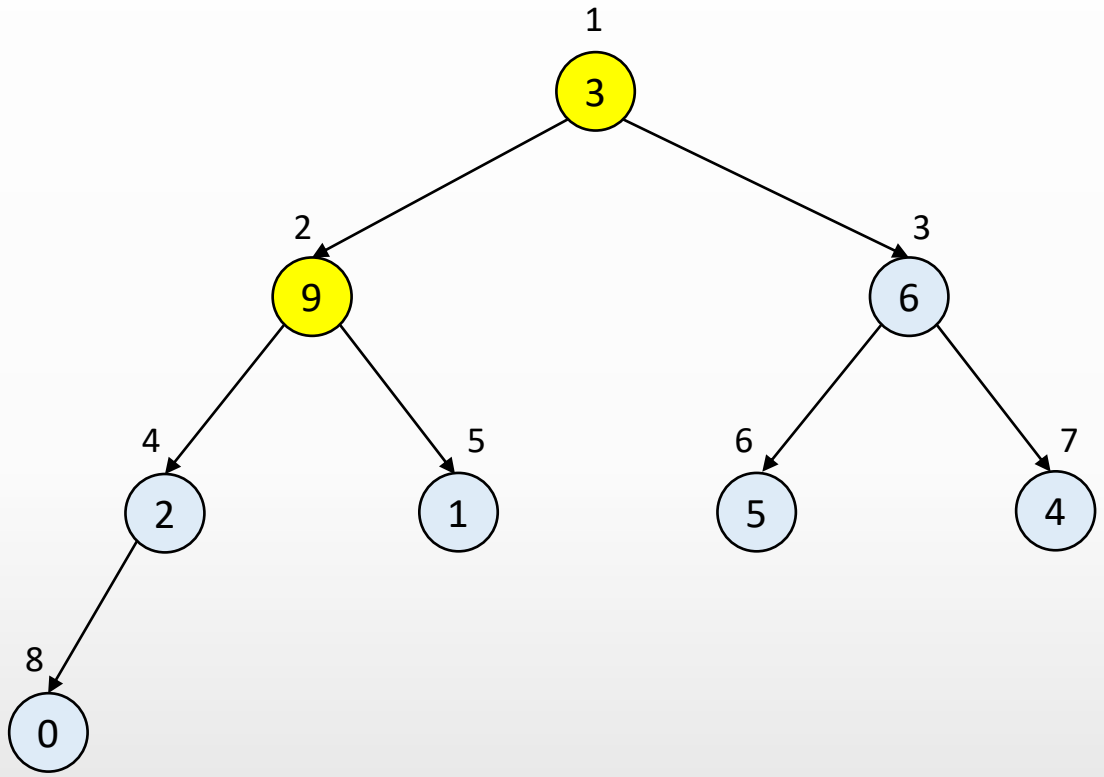
silMax()

max = 10



silMax()

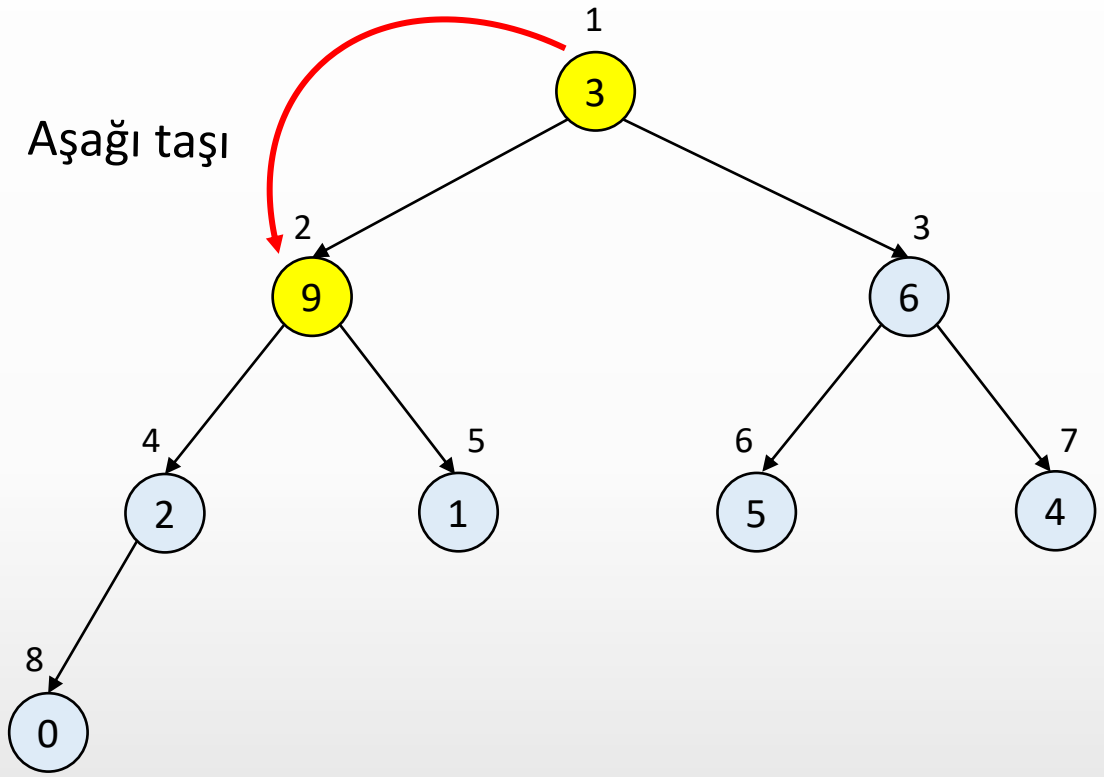
max = 10



`silMax()`

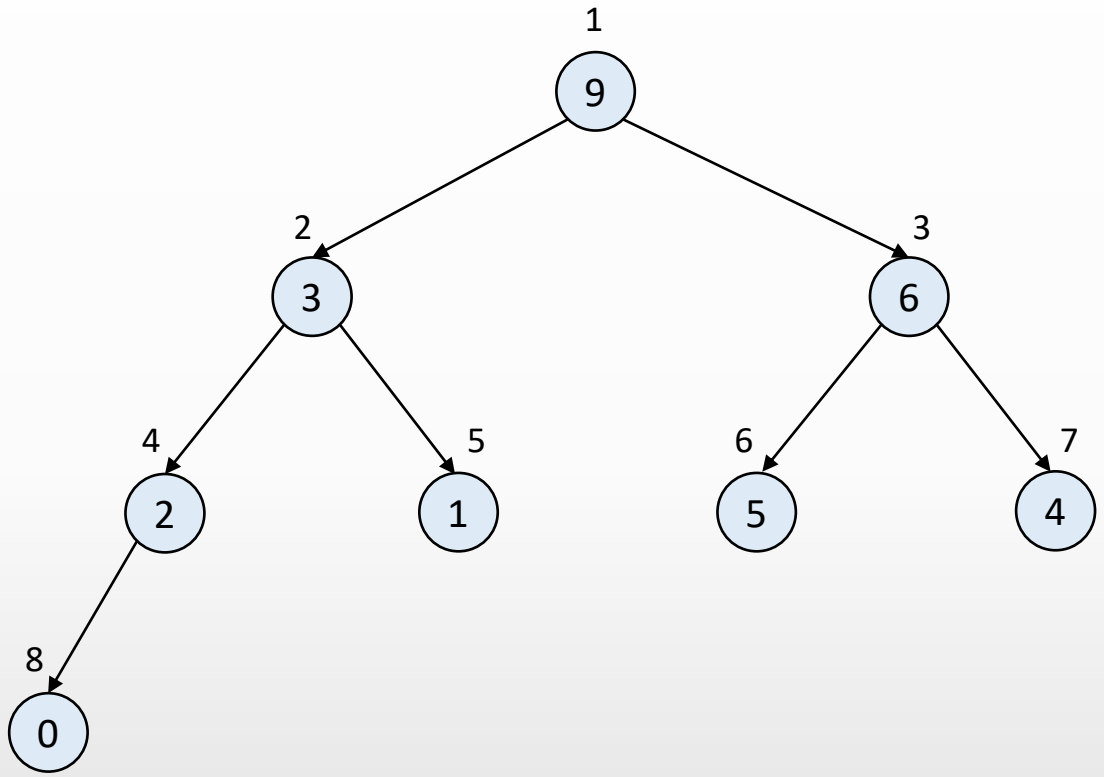
`max = 10`

Aşağı taşı



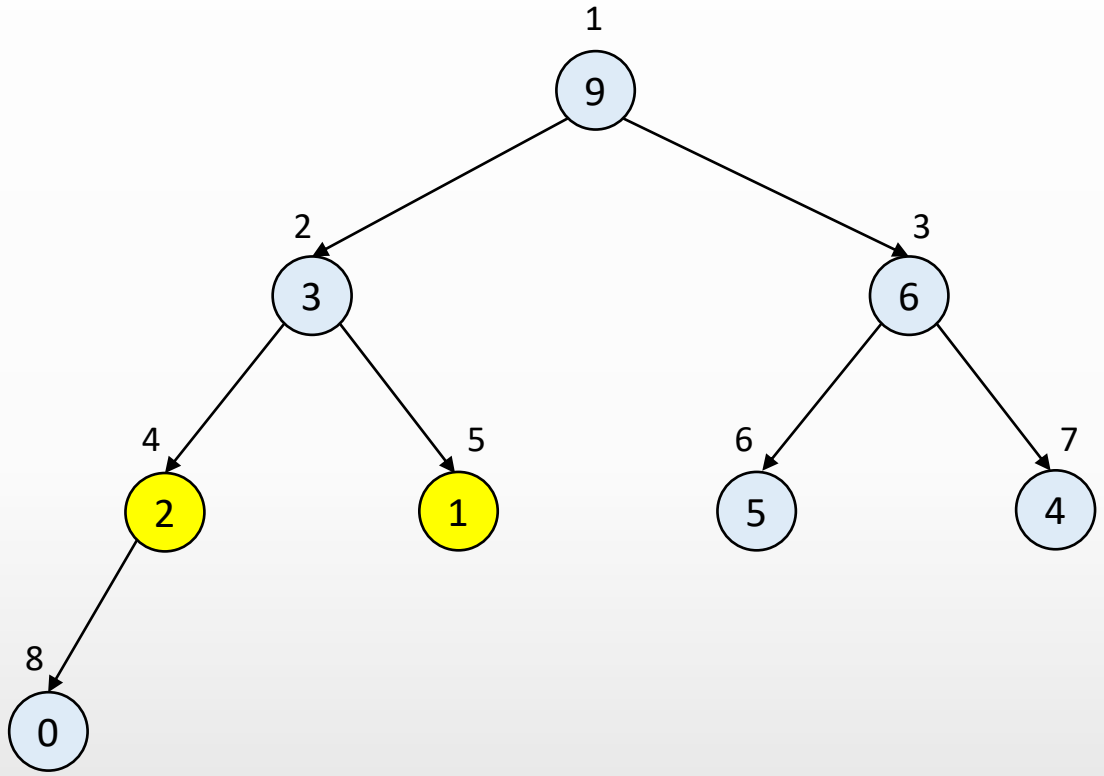
silMax()

