Bölüm 8: Öncelikli Kuyruk

Veri Yapıları

Öncelik Kuyruğu (Priority Queue)

- Verilerin öncelik sırasına göre saklandığı bir veri yapısıdır.
- Özellikle öncelikli işlemlerin yönetiminde sıkça kullanılır.
- En yüksek öncelikli öğeyi çıkarmak için sadece O(1) zaman gerekir.

Temel Kavramlar

- **Öncelik Kuyruğu**: Verilerin saklandığı yapı.
- **Öncelik**: Her veriye atanan bir öncelik değeri.
- **En Yüksek Öncelik**: Kuyruğun başında bulunan en önemli veri.
- **FIFO İlkesi**: Aynı öncelikteki veriler arasında sıra.

Kullanım Alanları

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

Temel İşlemler

- **Ekleme (Insertion):** Veri eklerken önceliğe göre sıralama yapar.
- **Çıkarma (Extraction):** En yüksek öncelikli veriyi çıkarır.
- **Sorgulama (Peek):** En yüksek öncelikli veriyi döndürür.
- **Boş mu Kontrolü (isEmpty):** Kuyruk boş mu dolu mu?

Dizi Temelli Gerçekleştirim

- Öncelik kuyruğu basit bir şekilde dizi ile temsil edilebilir.
- Dizi öncelik sırasına göre sıralanarak, en yüksek öncelikli veri bulunabilir.
- Yeni öğeler eklemek ve öğeleri çıkarmak için sıralamayı güncellemek zor olabilir.

Bağlı Liste Temelli Gerçekleştirim

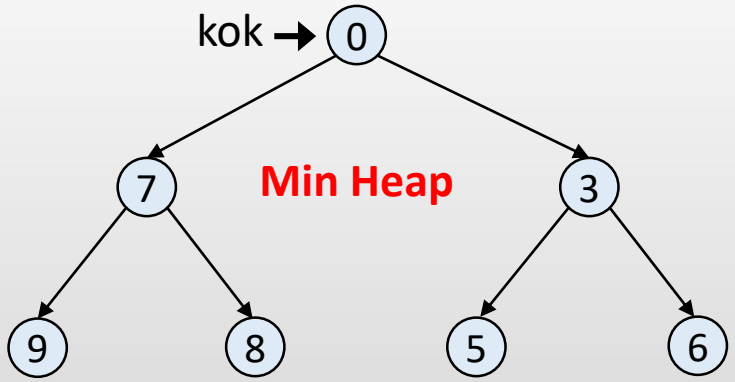
- Öncelik kuyruğu bağlı liste ile temsil edilebilir.
- Veriler önceliklerine göre sıralı bir şekilde bağlı listeye eklenebilir.
- En yüksek öncelikli öğeyi bulmak için bağlı listeyi taramak gerekebilir.

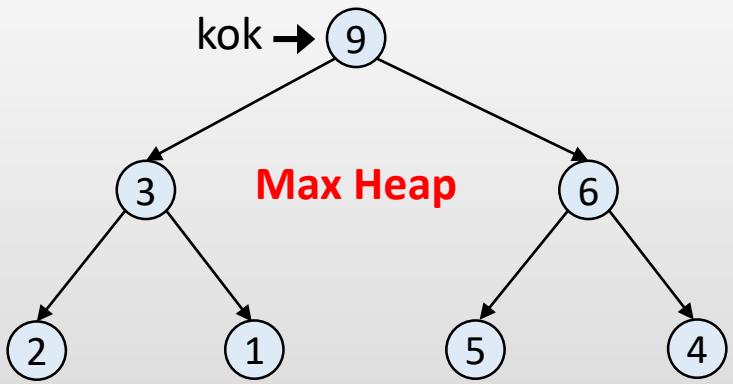
İkili Heap Gerçekleştirimi

- İkili heap (min-heap veya max-heap) öncelik kuyruğu uygulamaları için en yaygın kullanılan veri yapısıdır.
- En yüksek öncelikli öğe kökte bulunur.
- Ekleme ve öğe çıkarma işlemleri O(log n) zaman karmaşıklığına sahiptir.

İkili Heap

- İkili Heap, özel bir ikili ağaç yapısıdır.
- Min-Heap ve Max-Heap olmak üzere iki türü vardır.
- **Min-Heap:** En küçük öğe kökte, her alt ağaç da bir min-heap'tir.
- **Max-Heap:** En büyük öğe kökte, her alt ağaç da bir max-heap'tir.





Ekleme İşlemi

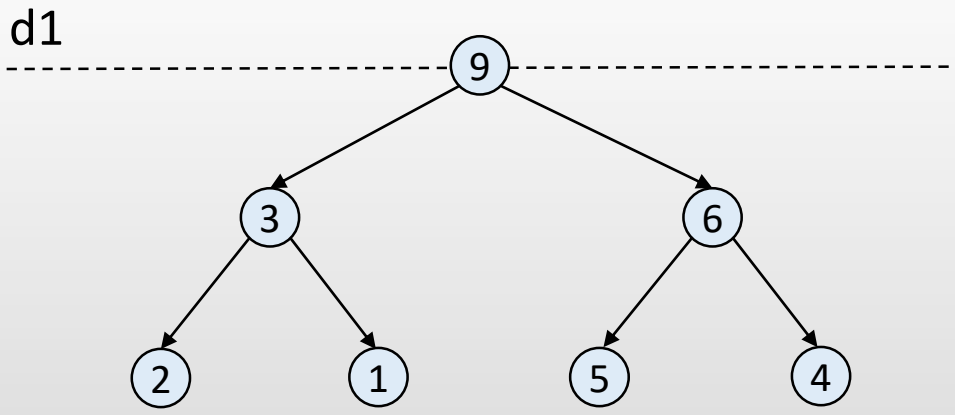
- Ekleme işlemi, ağacın sonuna yeni bir öğe ekler.
- Ekledikten sonra ağacın yapısı bozulmuş olabilir.
- Yapının yeniden dengelemesi için "heapify" adı verilen bir işlem yapılır.

Çıkarma İşlemi

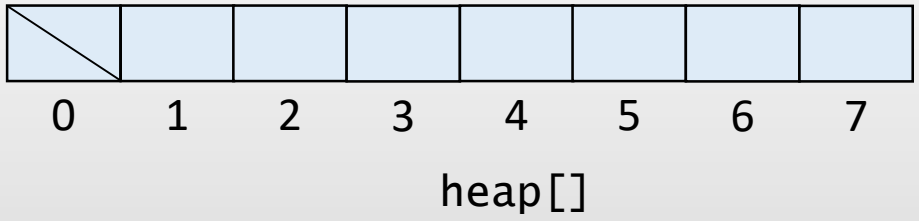
- Çıkarma işlemi, kökteki en küçük (veya en büyük) öğeyi çıkarır.
- Son öğeyi köke taşır ve "heapify" işlemi ile ağacın yapısını yeniden dengeler.
- İkili Heap, bu işlemi hızlı bir şekilde yapar (O(log n)).

İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.

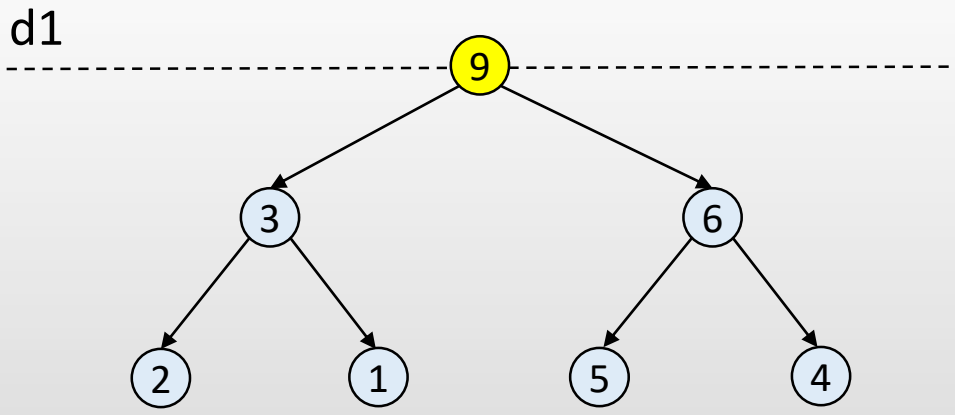


Max Heap

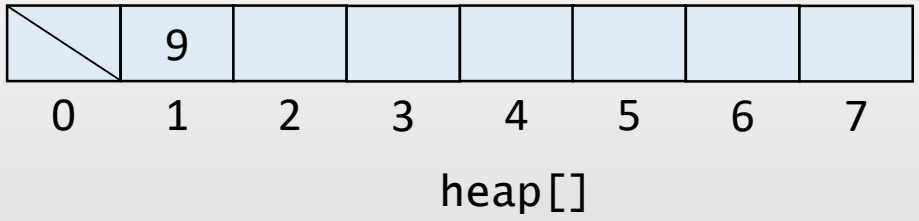


İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.

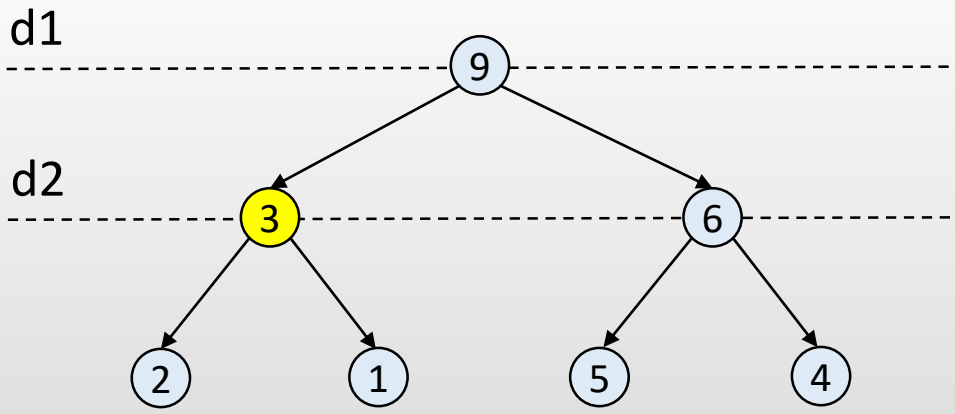


Max Heap



İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.



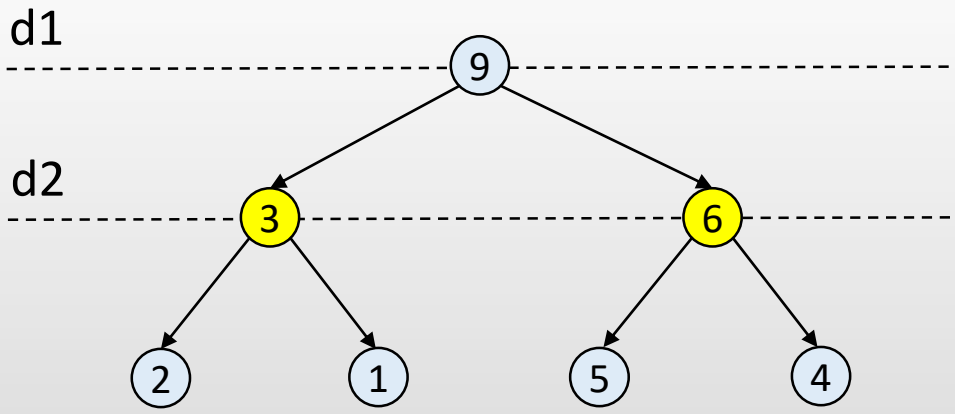
Max Heap

	9	3					
0	1	2	3	4	5	6	7

heap[]

İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.



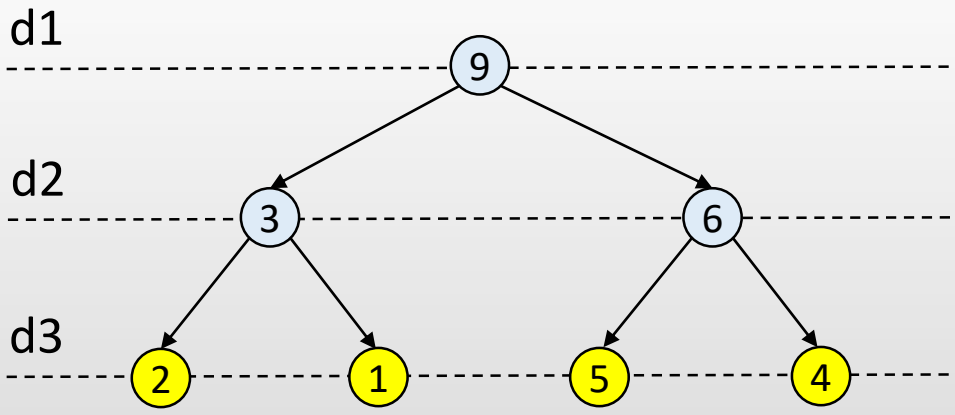
Max Heap

	9	3	6				
0	1	2	3	4	5	6	7

heap[]

İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.



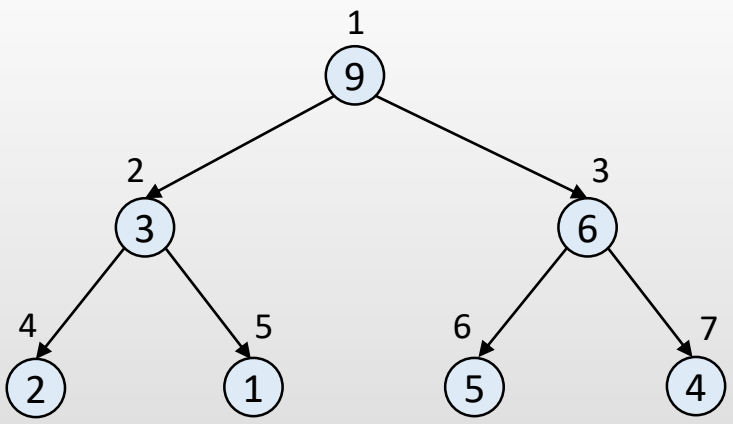
Max Heap

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

heap[]

İkili Heap Gösterimi

- İkili heap genellikle dizi veri yapısı ile gerçeklenir. İlk elemanı boş bırakılır.
- Tam ikili ağaçtır. Değerler soldan sağa doğru düzey ağaç dolaşımı ile dizi içinde saklanır.

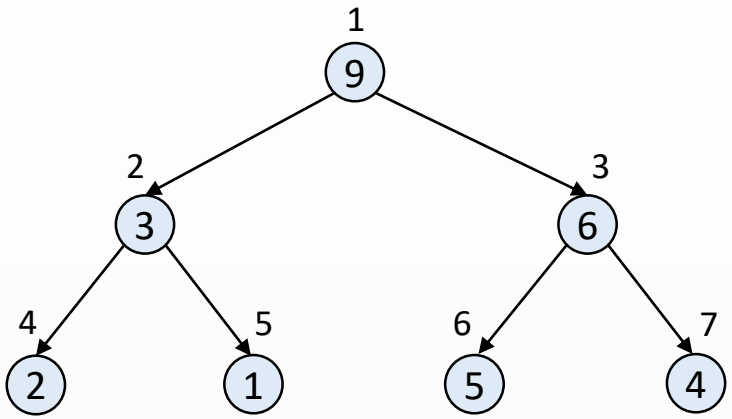


Max Heap

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

heap[]

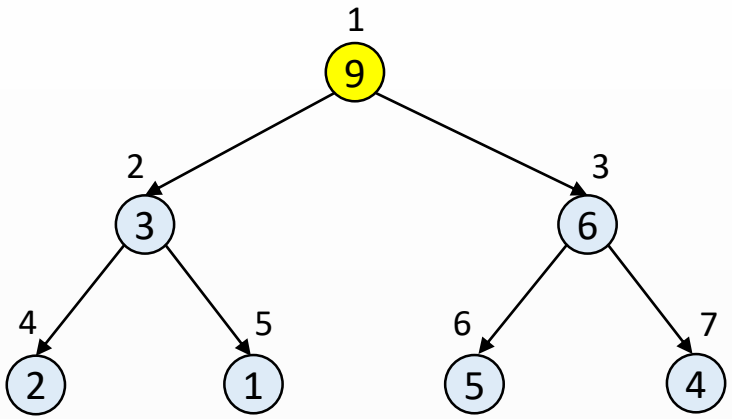
Ebeveyn ve Çocuk Hesaplamaları



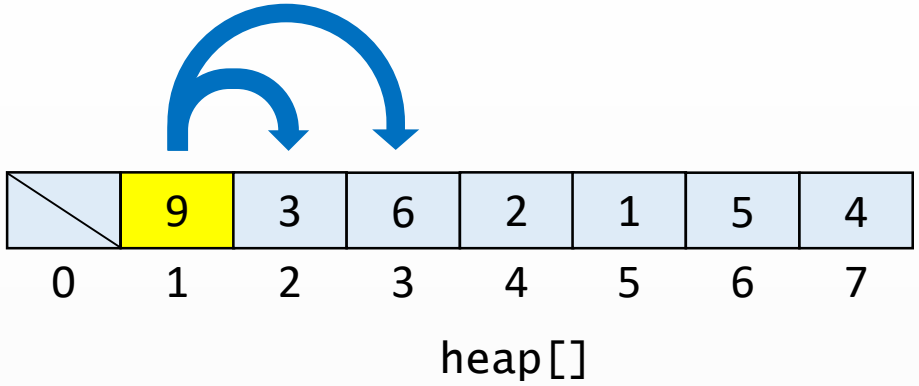
Max Heap

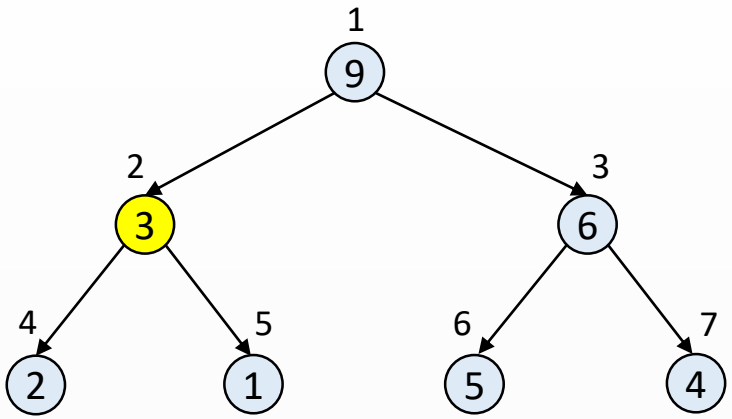
	9	3	6	2	1	5	4
0	1	2	3	4	5	6	7

heap[]

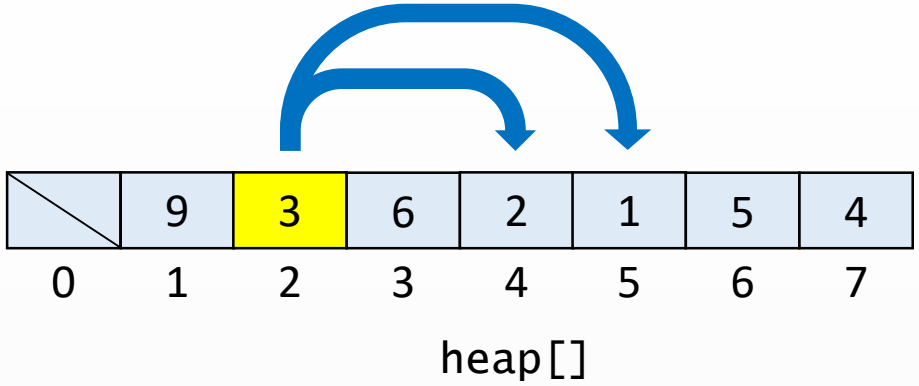


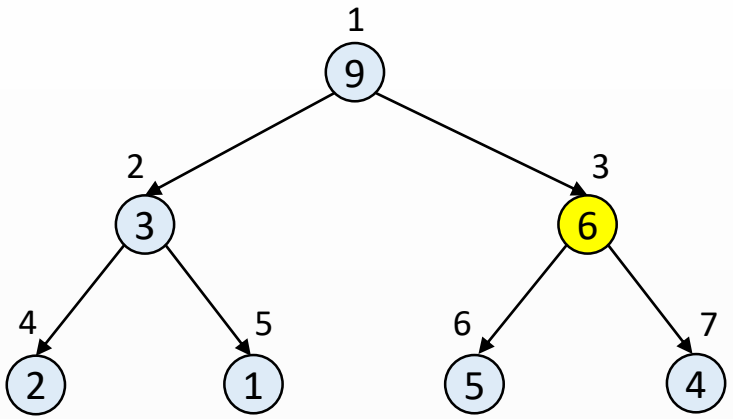
Max Heap



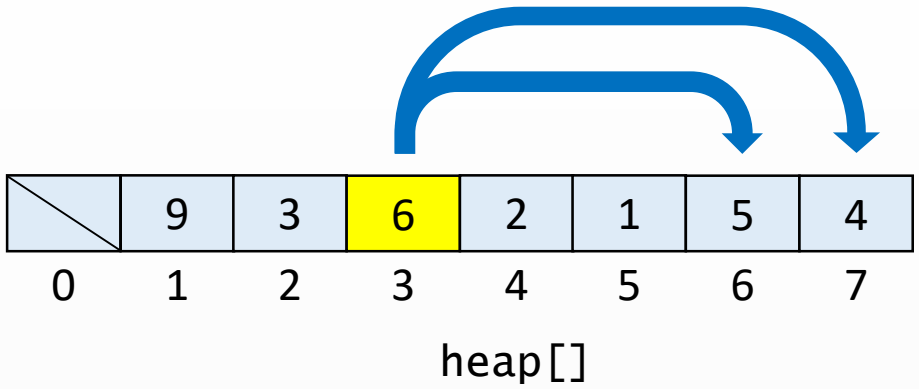


Max Heap





Max Heap



Çocuklar:

indeks 1 → 2, 3

indeks 2 → 4, 5

indeks 3 → 6, 7

indeks k → 2*k, 2*k + 1

Ebeveyn:

indeks 7 → ⌊ 7/2 ⌋ = 3

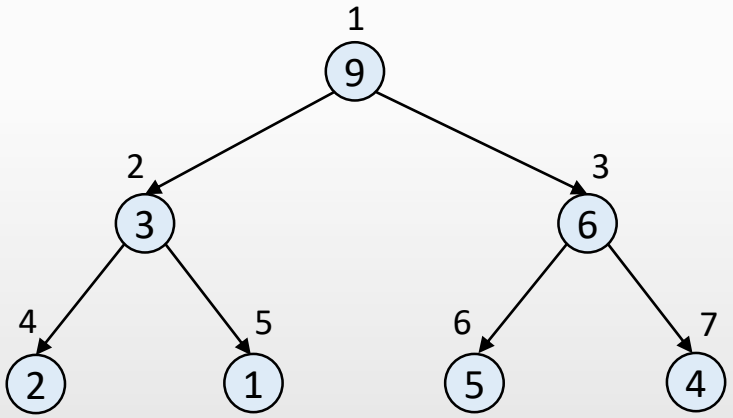
indeks 6 → ⌊ 6/2 ⌋ = 3

indeks 5 → ⌊ 5/2 ⌋ = 2

indeks k → ⌊ k/2 ⌋

Max İkili Heap Ağacı

- Her bir düğümün değerinin, çocuklarının değerlerinden büyük olduğu tam ikili ağaçtır.
- En büyük değer köktedir ve kökün indeksi 1'dir.