max = 10



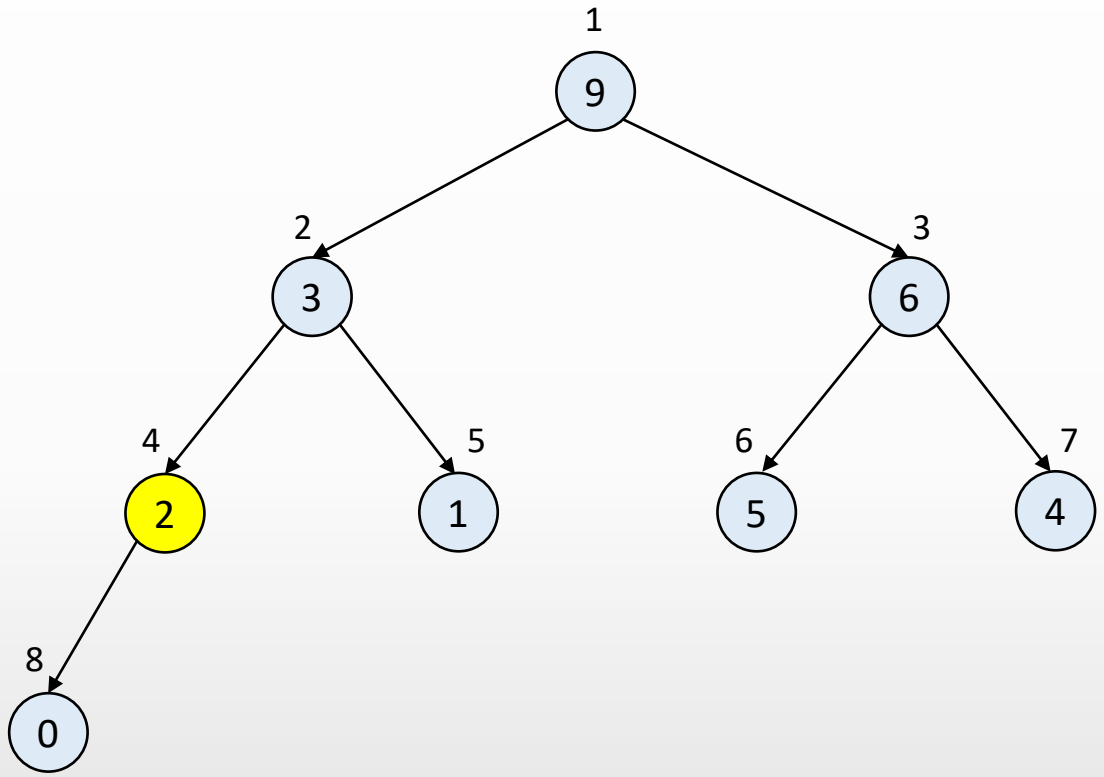
```
silMax()
```

max = 10



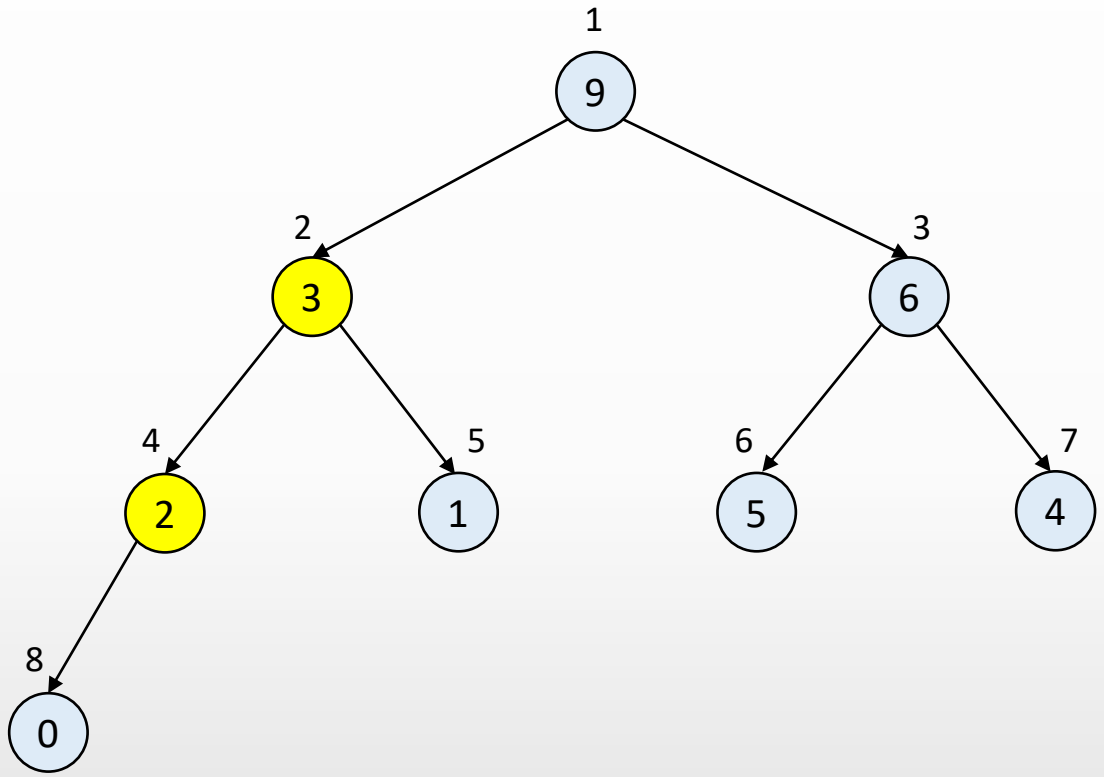
```
silMax()
```

```
max = 10
```



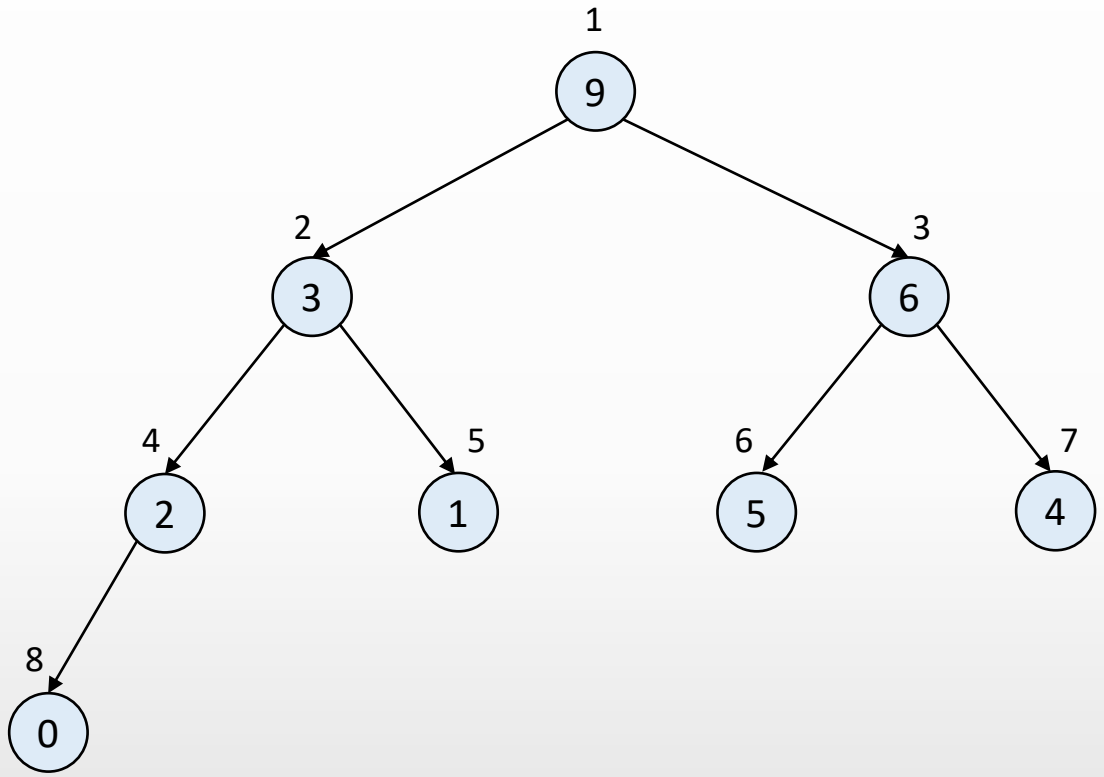
```
silMax()
```

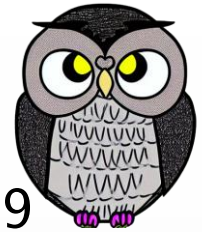
```
max = 10
```



silMax()

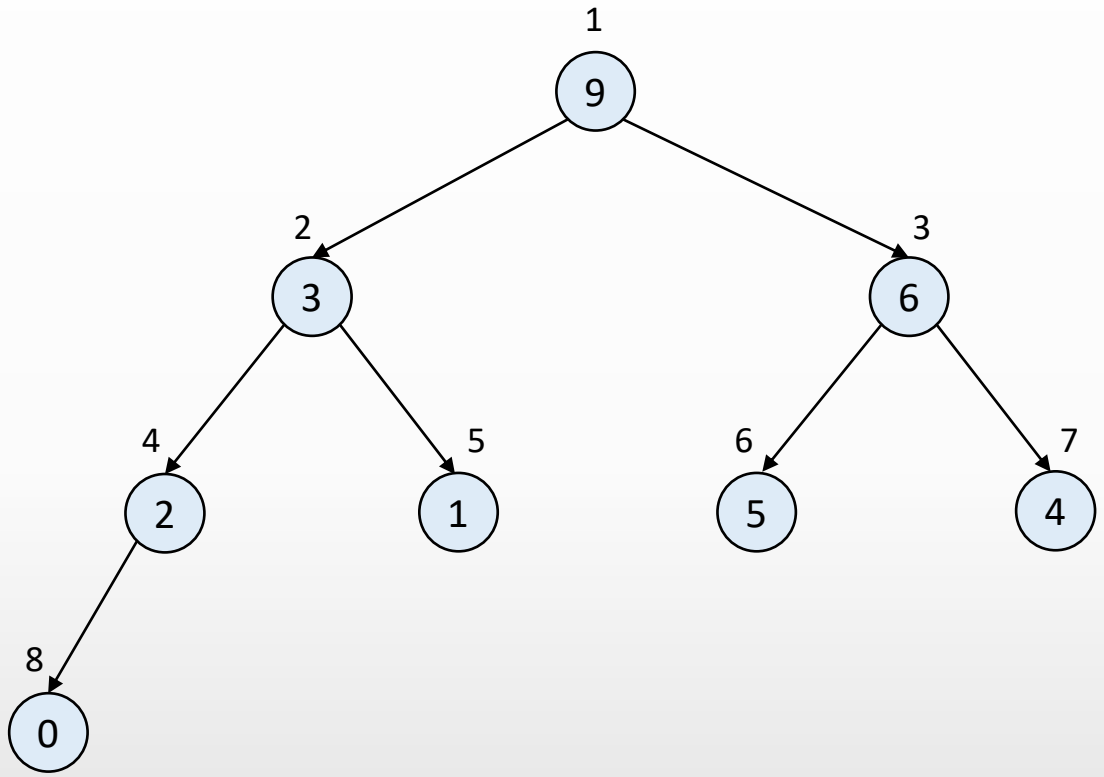
max = 10



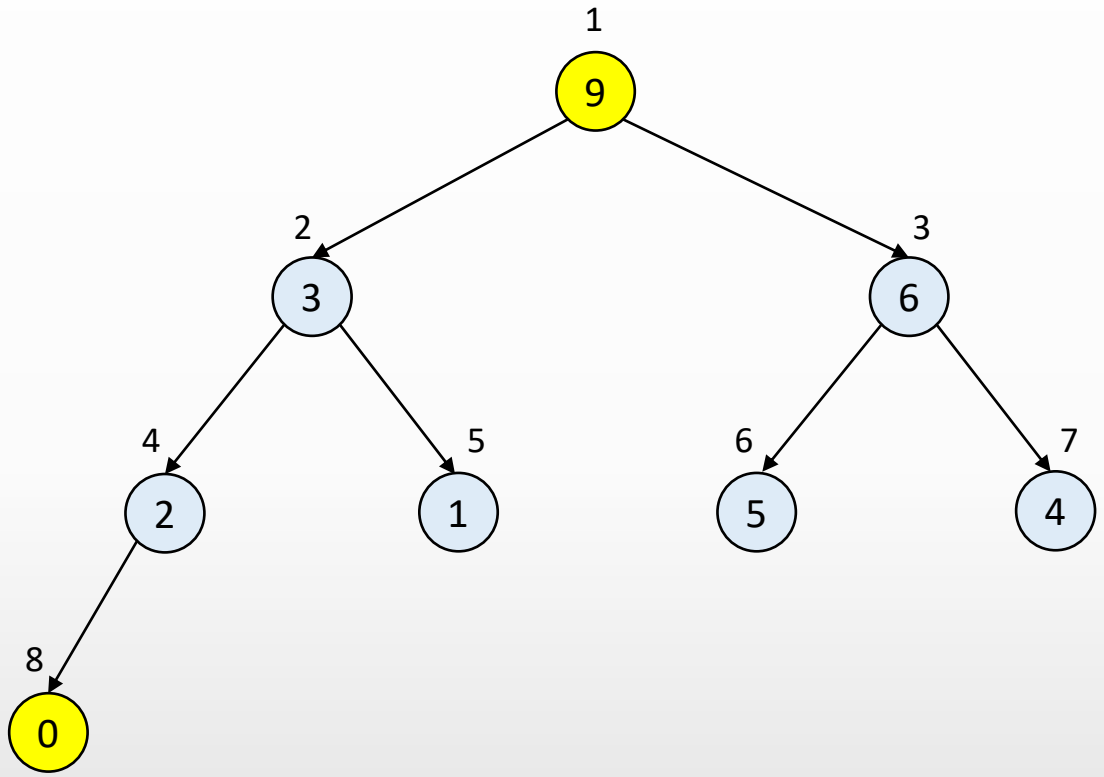


silMax()

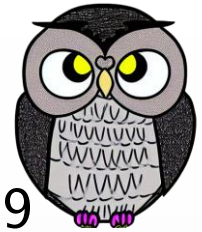
max = 9



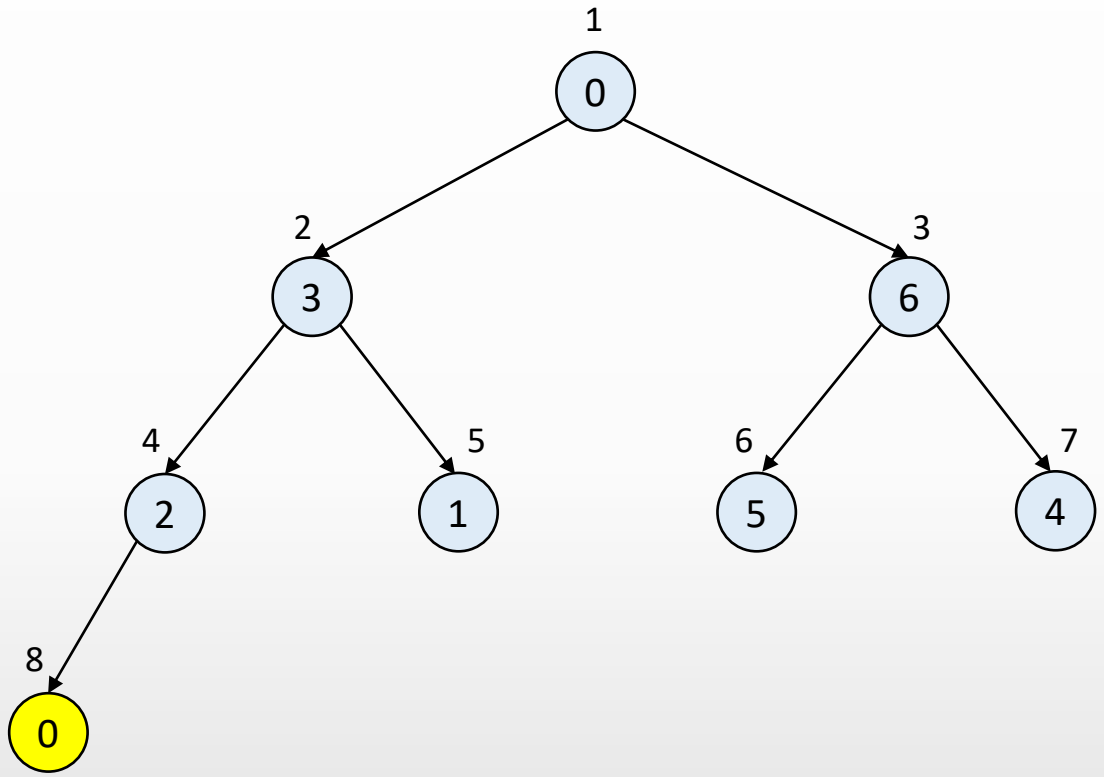
silMax()



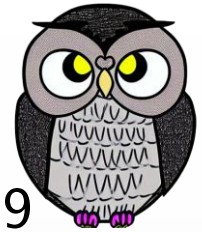
max = 9

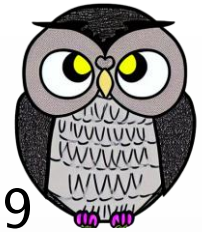


silMax()



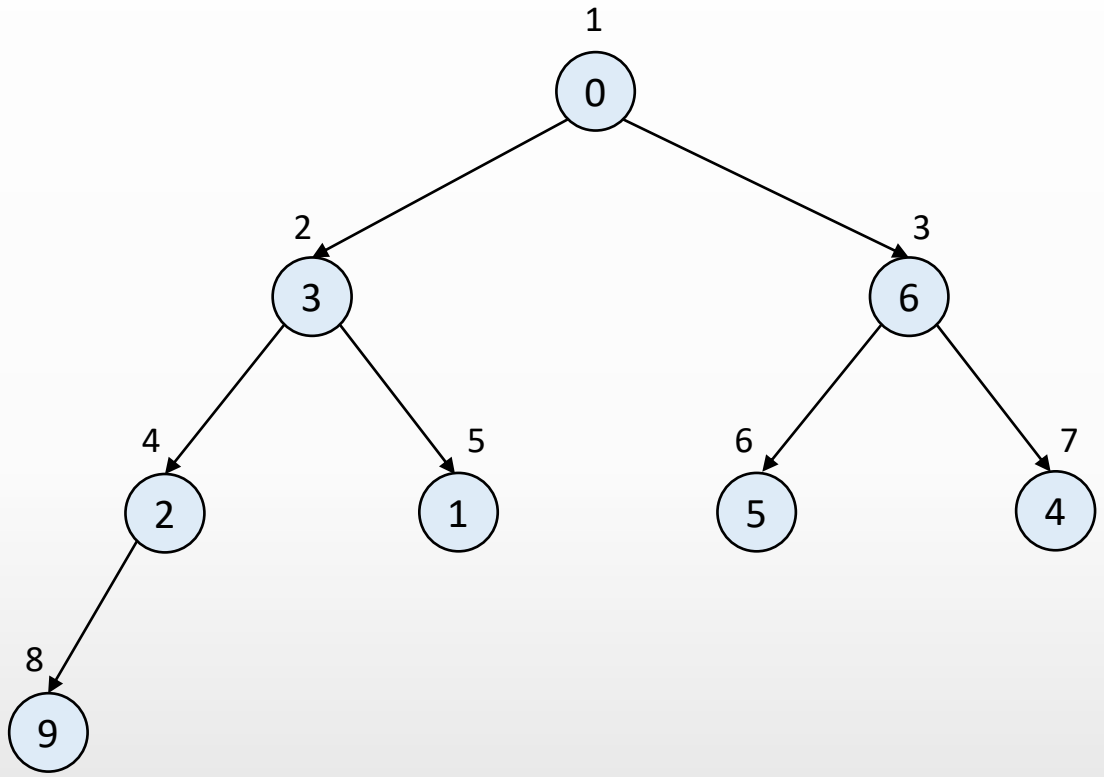
max = 9



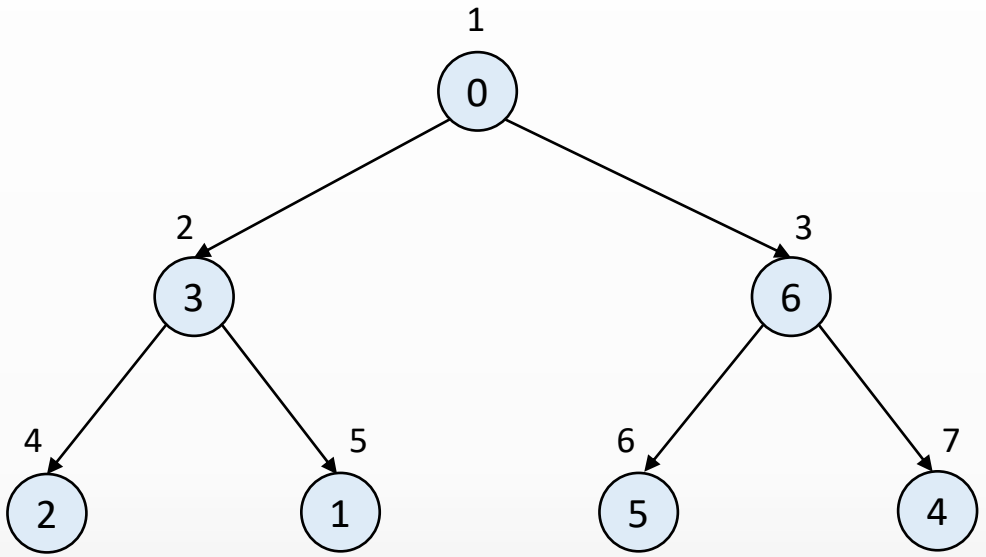


silMax()

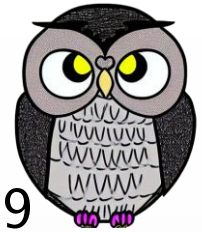
max = 9

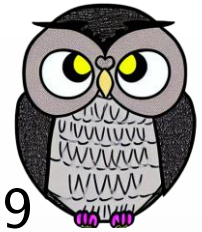


silMax()



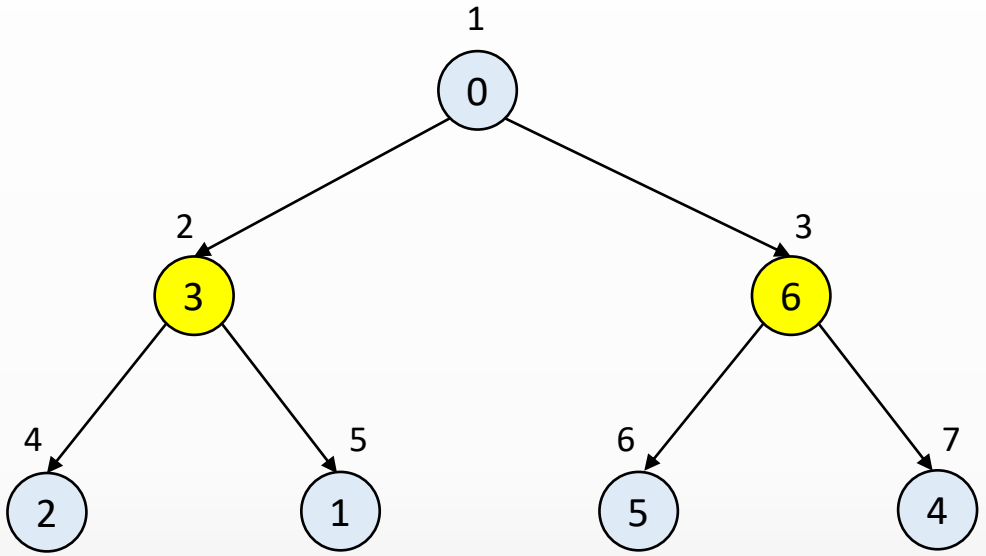
max = 9

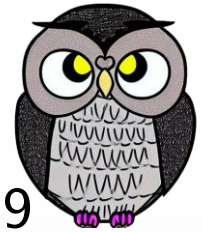




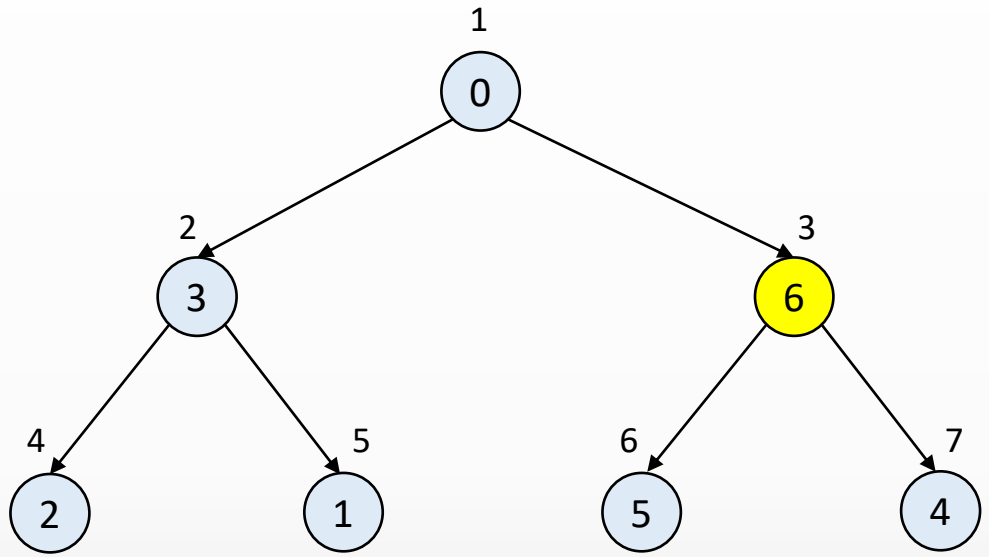
silMax()

max = 9



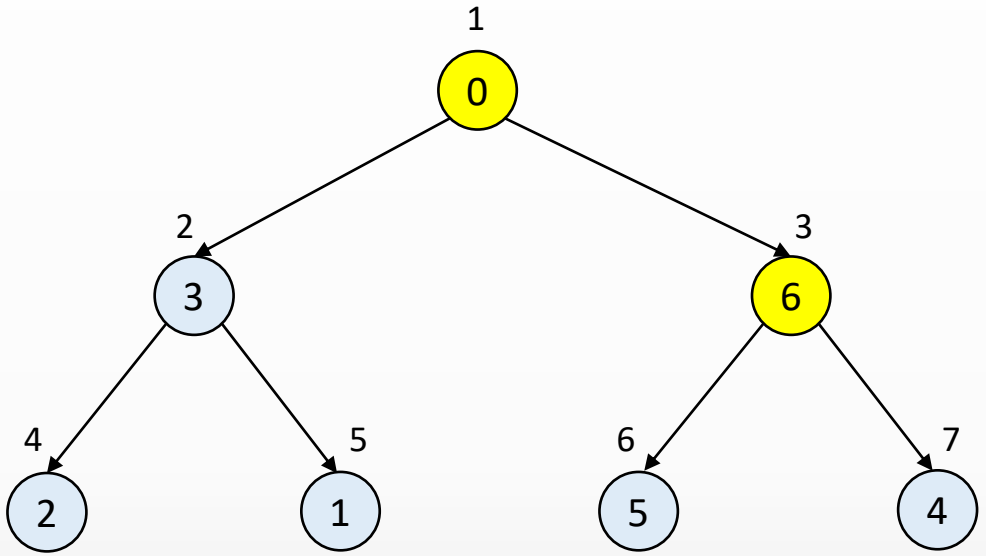


```
silMax()
```

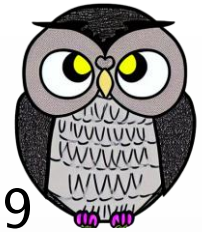



```
max = 9
```

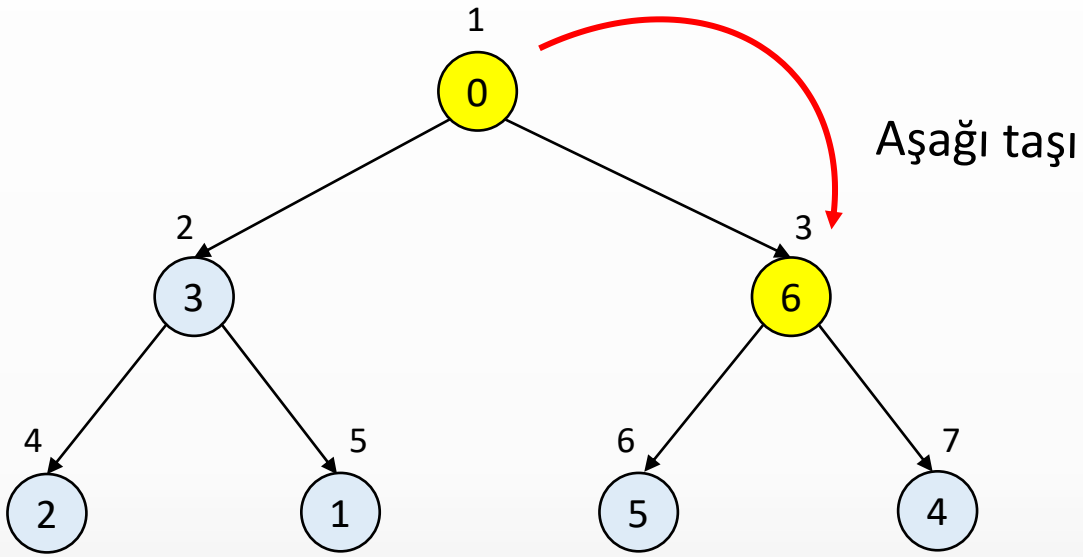
```
silMax()
```



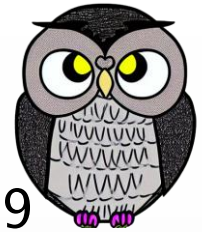

```
max = 9
```

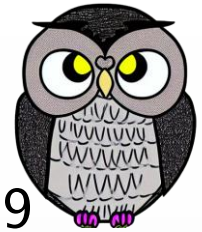


silMax()



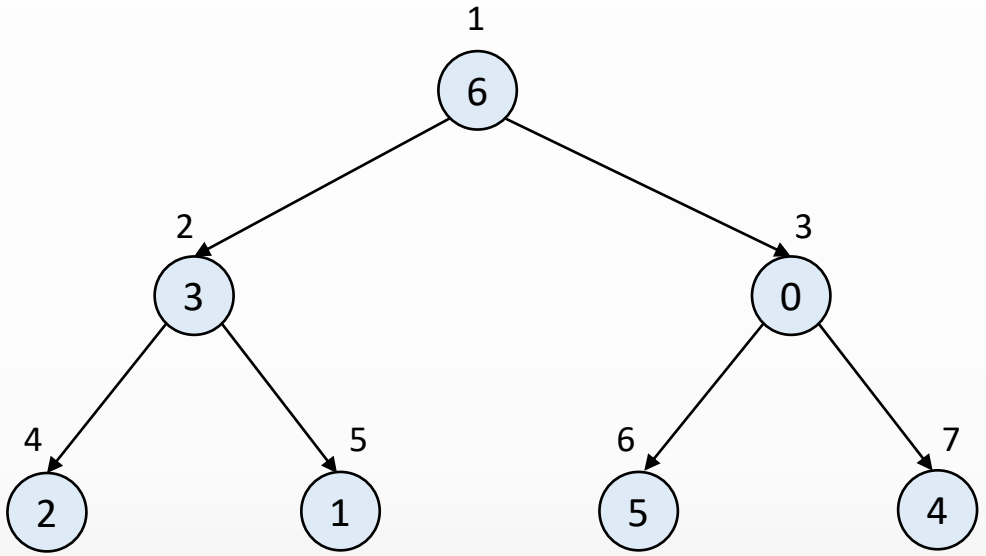
max = 9



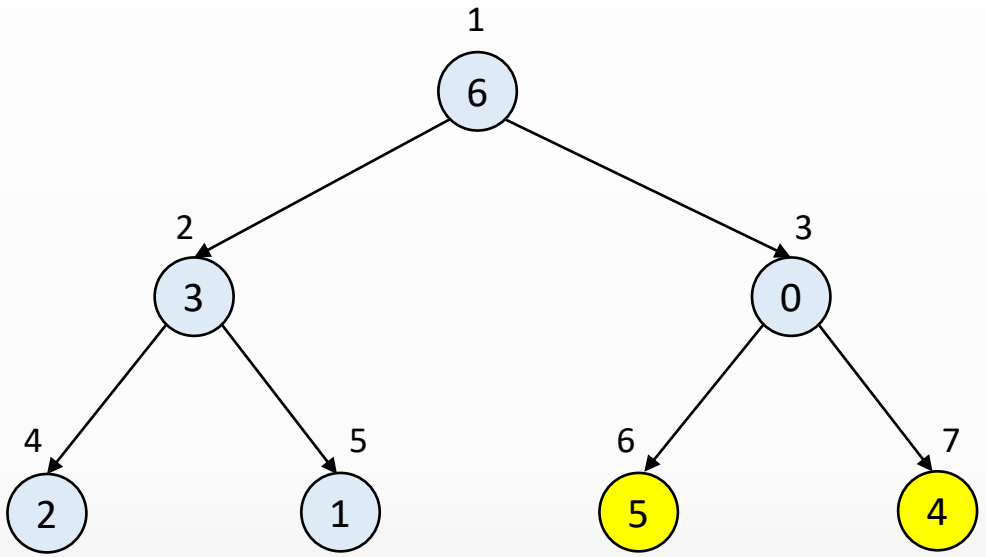


silMax()