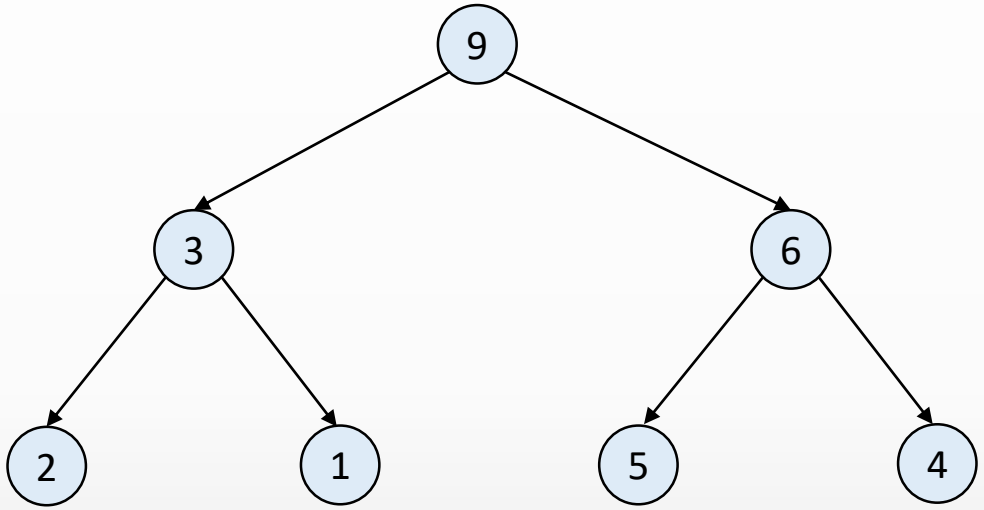
Max Heap

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

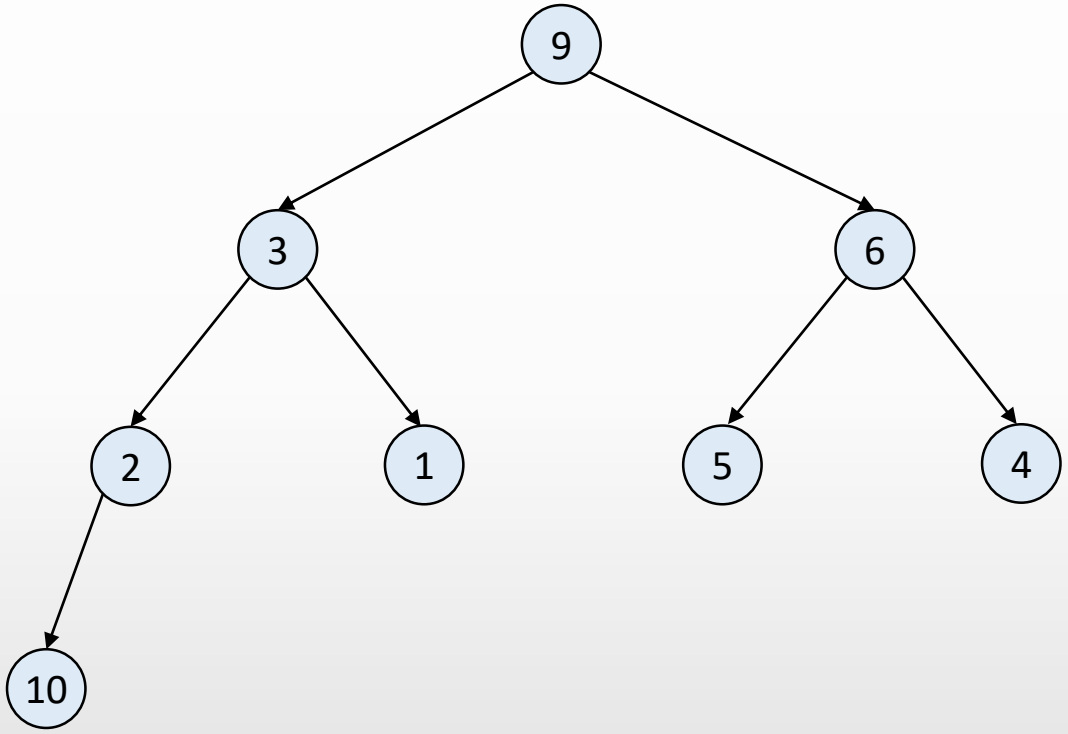
heap[]

Aşağıdan Yukarıya Heap Ağacına Dönüştürme

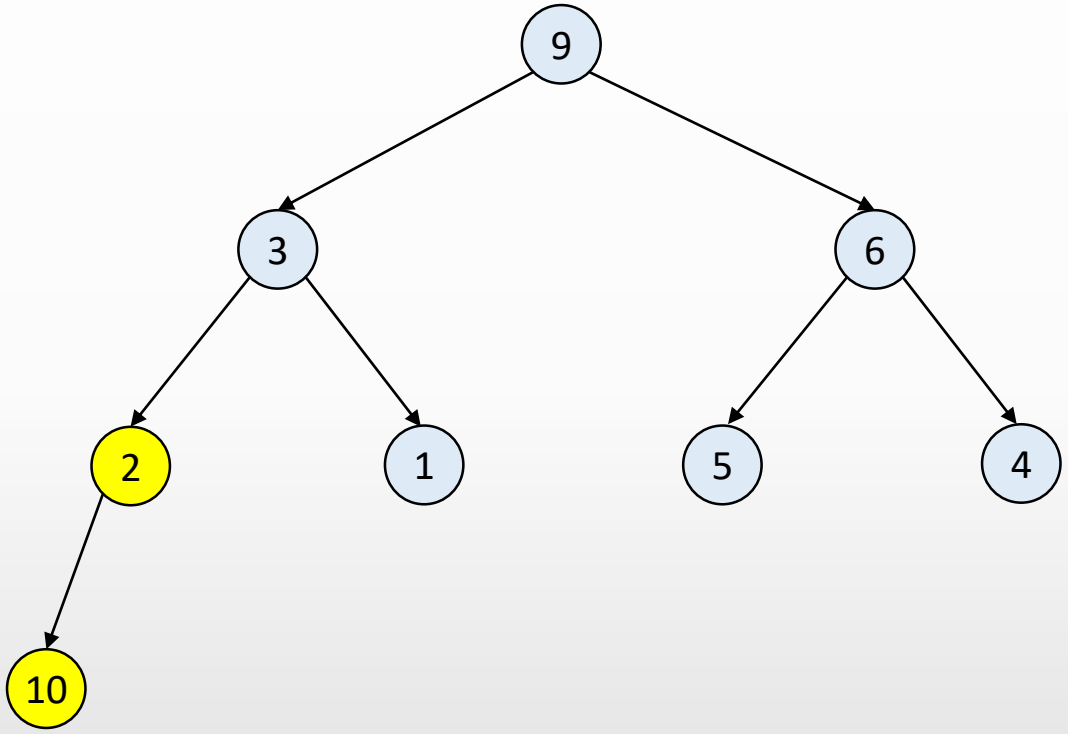
- Max heap ikili ağacının her bir düğümünün değeri, çocuklarının değerlerinden büyük olma özelliğini taşır.
- Ancak heap ağacına bir eleman eklendikten sonra bu özellik bozulabilir.
- Bu nedenle elemanların yerlerini aşağıdan yukarıya değiştirerek yeniden heap ağacına dönüştürme işlemi (yüzdür - swim) gerçekleştirilir (bottom-up heapify).