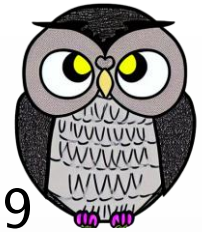
max = 9



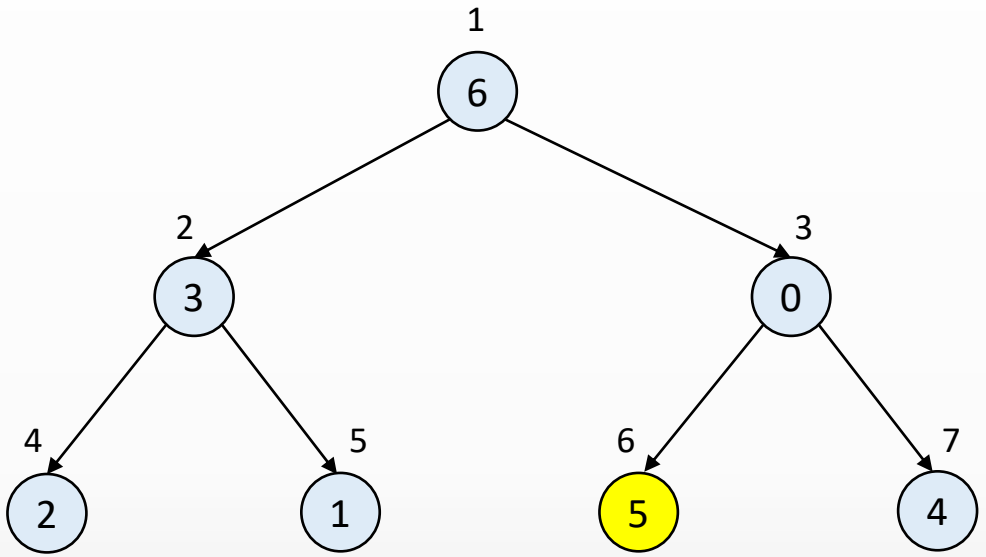
silMax()



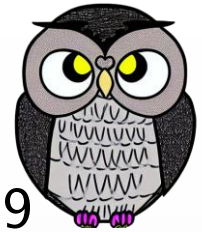
max = 9

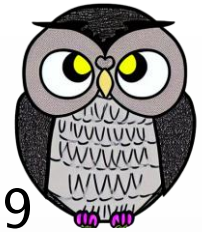


silMax()



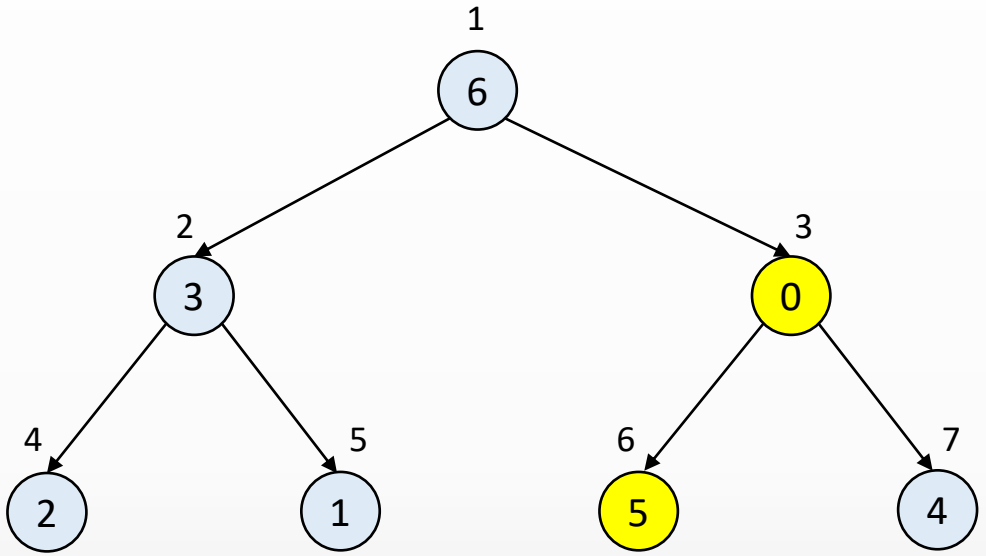
max = 9





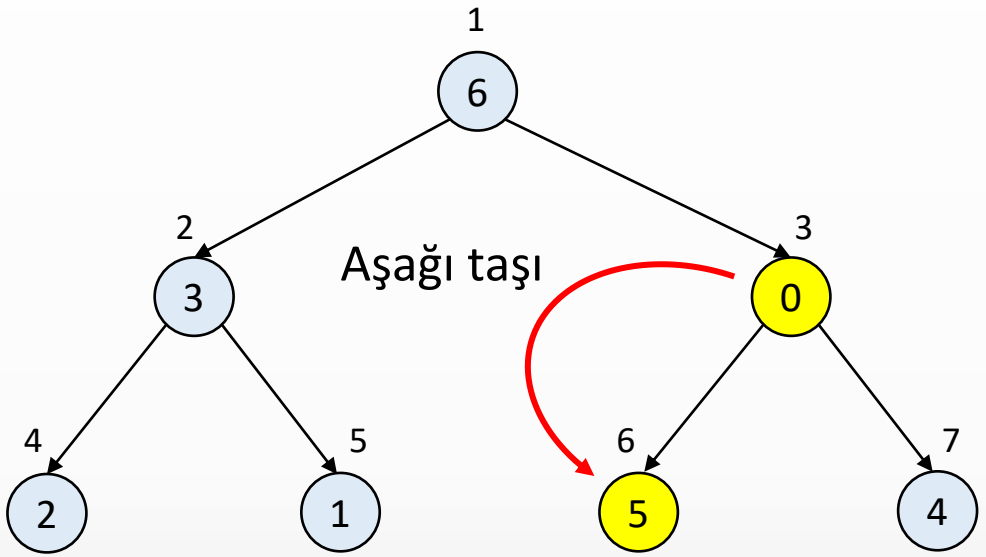
silMax()

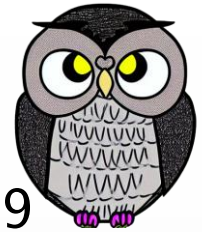
max = 9



silMax()

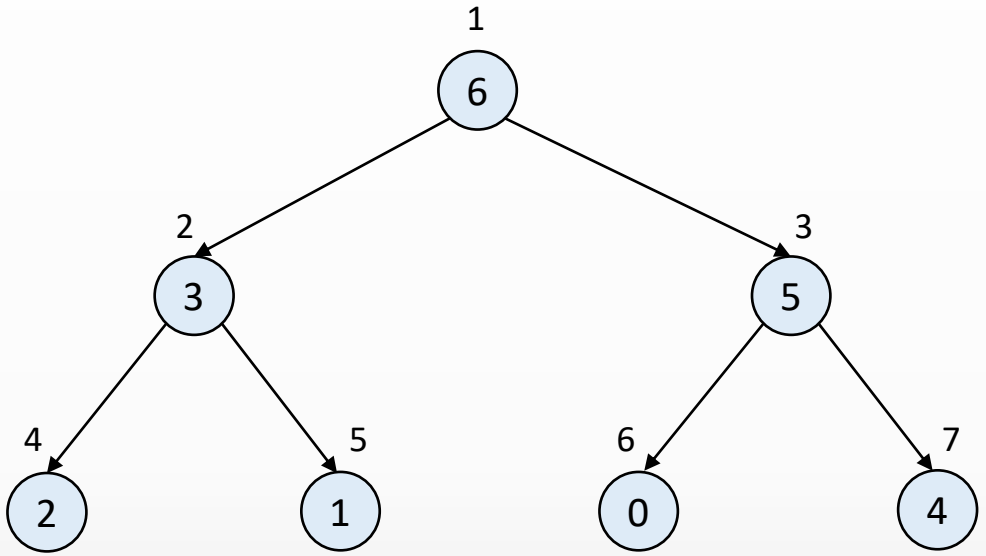
max = 9





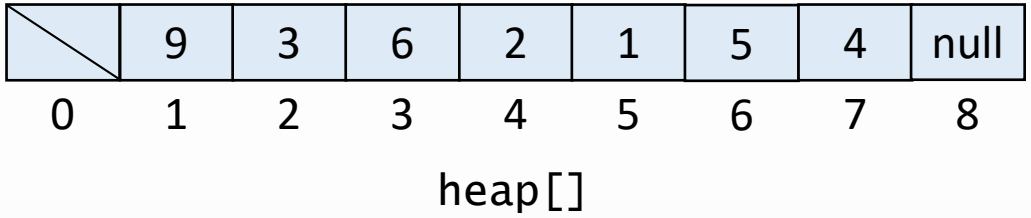
silMax()

max = 9

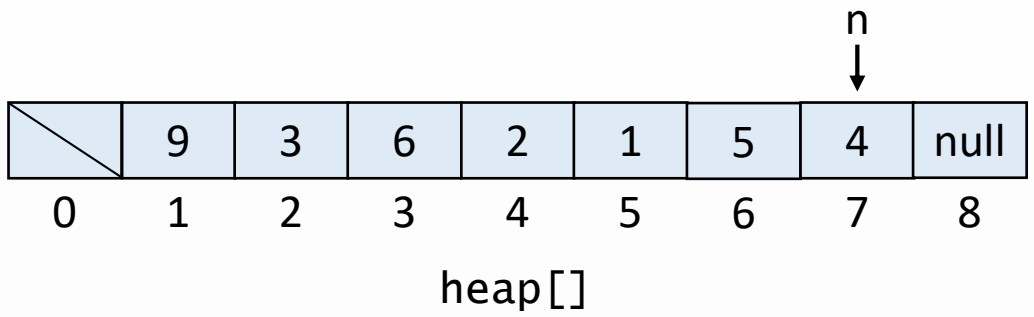


Max Heap Ağacında En Büyük Elemanı Silme

Sercan KÜLCÜ, Tüm hakları saklıdır.

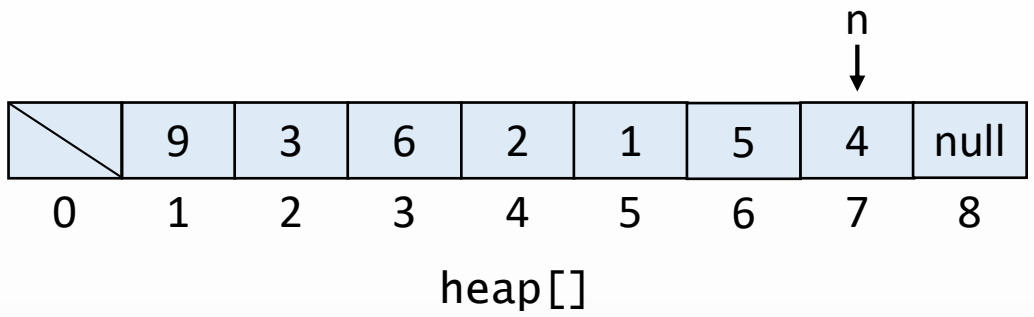


```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



n = 7

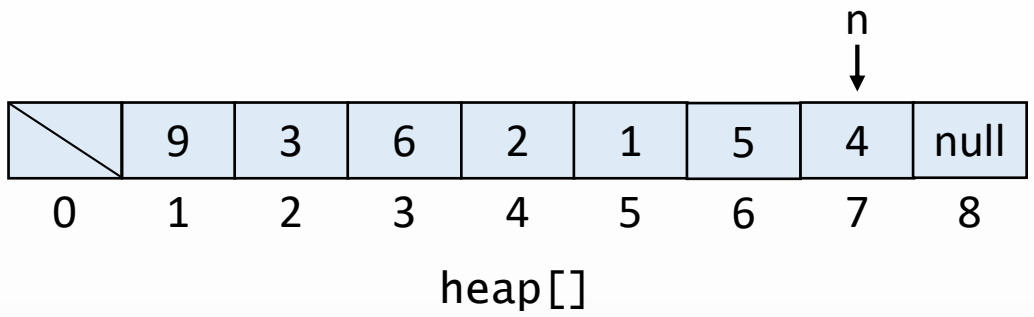
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



n = 7

silMax()

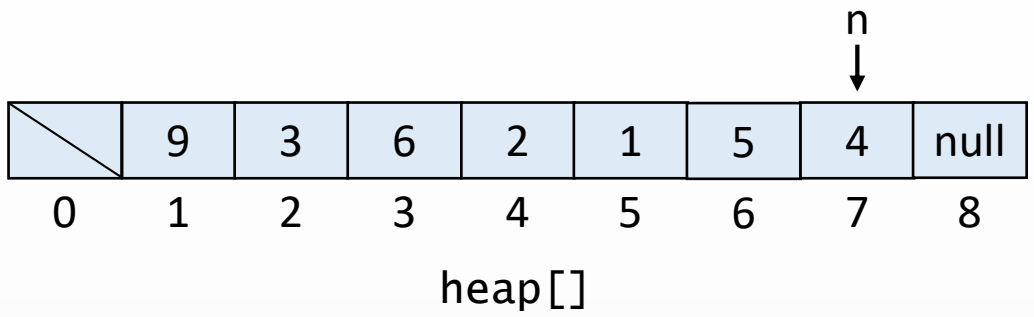
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



n = 7

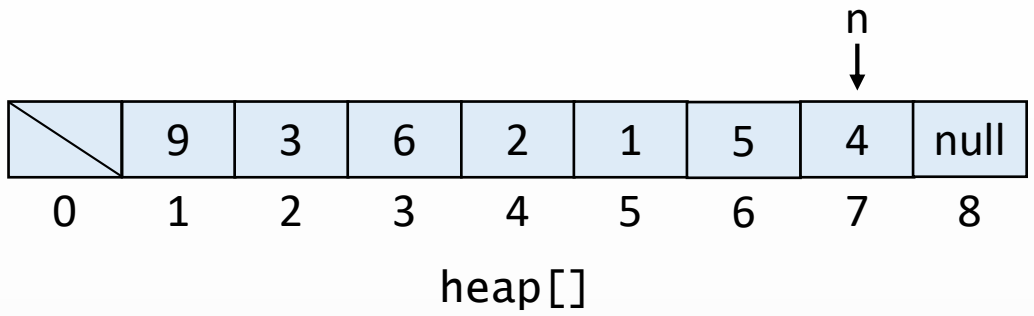
silMax()

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



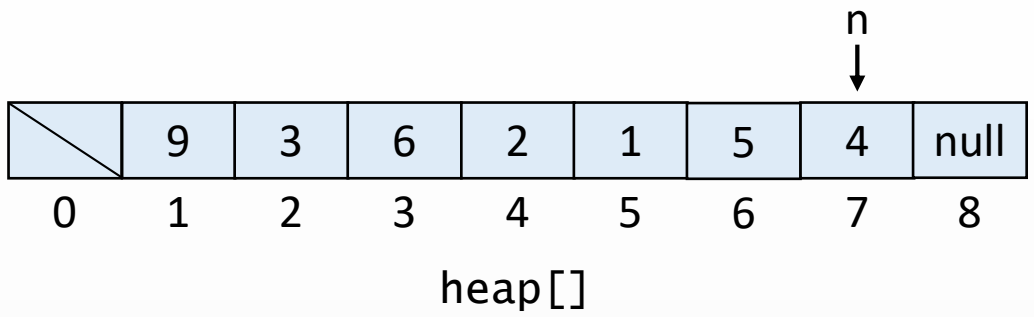
```
max = 9
n = 7