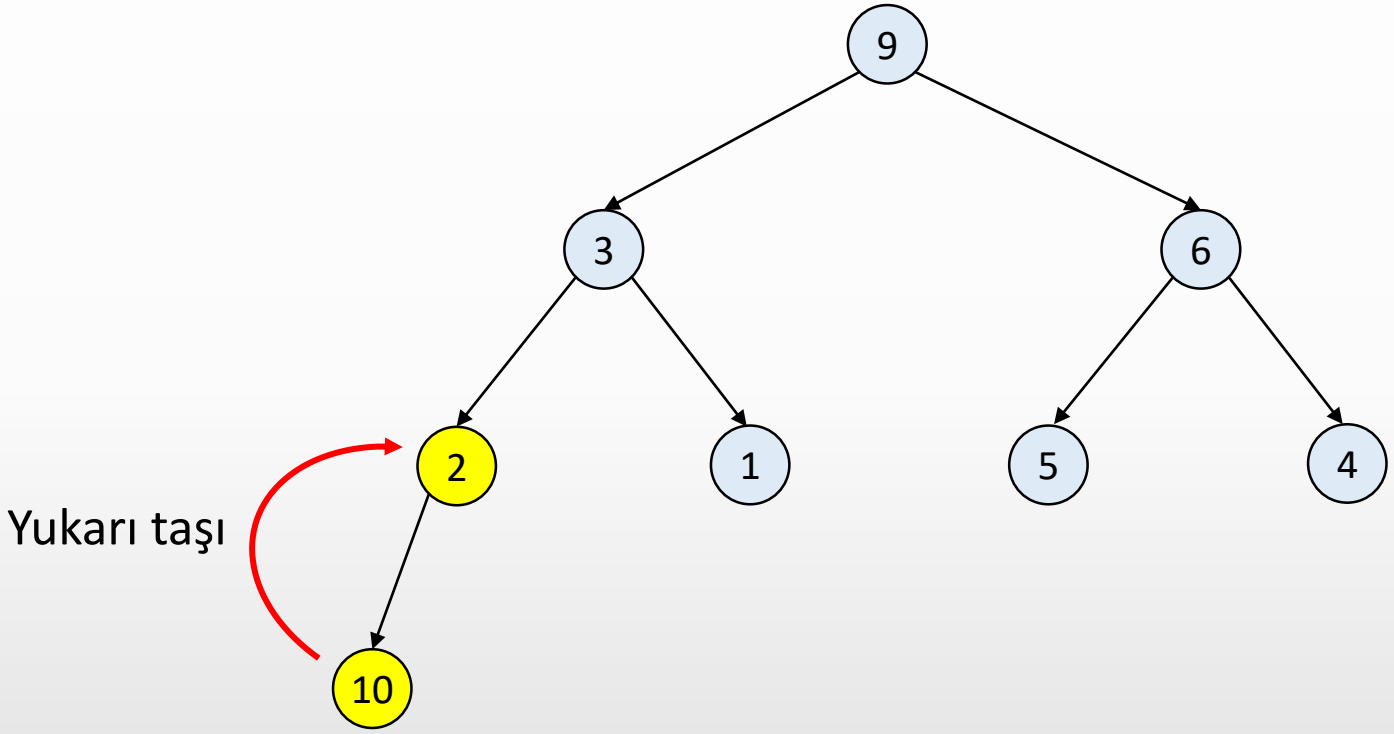

```
ekle(10)
```



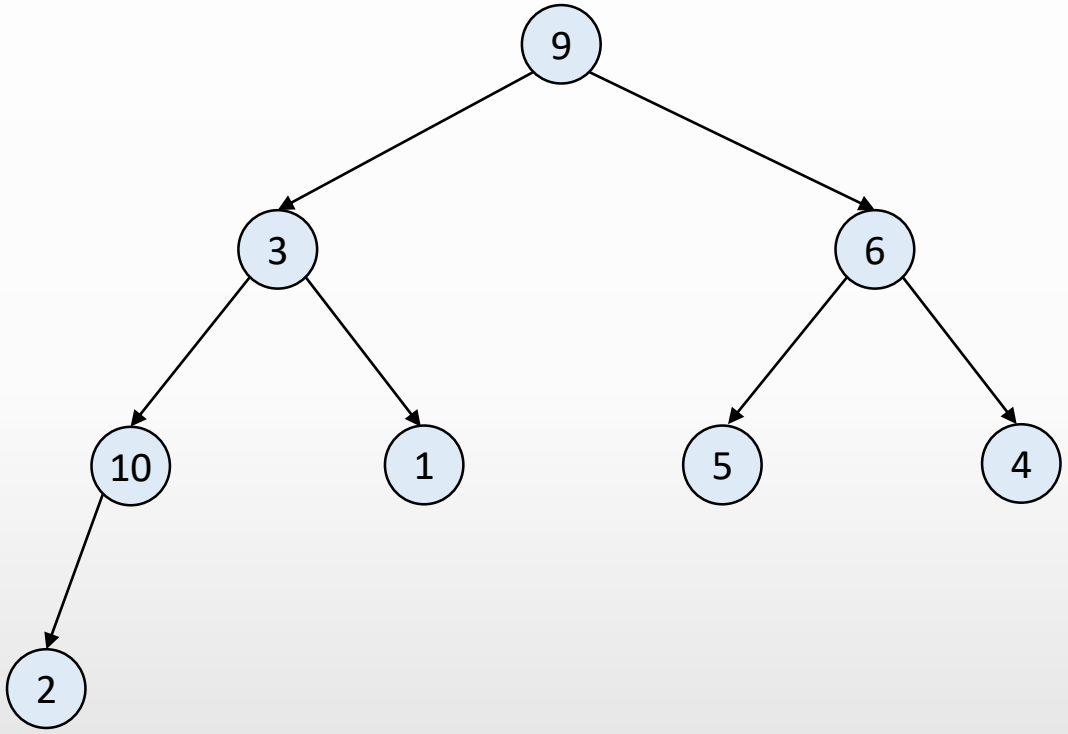

```
ekle(10)
```



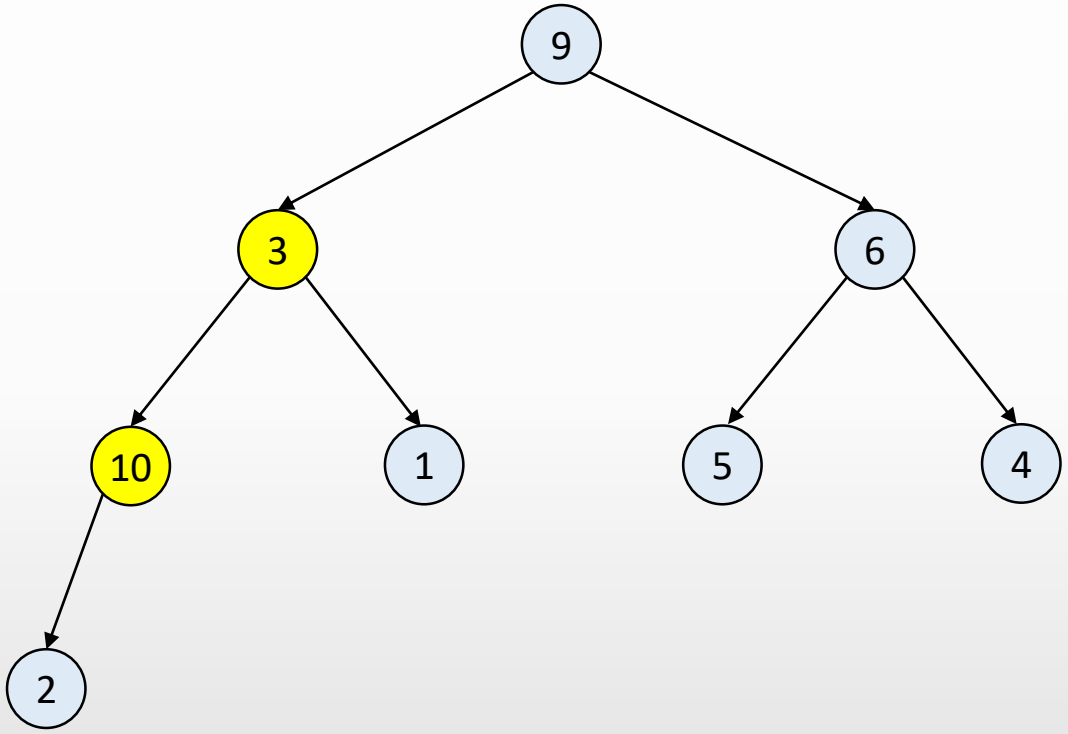

```
ekle(10)
```



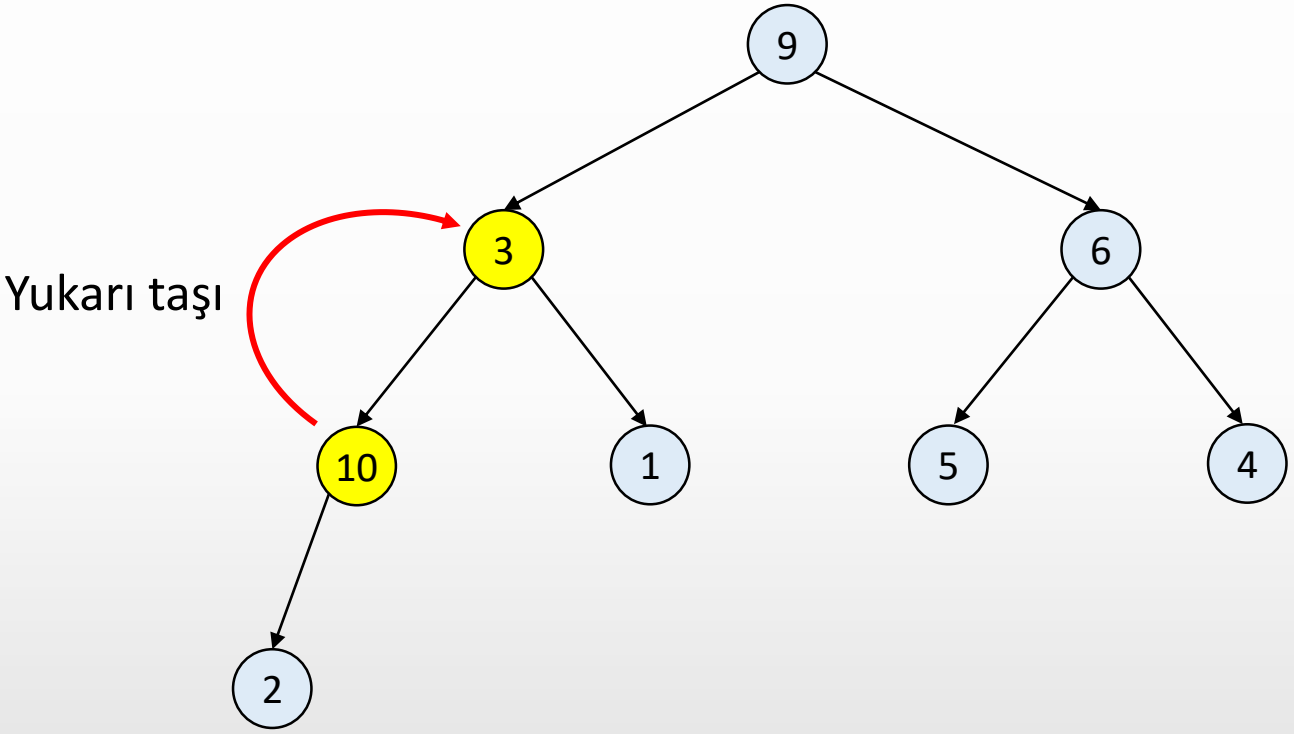

```
ekle(10)
```



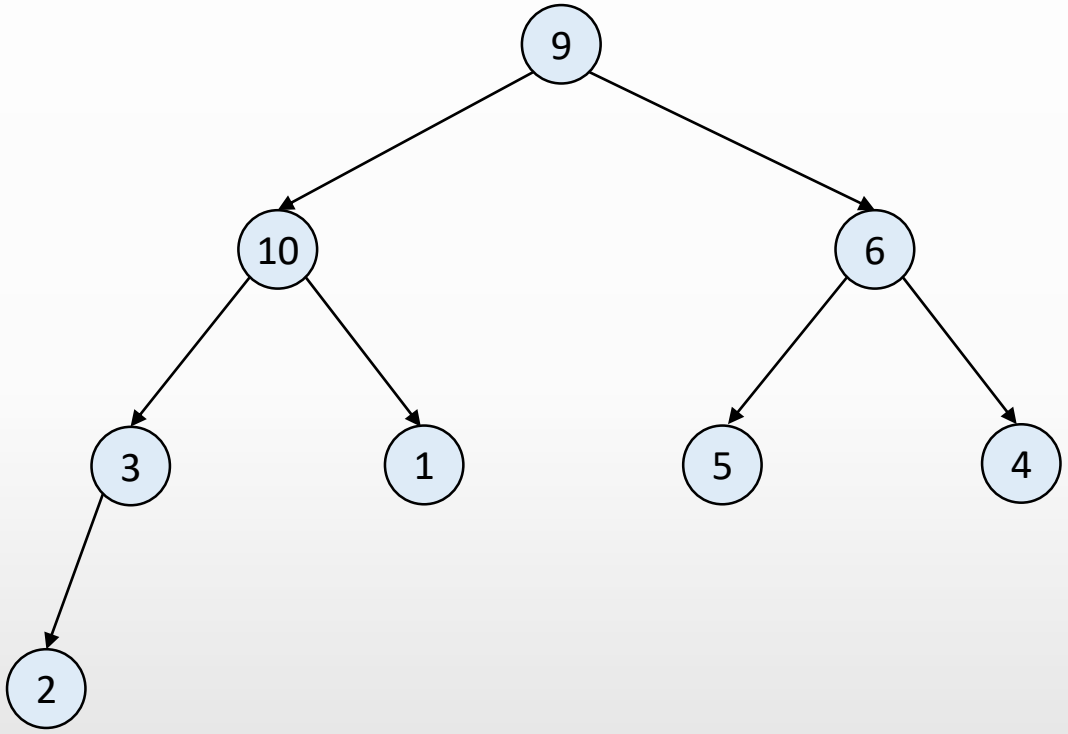