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



```
max = 9
n = 7


silMax()
```

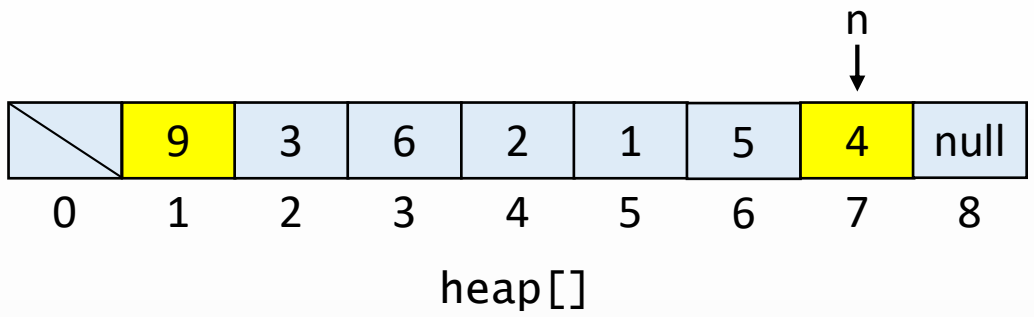
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



max = 9
n = 7

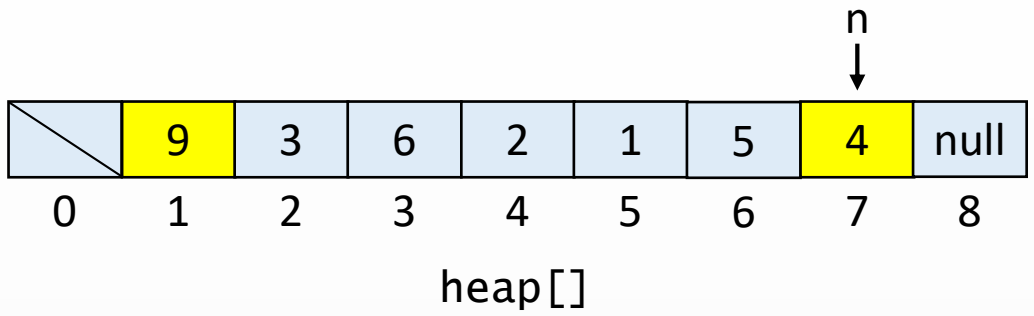
silMax()

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



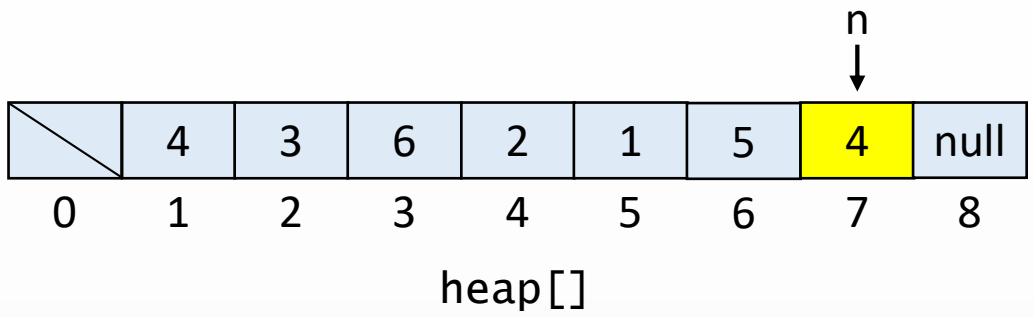
```
b = 7
a = 1
max = 9
n = 7


silMax()
```

```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



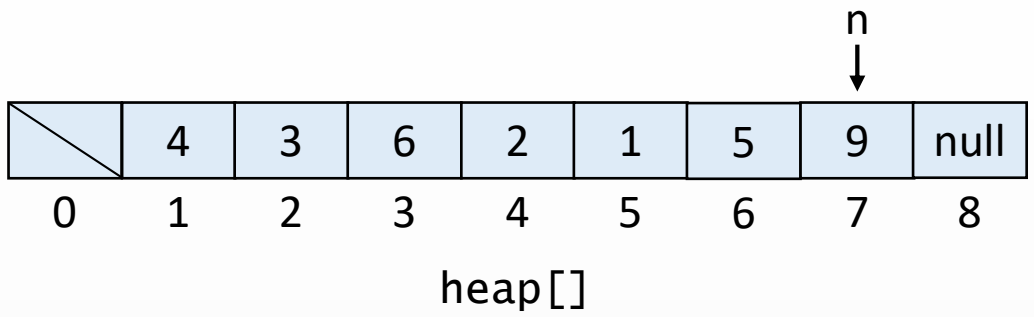
```
gecici = 9
b = 7
a = 1
max = 9
n = 7


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



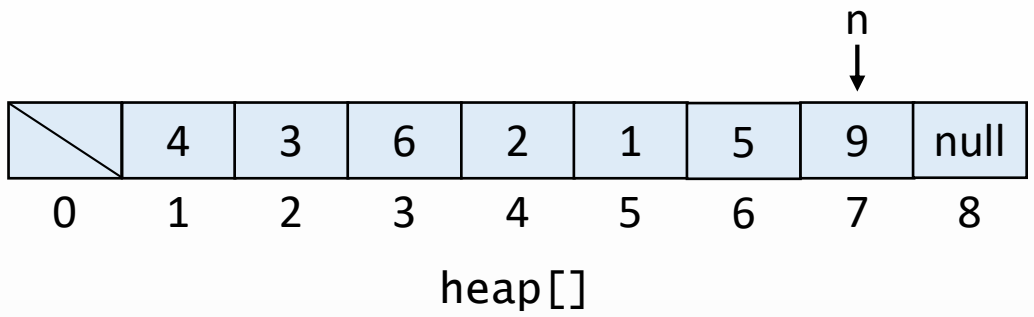
```
gecici = 9
b = 7
a = 1
max = 9
n = 7


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



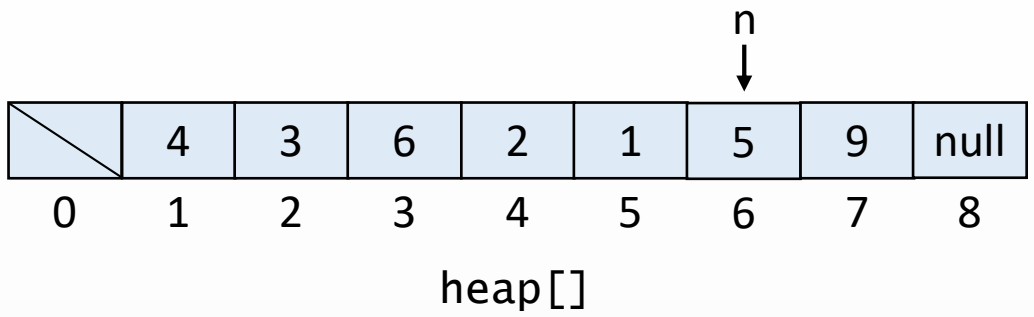
```
gecici = 9
b = 7
a = 1
max = 9
n = 7


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



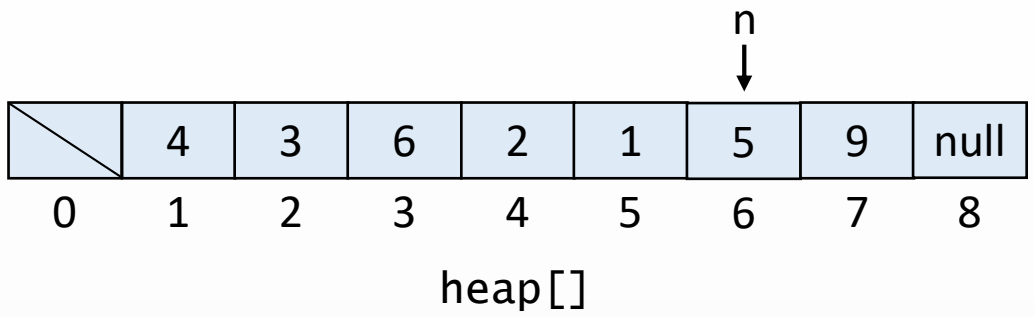
```
max = 9
n = 7


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



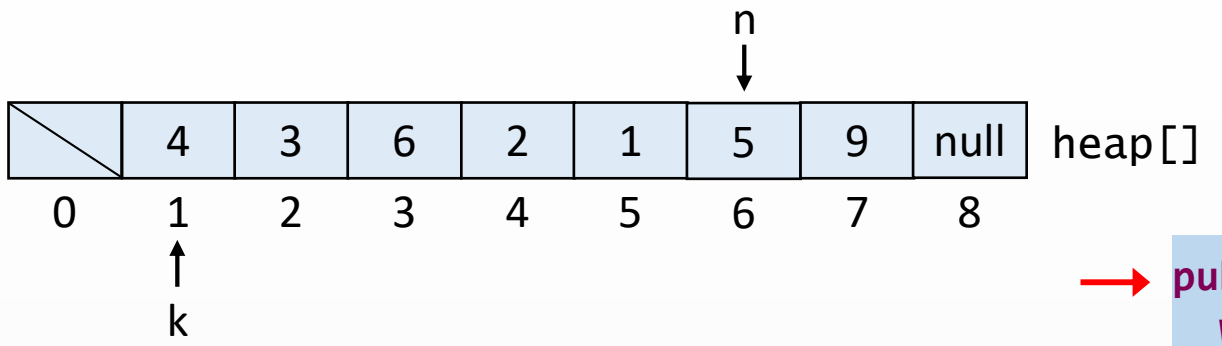
```
max = 9
n = 6


silMax()
```

```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



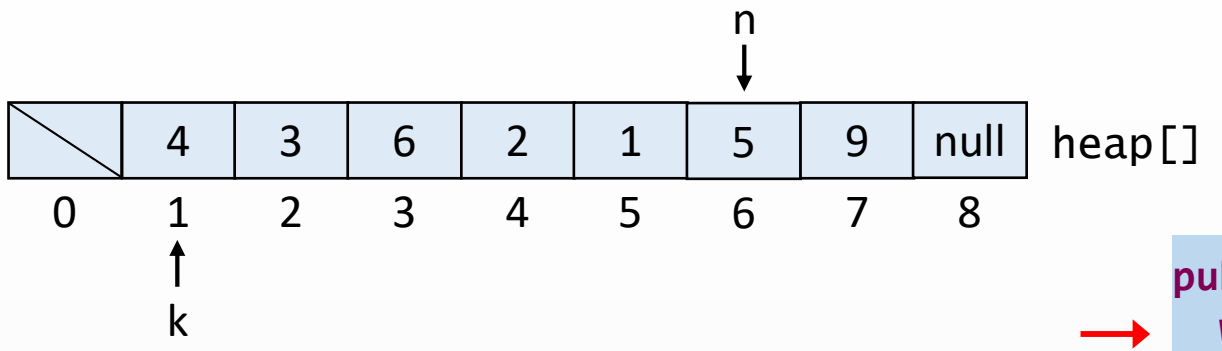
```
max = 9
n = 6


silMax()
```

```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



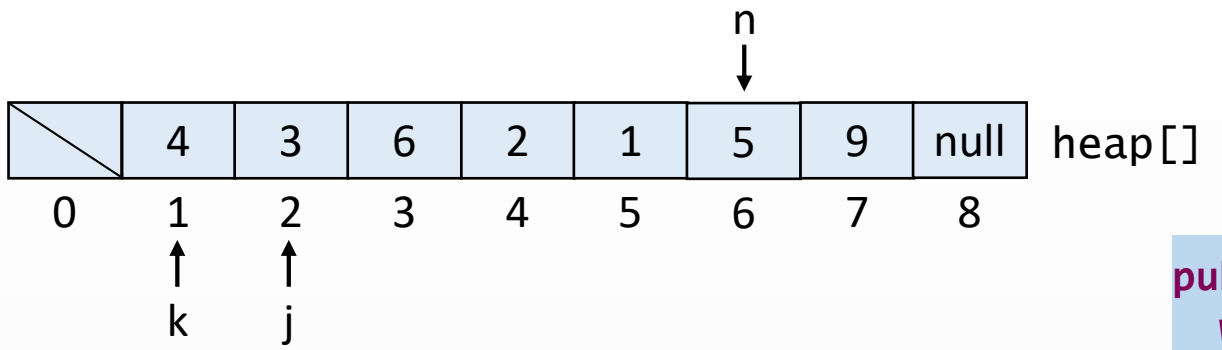