```
ekle(10)
```



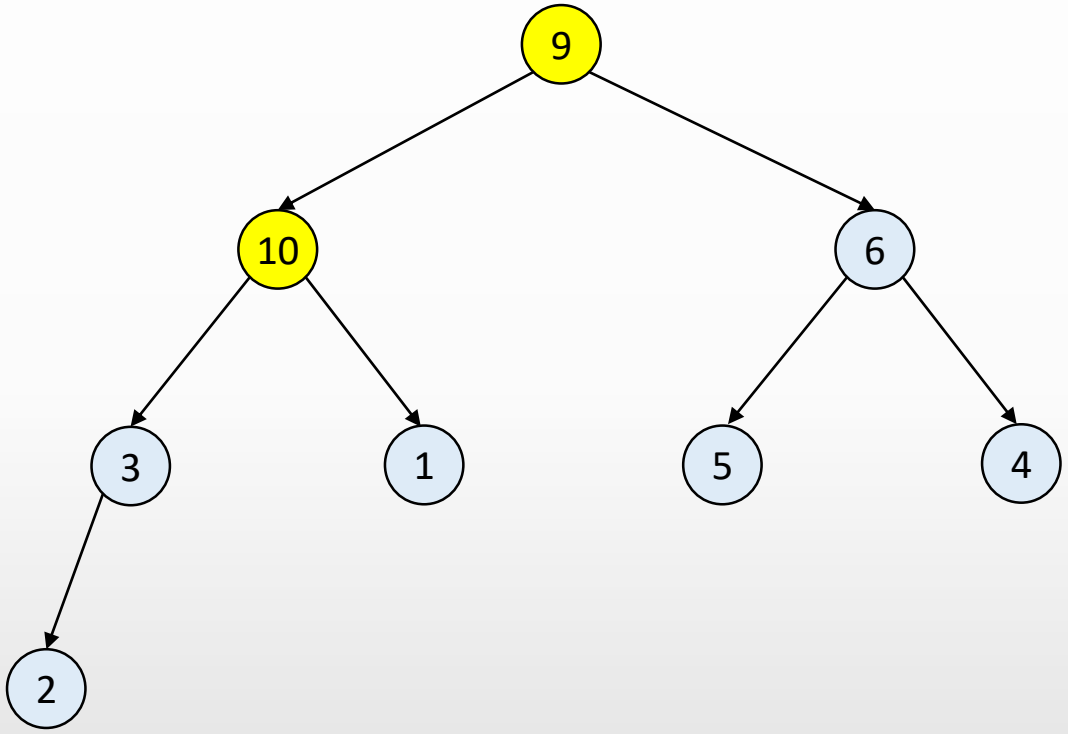

```
ekle(10)
```



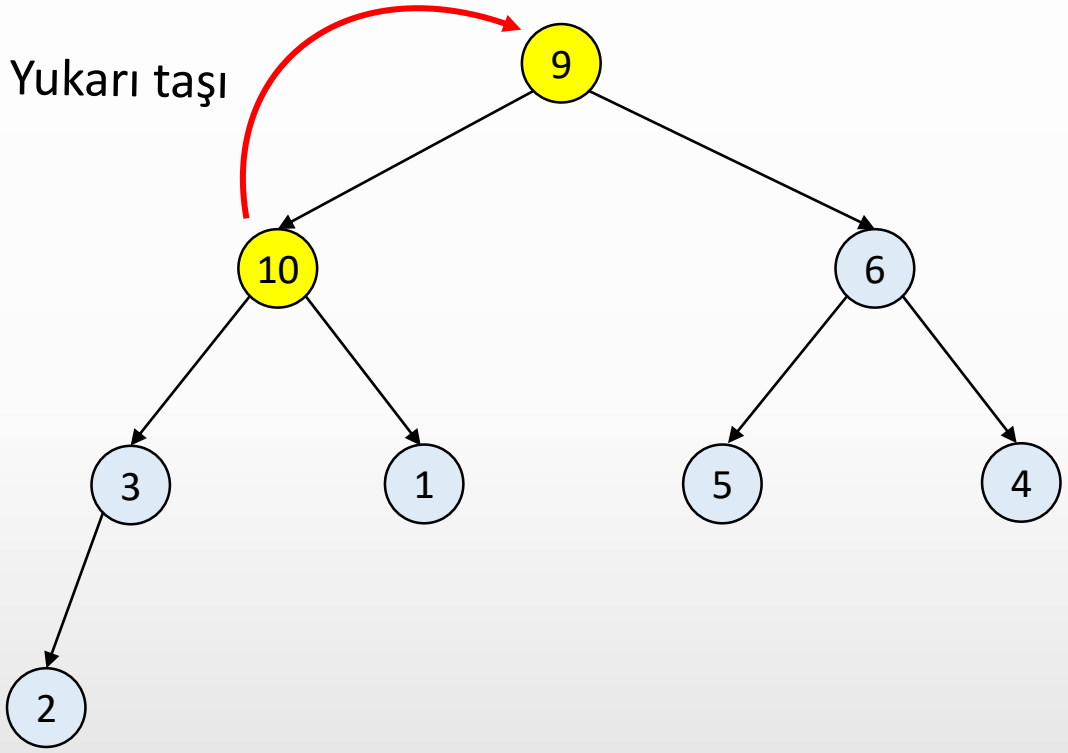

```
ekle(10)
```



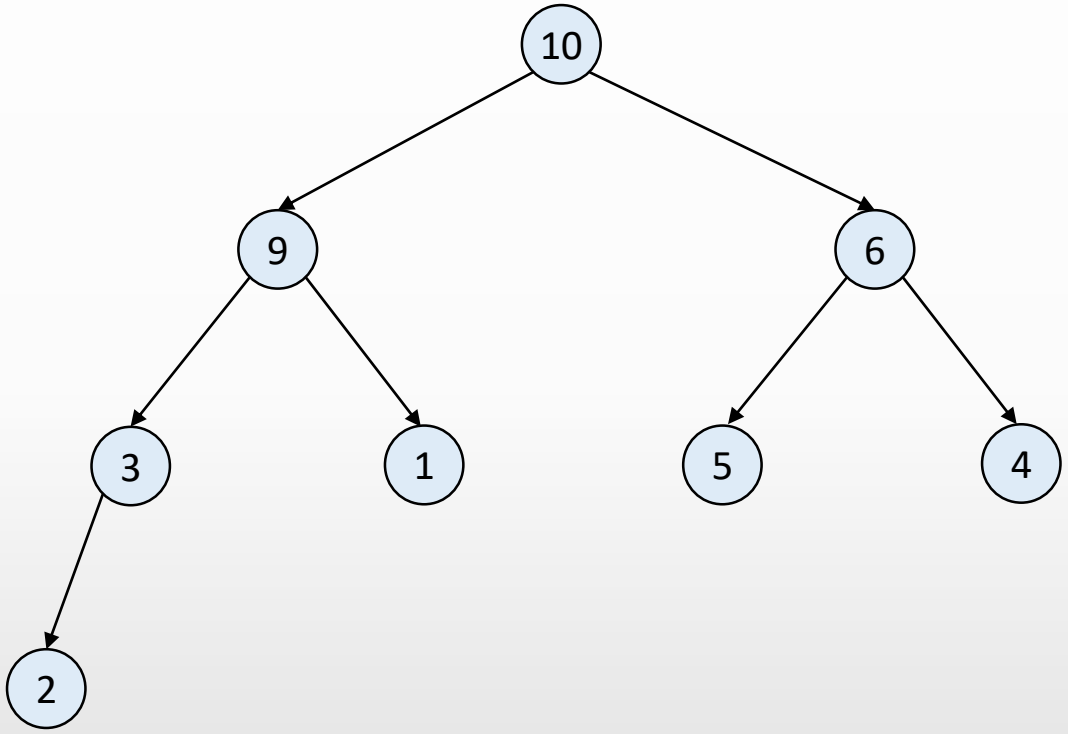

```
ekle(10)
```

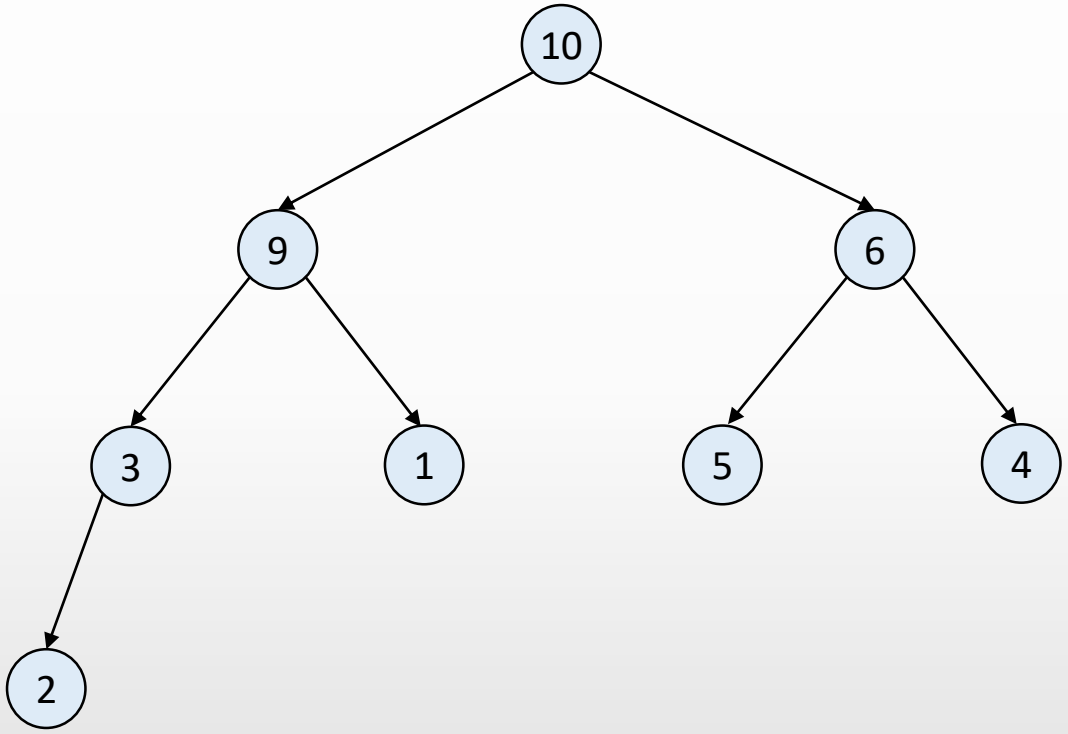



```
ekle(10)
```

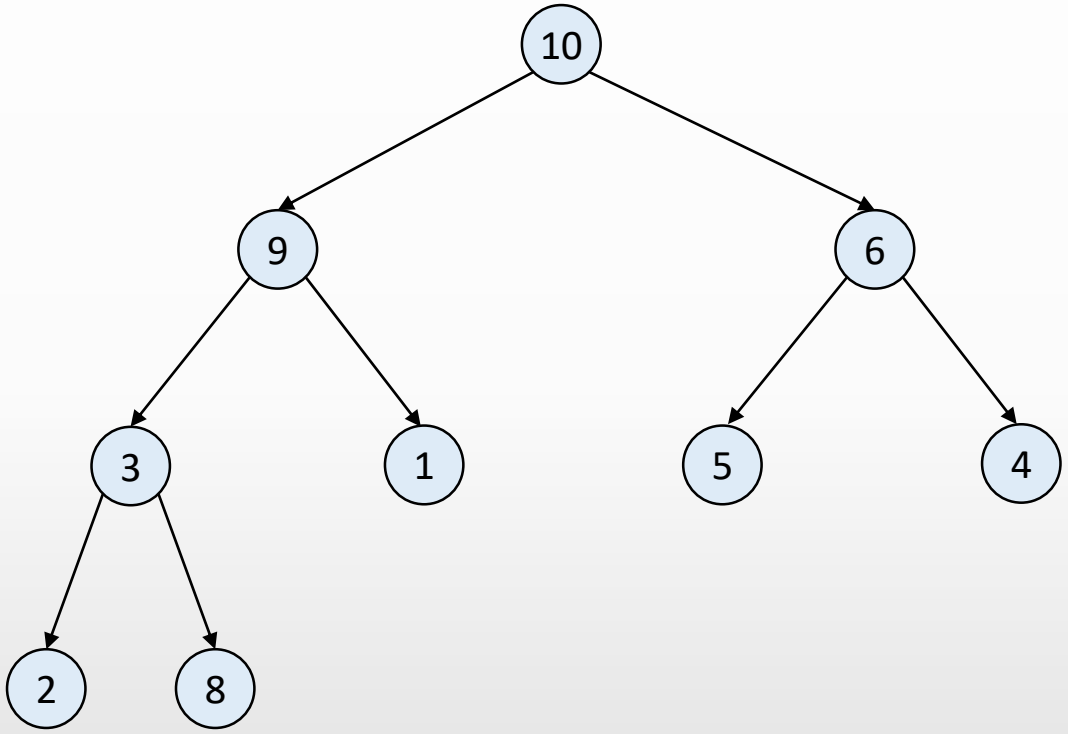



```
ekle(10)
```

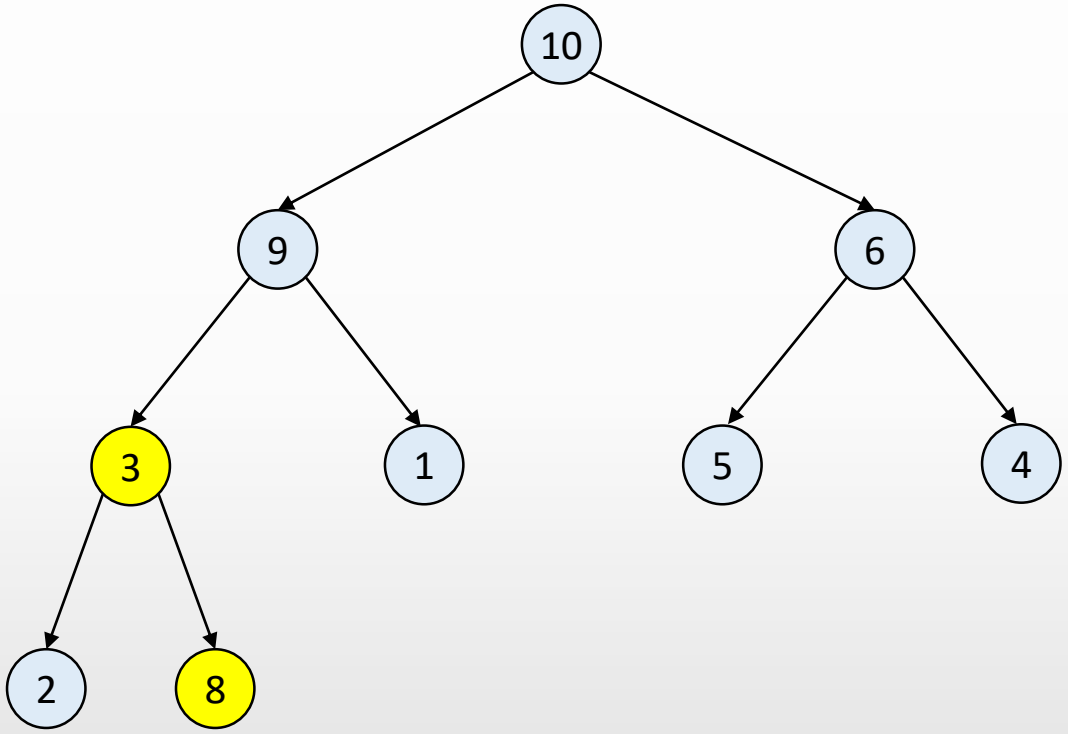

10
9
6
3
1
5
4
2



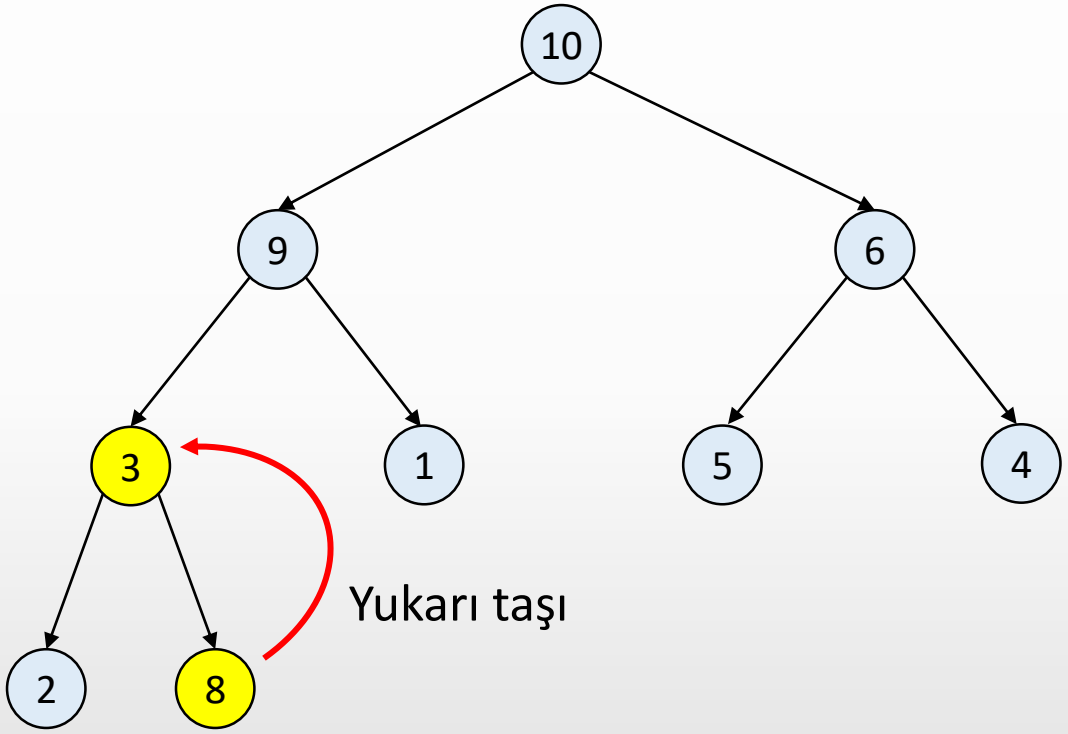

```
ekle(8)
```



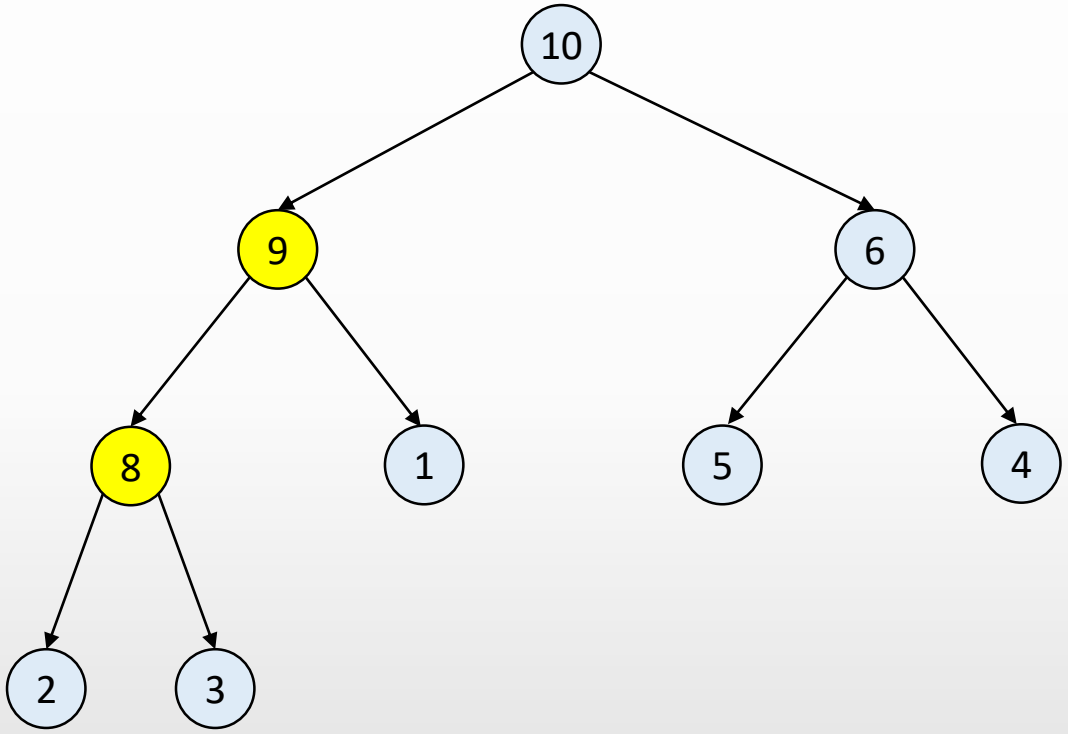

```
ekle(8)
```




```
ekle(8)
```




```
ekle(8)
```




```
ekle(8)
```