```
k = 1
max = 9
n = 6


silMax()
```

```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



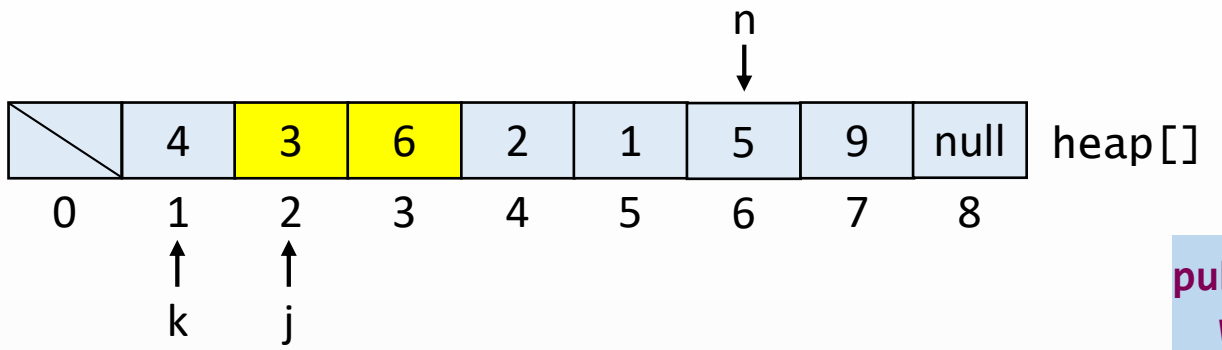

```
k = 1
max = 9
n = 6


silMax()
```

```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



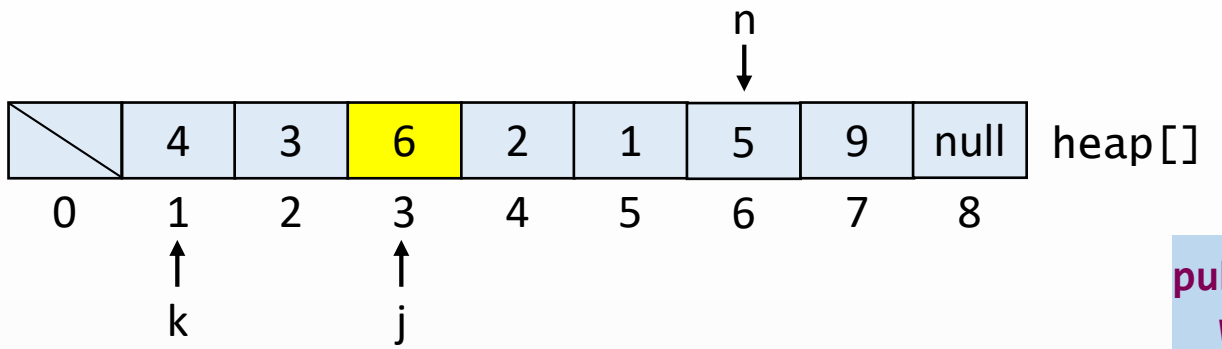
```
j = 2
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



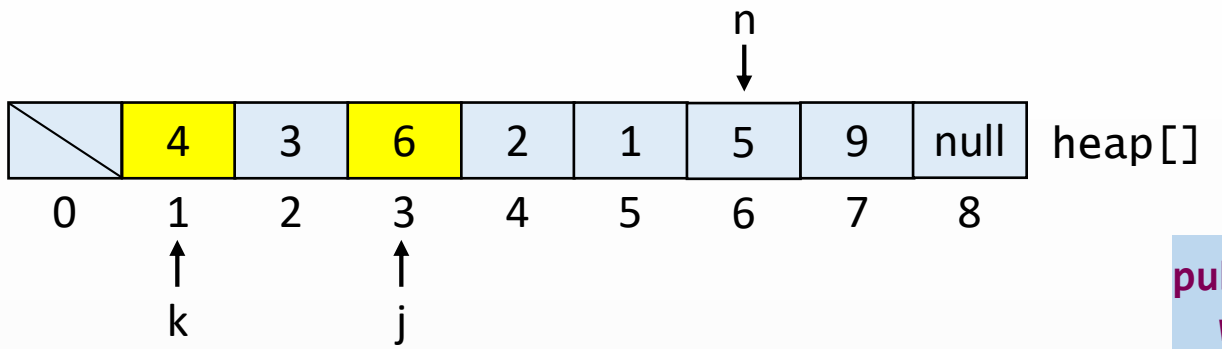
```
j = 2
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



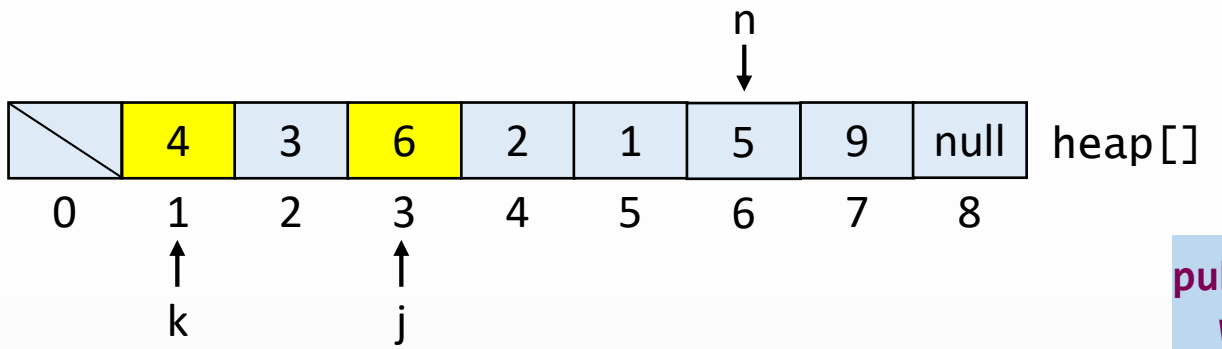
```
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



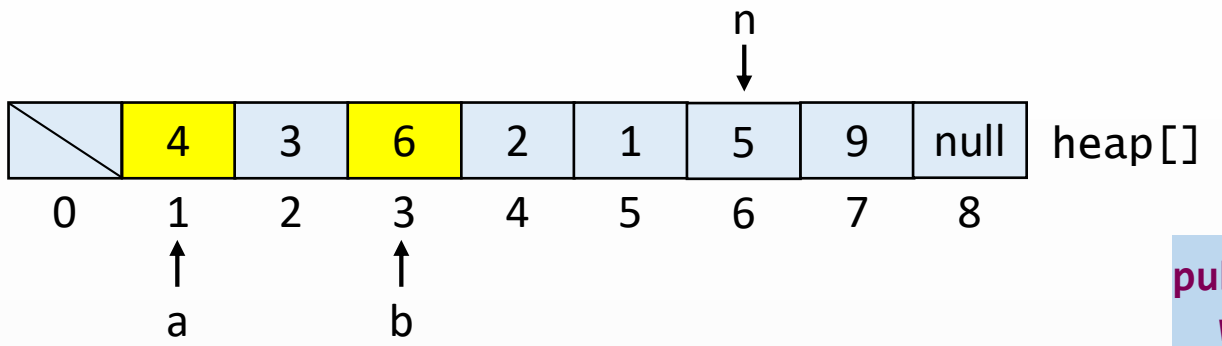
```
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



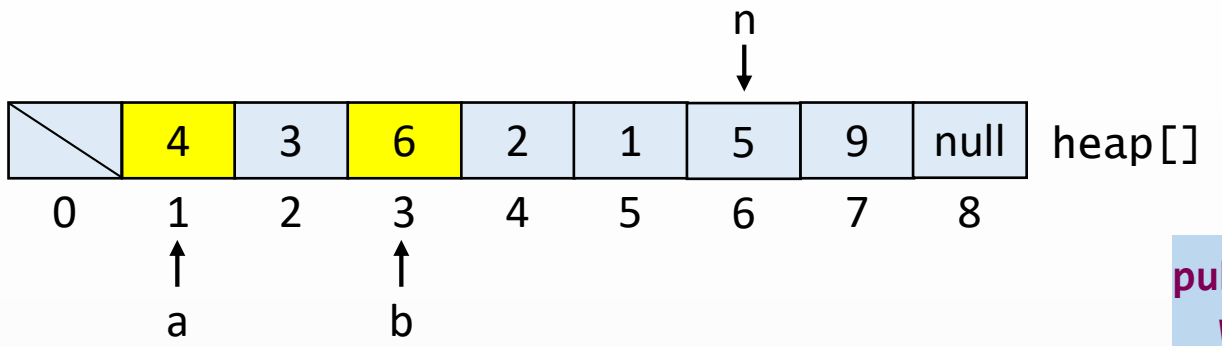

```
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



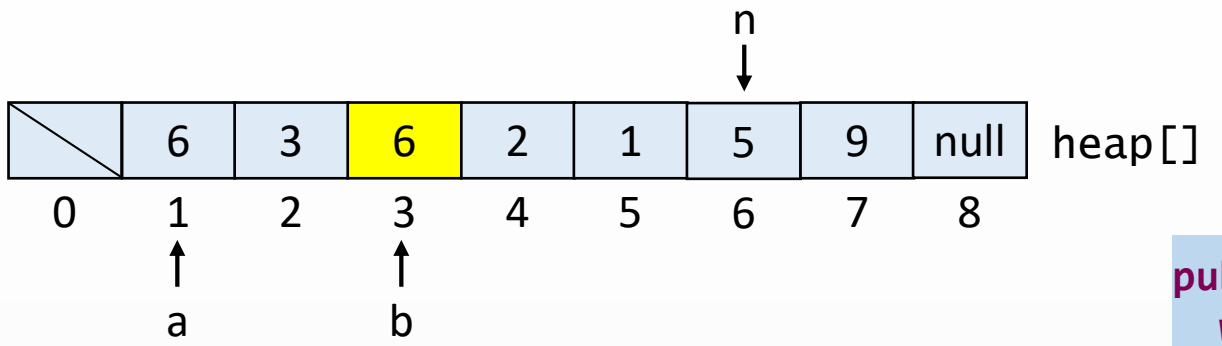
```
b = 3
a = 1
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



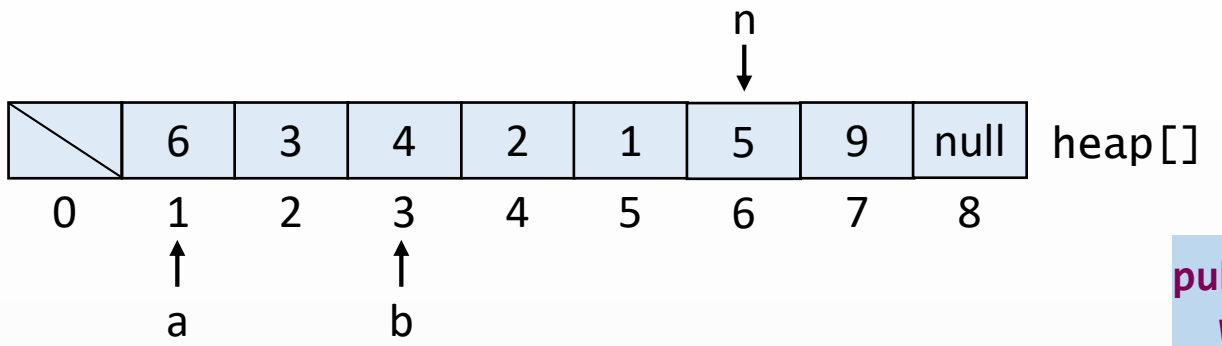
```
gecici = 4
b = 3
a = 1
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



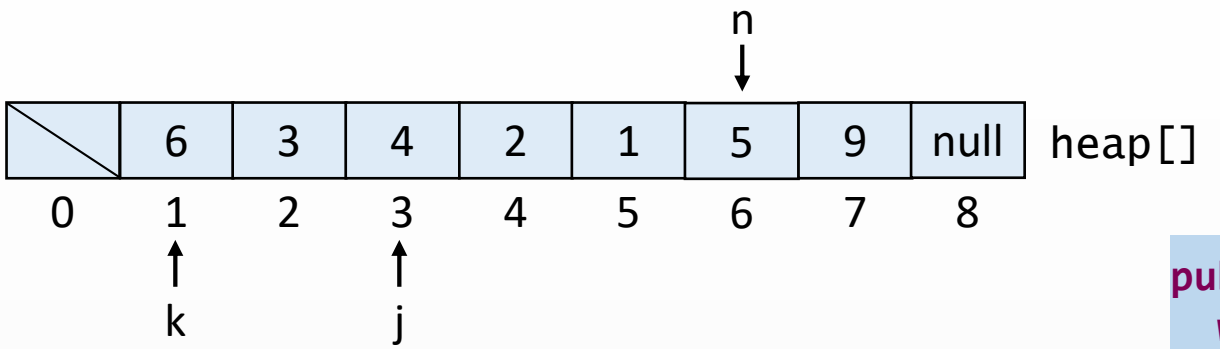
```
gecici = 4
b = 3
a = 1
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



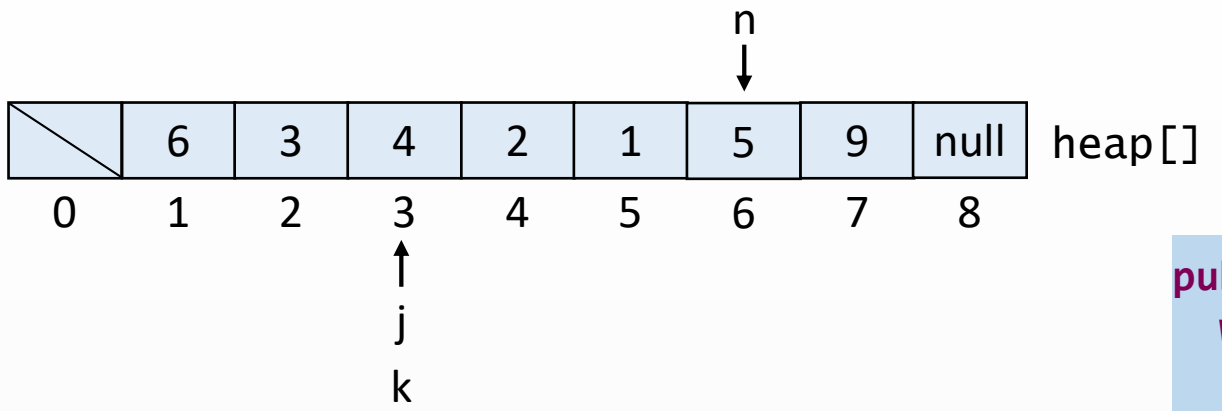
```
gecici = 4
b = 3
a = 1
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



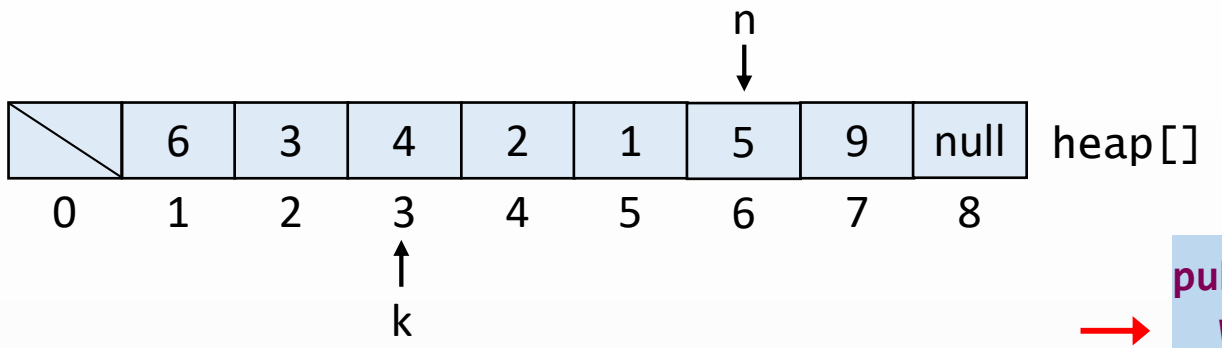
```
j = 3
k = 1
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



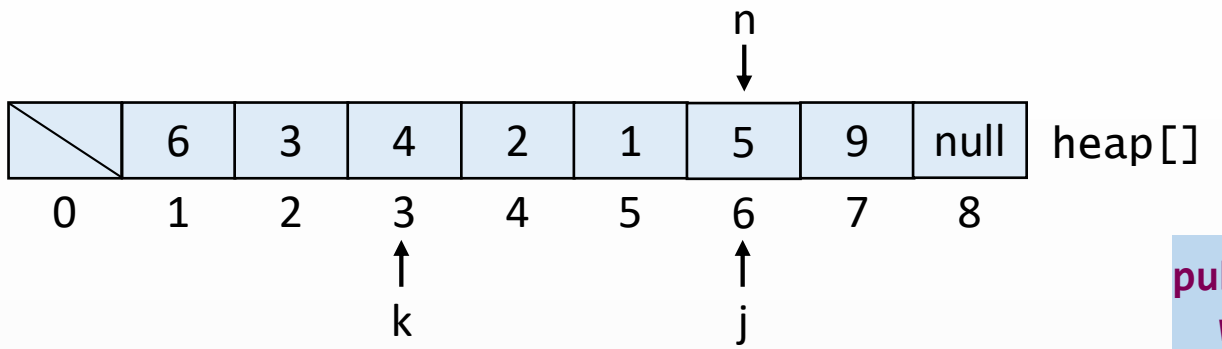

```
j = 3
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



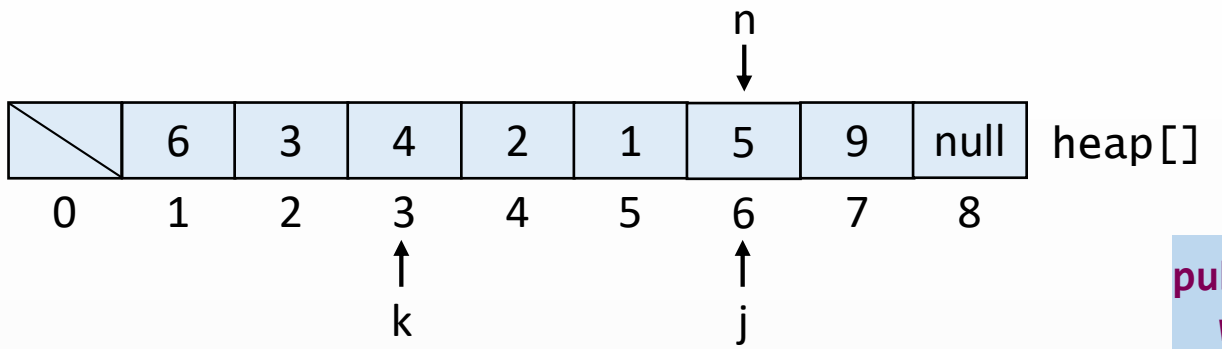
```
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



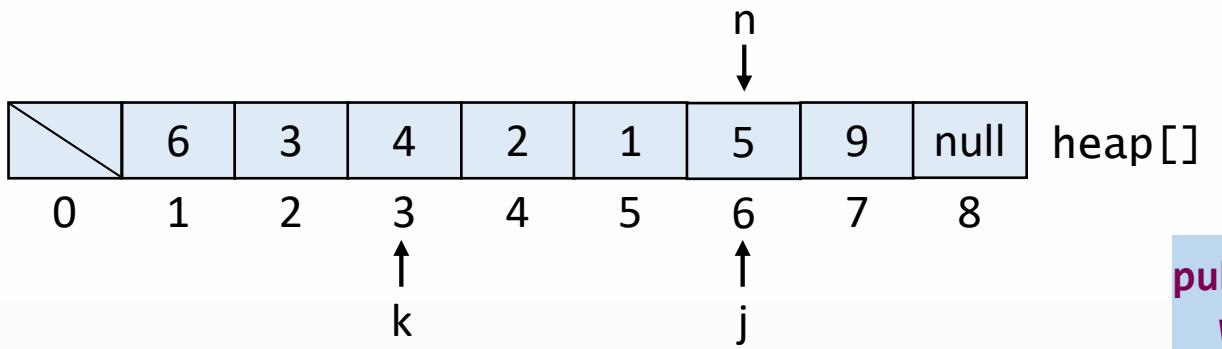

```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



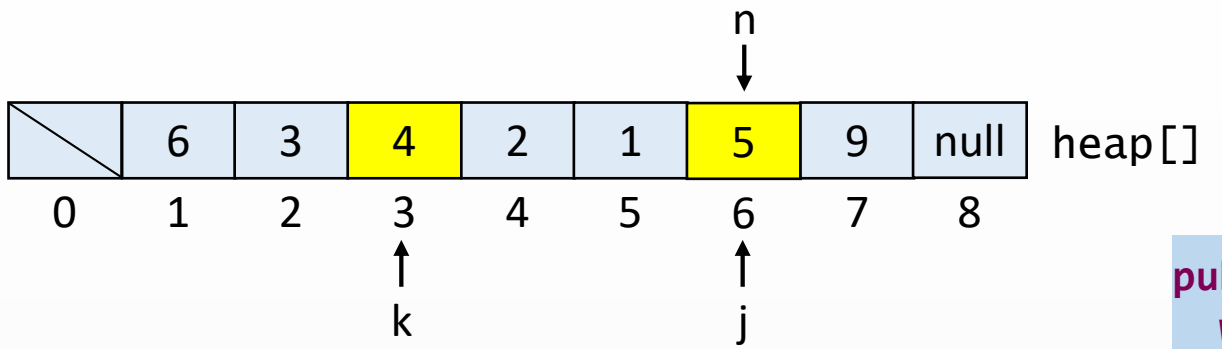
```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



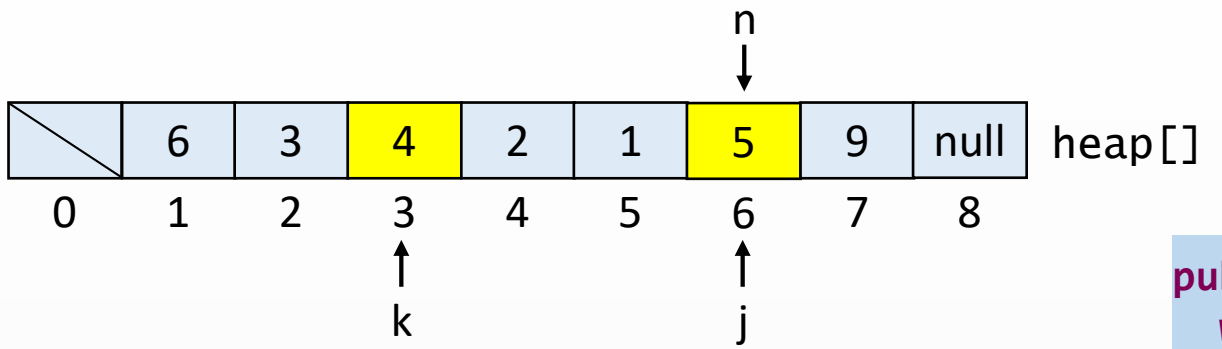
```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



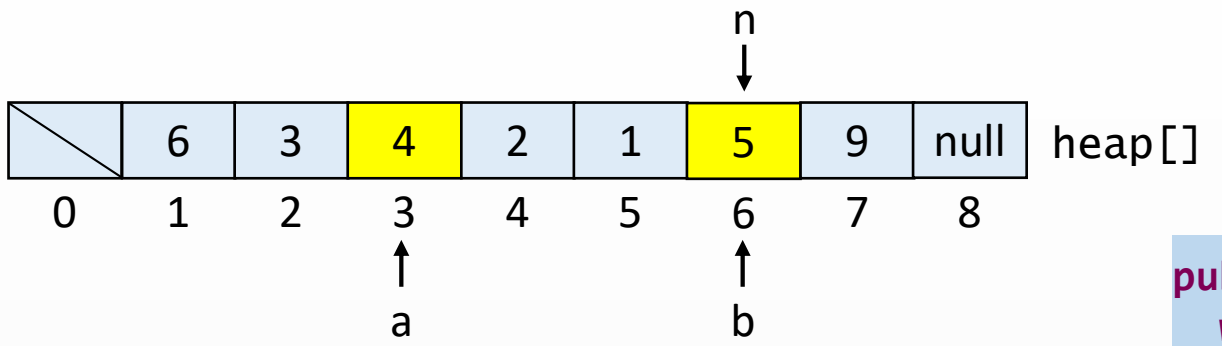

```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



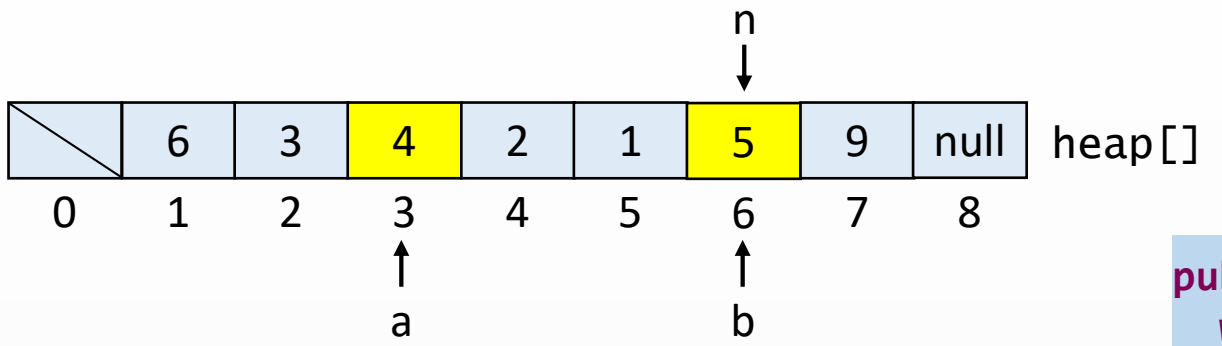

```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



```
b = 5
a = 4
j = 6
k = 3
max = 9
n = 6


silMax()
```

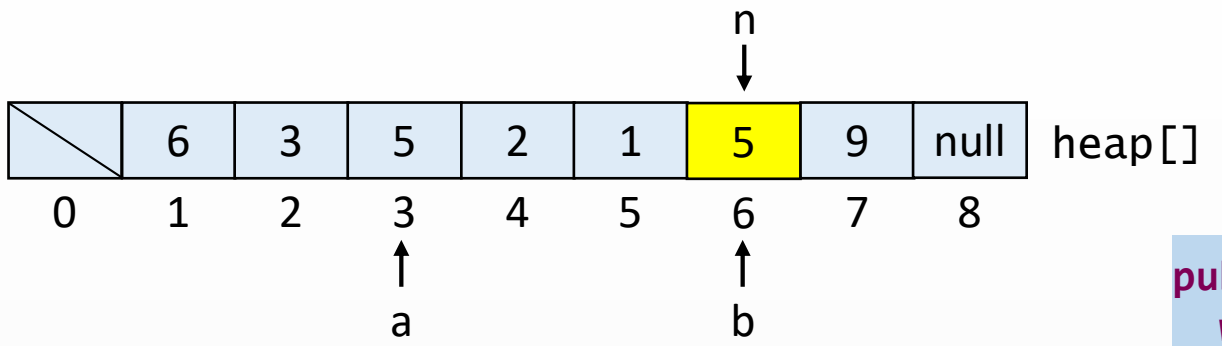
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



```
gecici = 4
b = 5
a = 4
j = 6
k = 3
max = 9
n = 6
```

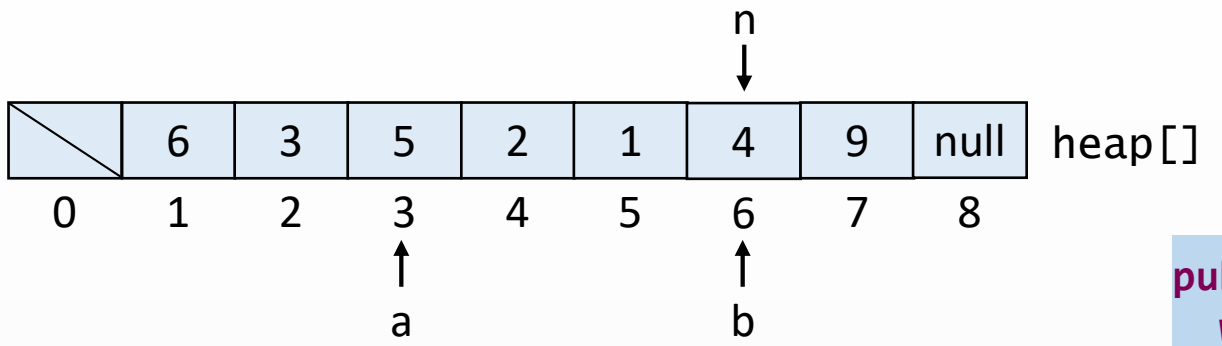
```
silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



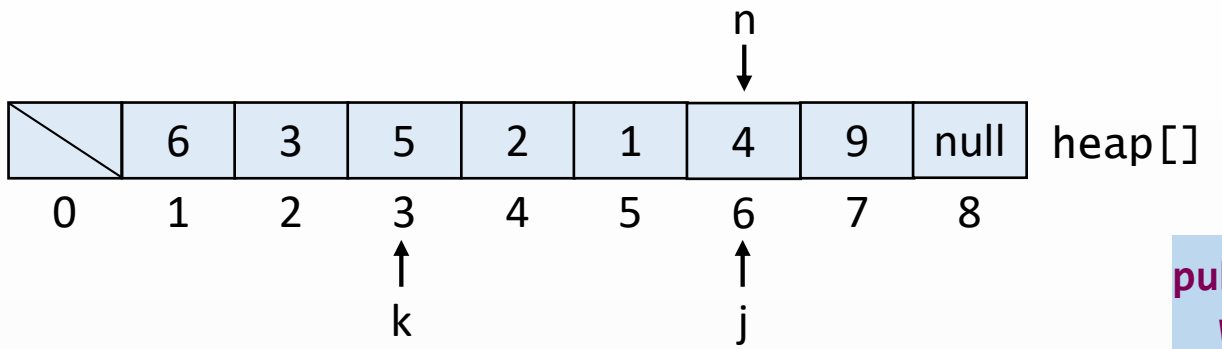
```
gecici = 4
b = 5
a = 4
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