Aşağıdan Yukarıya Heap Ağacına Dönüştürme

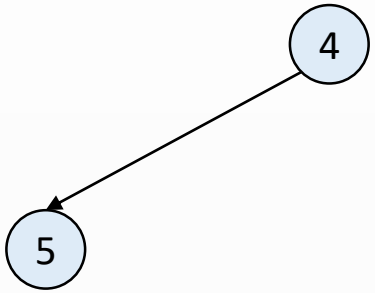
```
ekle(4)
```

4

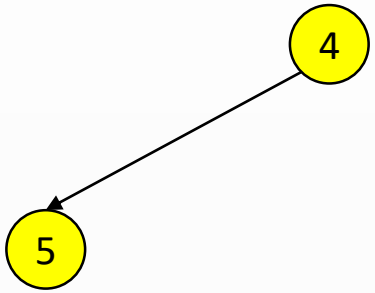
```
ekle(4)
```

4

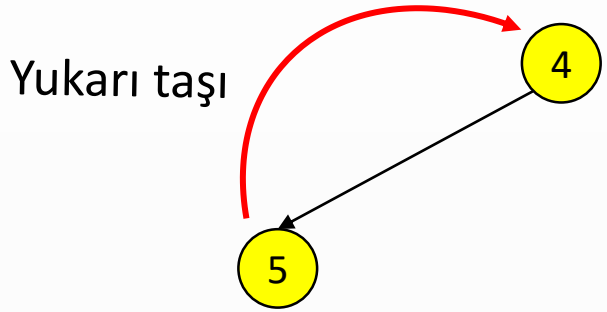
```
ekle(5)
```



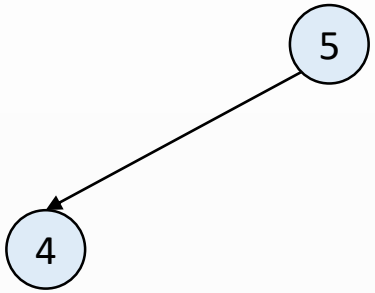

```
ekle(5)
```



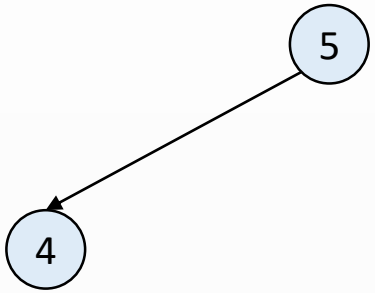

```
ekle(5)
```



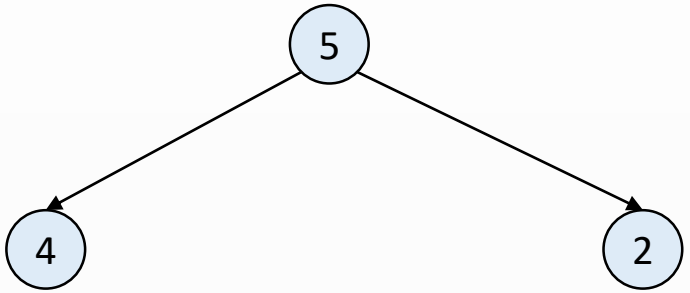

```
ekle(5)
```



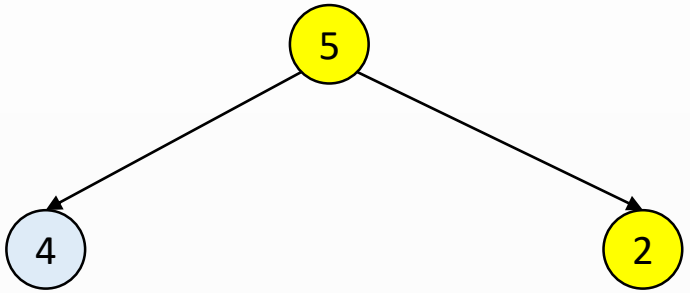

```
ekle(5)
```



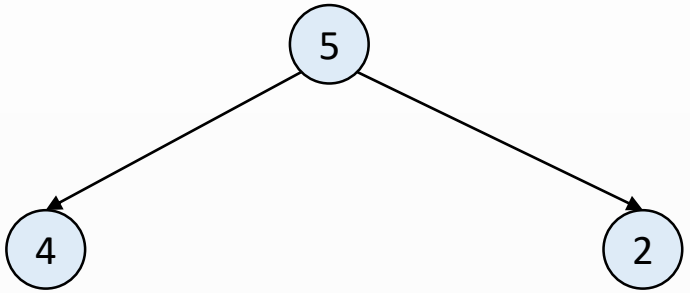

```
ekle(2)
```



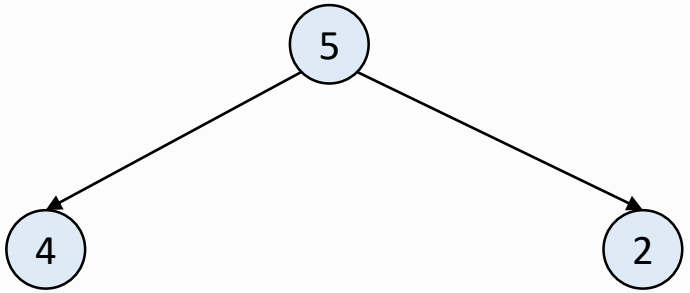

```
ekle(2)
```



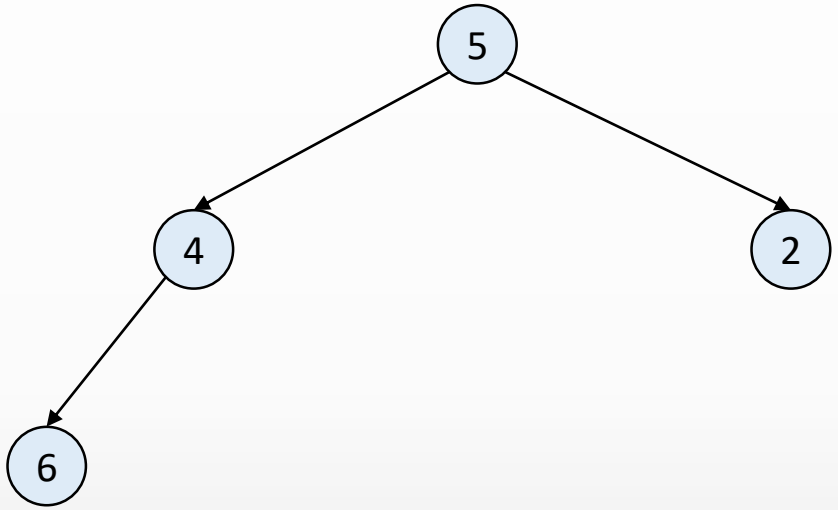

```
ekle(2)
```



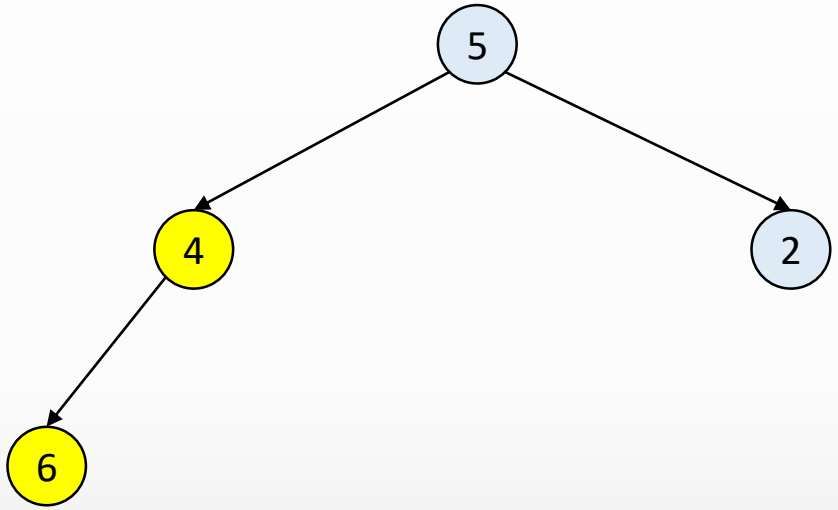