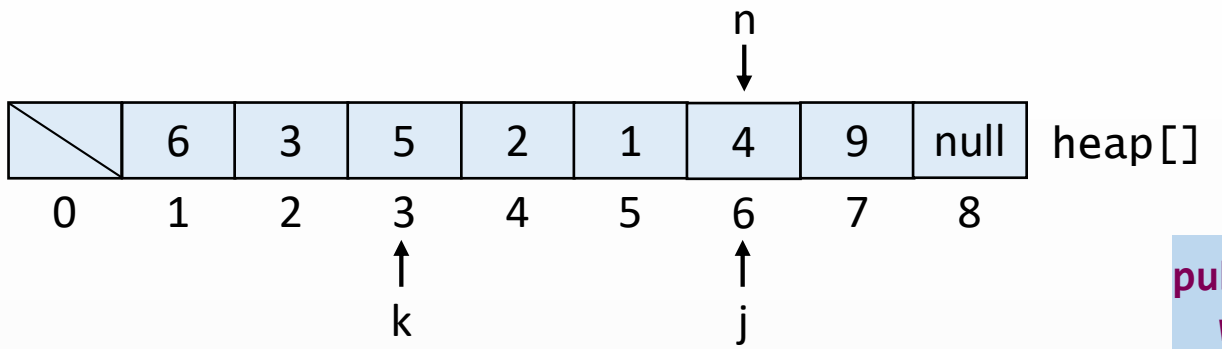
```
gecici = 4
b = 5
a = 4
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



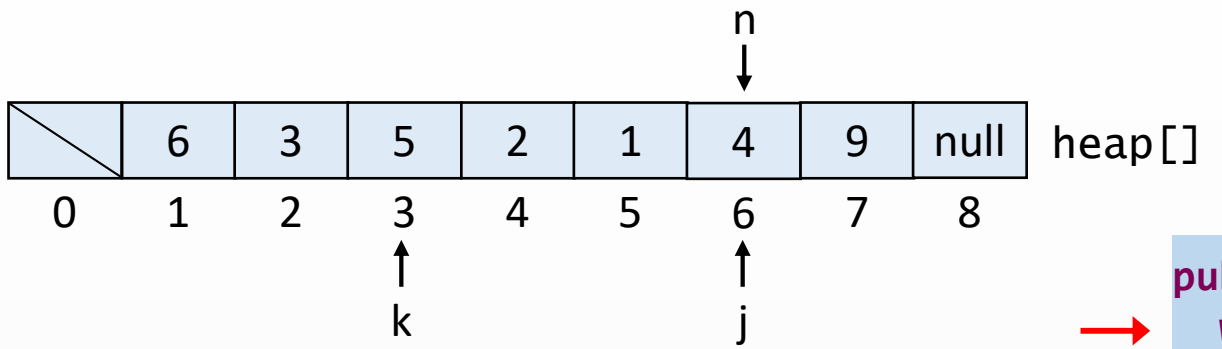
```
j = 6
k = 3
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



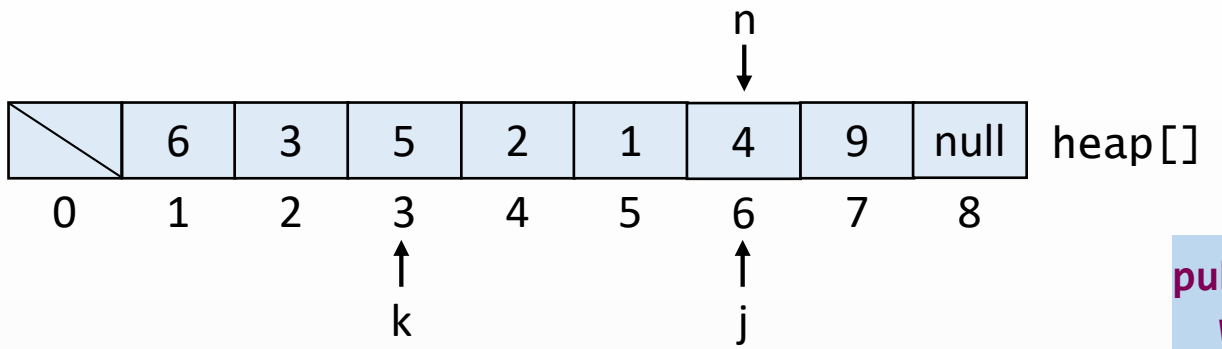
```
j = 6
k = 6
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



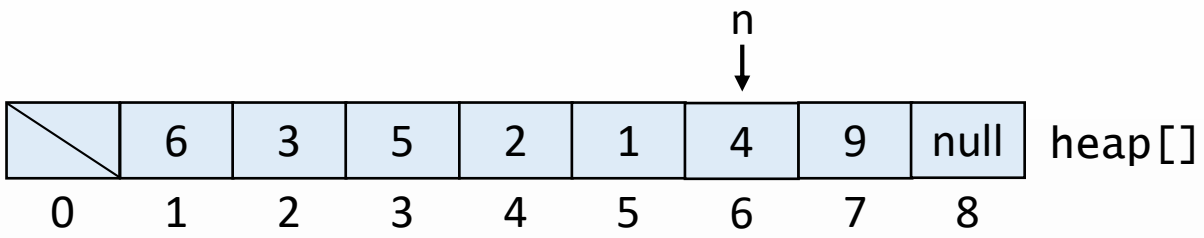
```
j = 6
k = 6
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



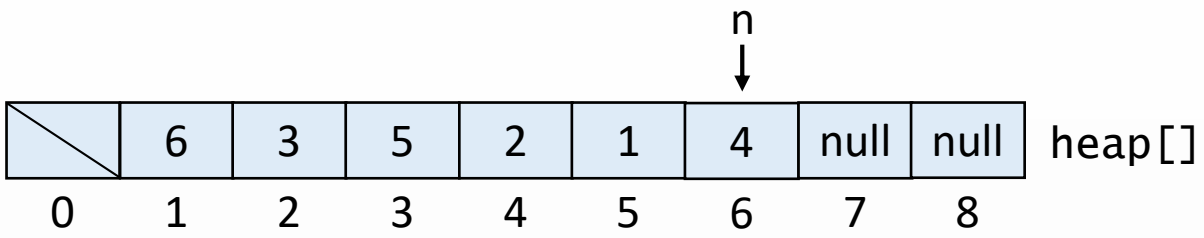

```
j = 6
k = 6
max = 9
n = 6


silMax()
```

```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



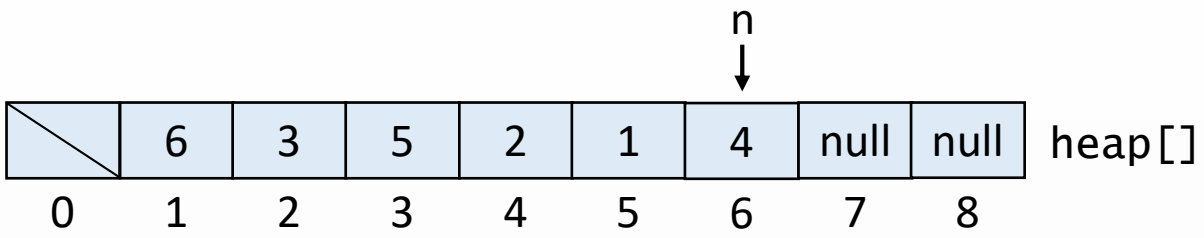
```
max = 9
n = 6


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



```
max = 9
n = 6


silMax()
```

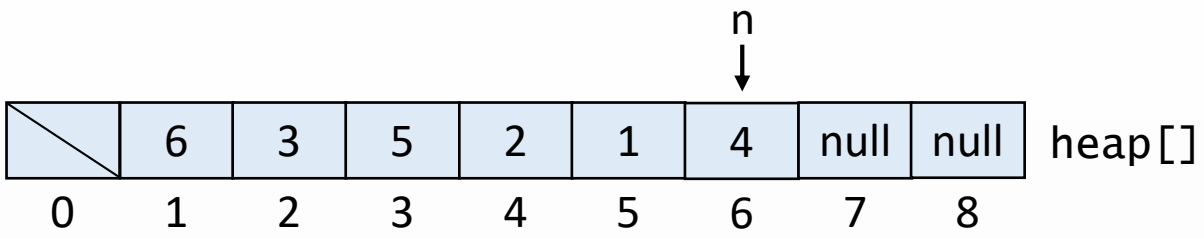
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



max = 9
n = 6

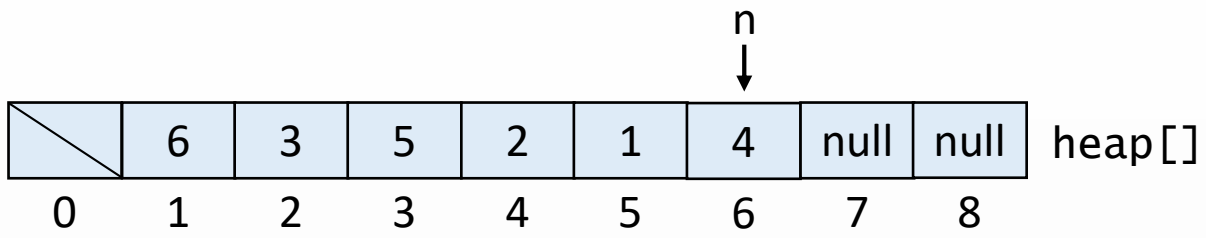
silMax()

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



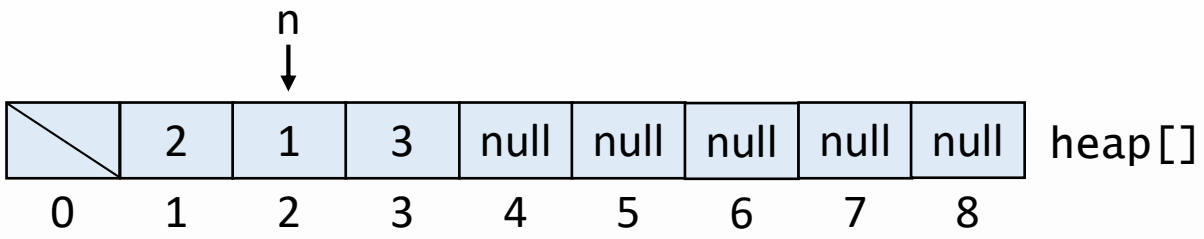
```
max = 9
n = 6


silMax()
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



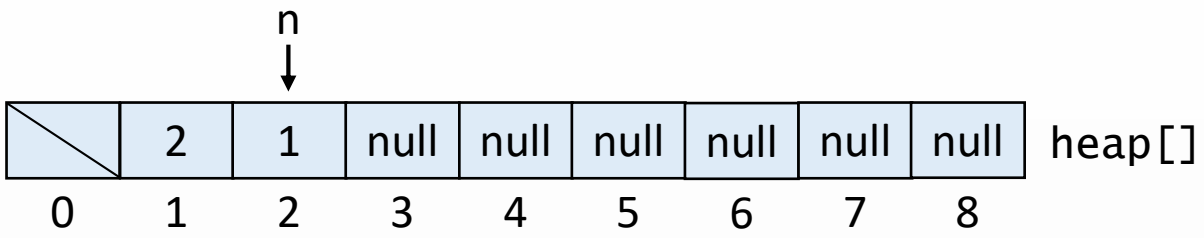
n = 6

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



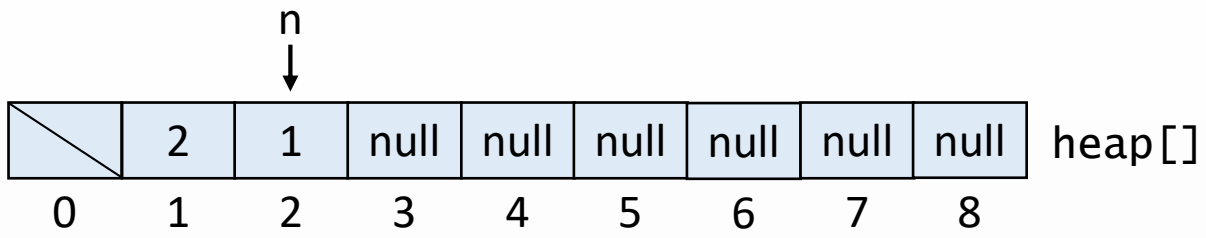
```
max = 3
n = 2
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



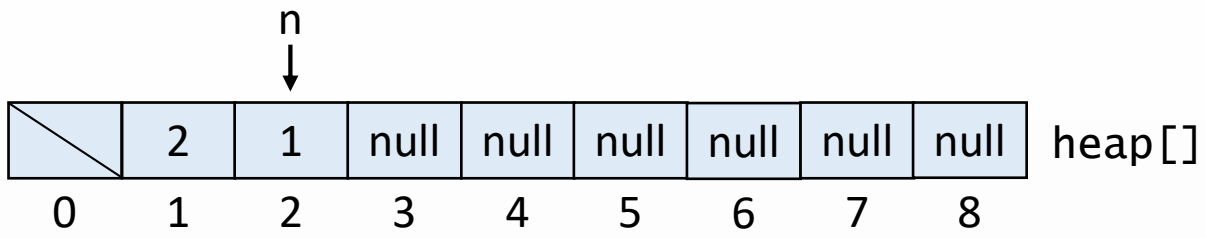
```
max = 3
n = 2
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



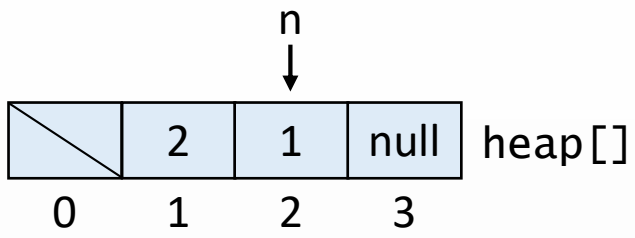
```
max = 3
n = 2
```

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



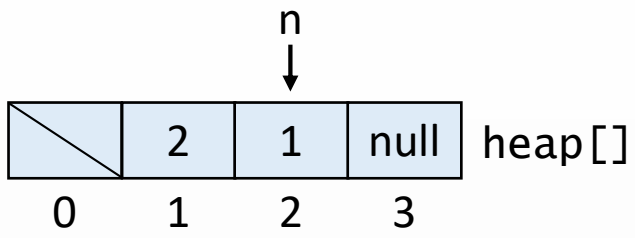
```
max = 3
n = 2
```

```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



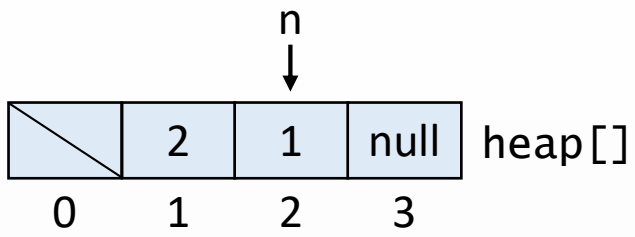
max = 3
n = 2

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



max = 3
n = 2

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```



n = 2

```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

public void yerDegistir(int a, int b) {
  int gecici = heap[a];
  heap[a] = heap[b];
  heap[b] = gecici;
}
```

SON

Sercan KÜLCÜ, Tüm hakları saklıdır.