```
ekle(2)
```



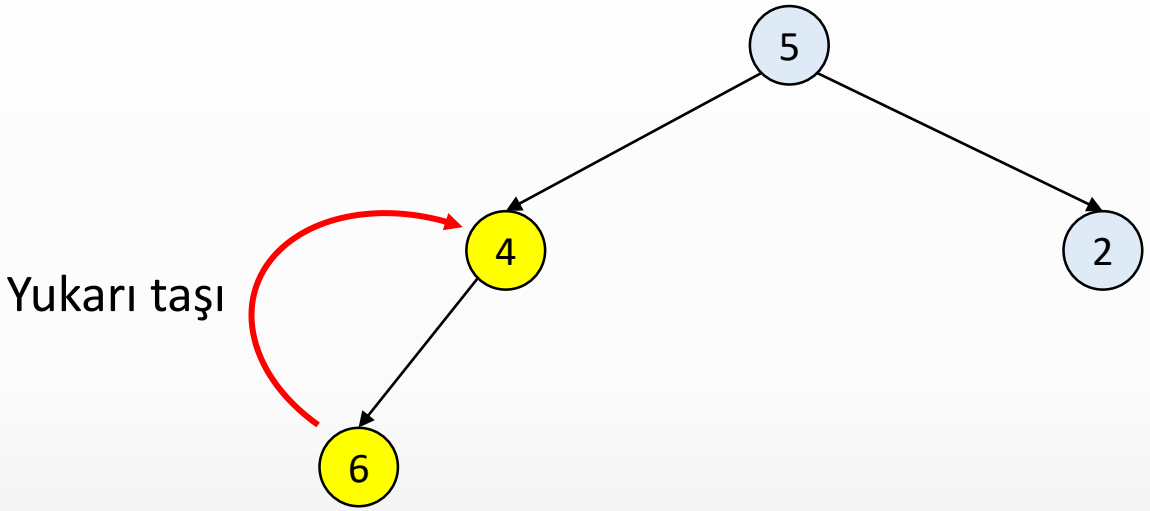

```
ekle(6)
```



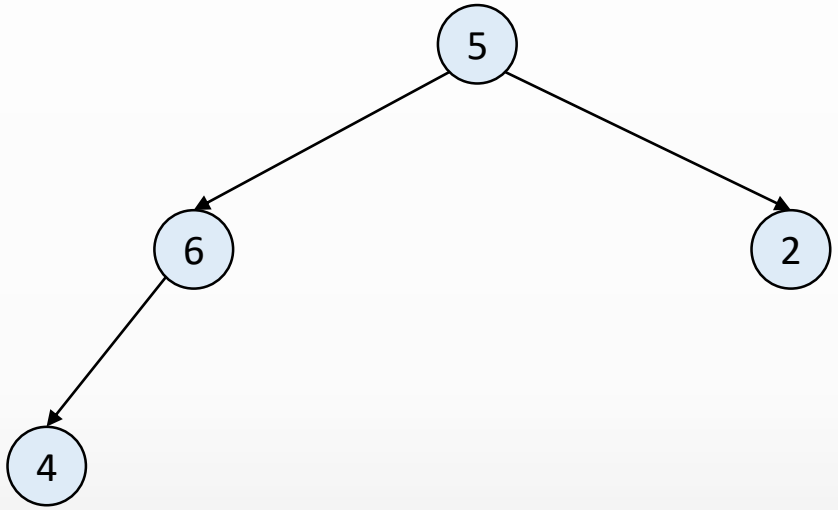

```
ekle(6)
```



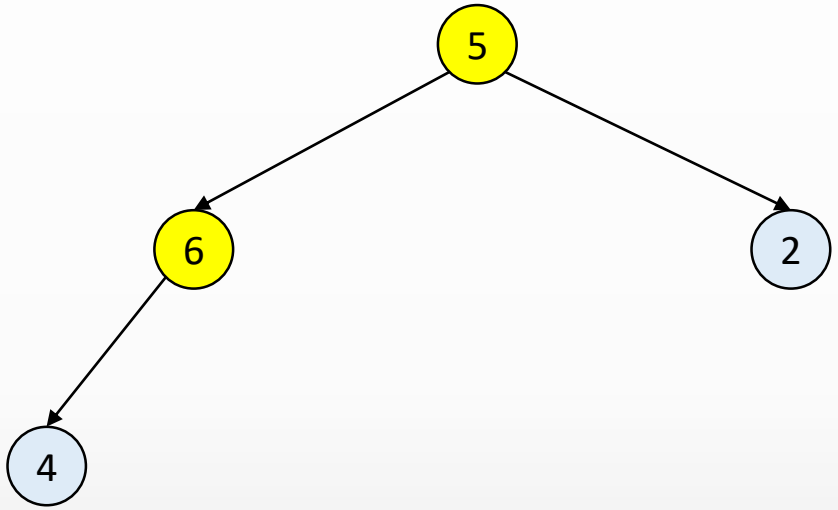

```
ekle(6)
```



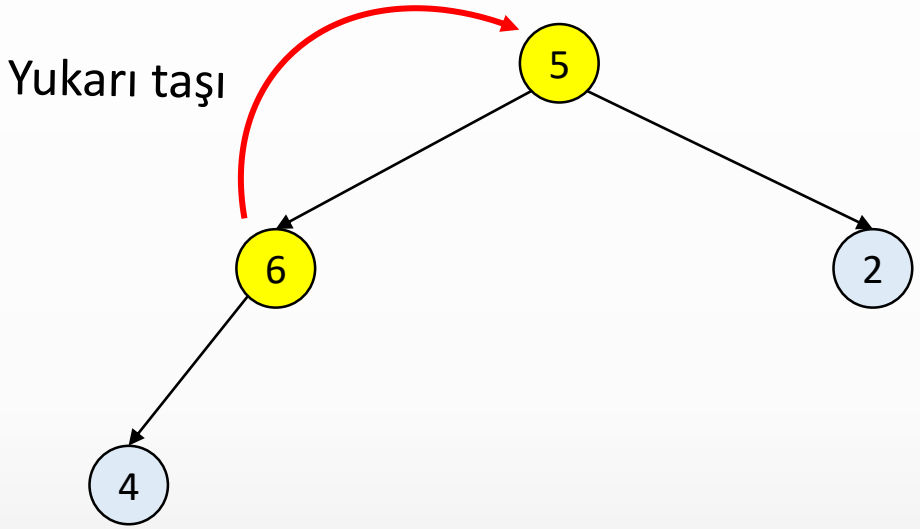

```
ekle(6)
```



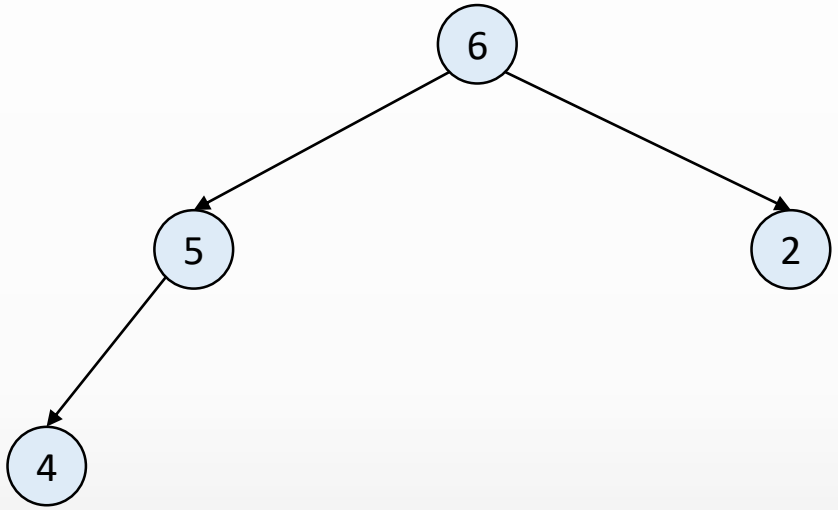

```
ekle(6)
```



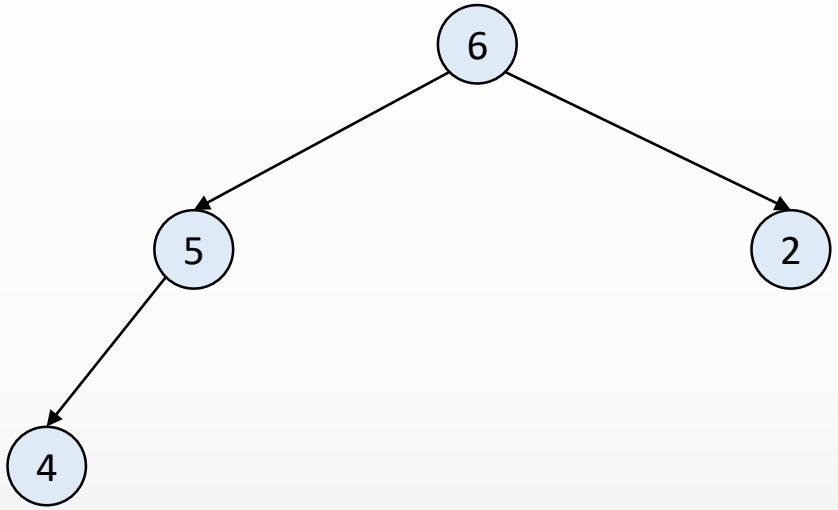

```
ekle(6)
```



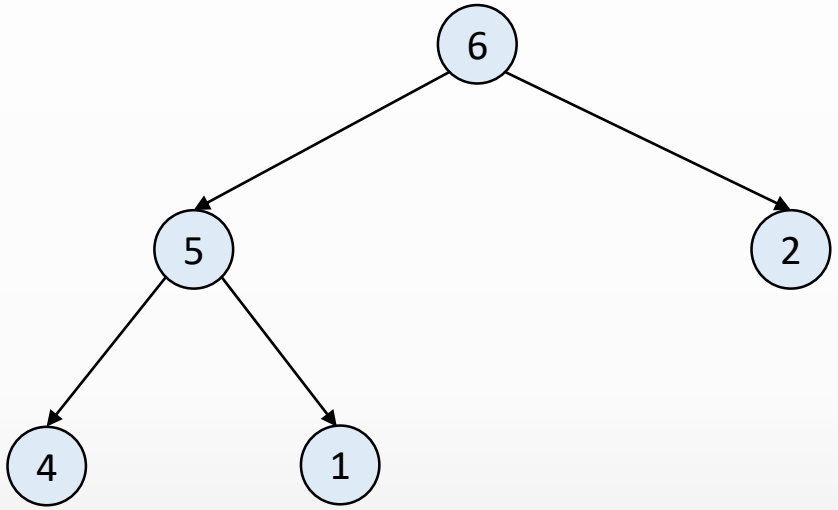

```
ekle(6)
```



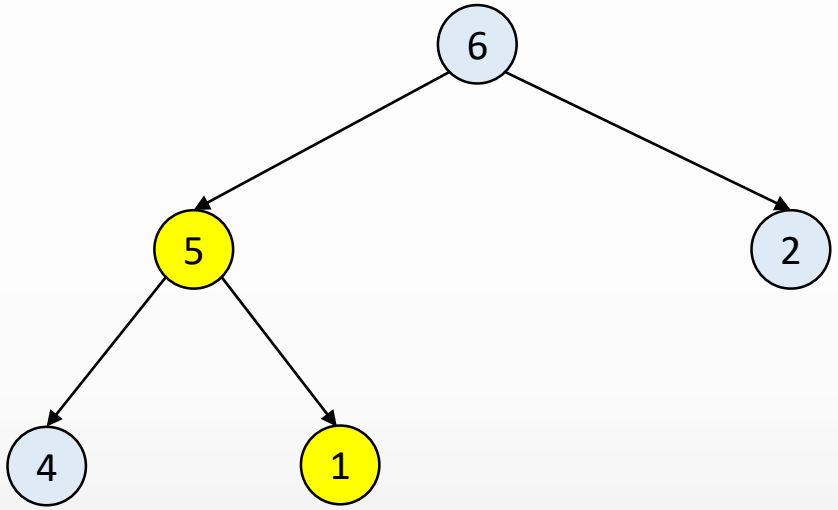

```
ekle(6)
```



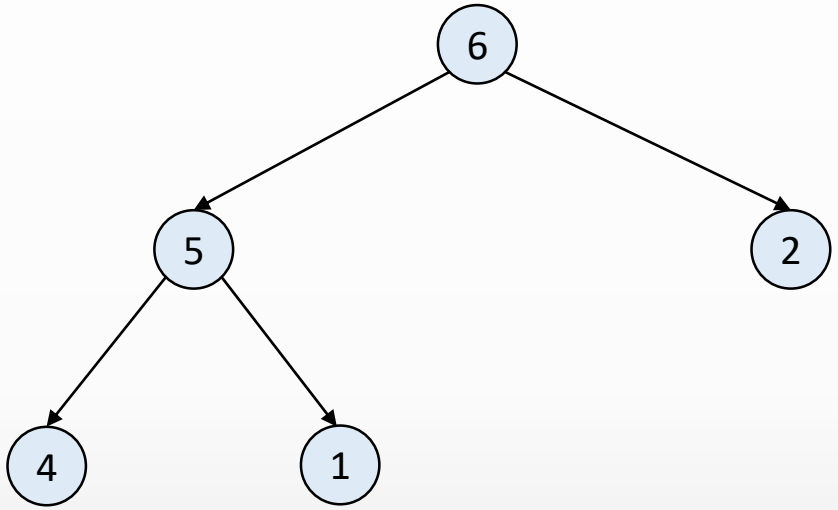

```
ekle(1)
```



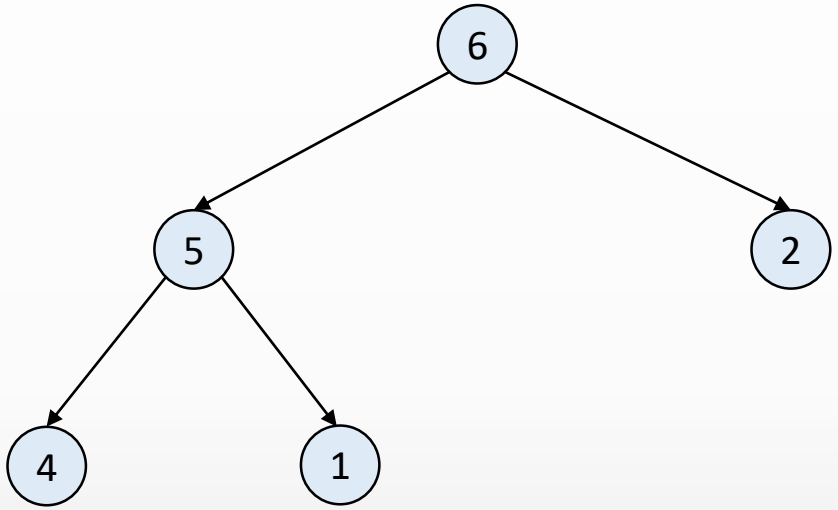

```
ekle(1)
```



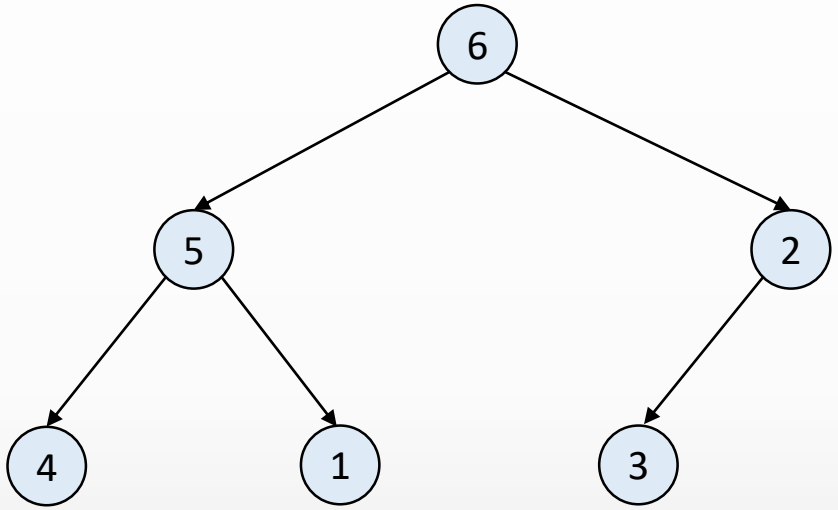

```
ekle(1)
```



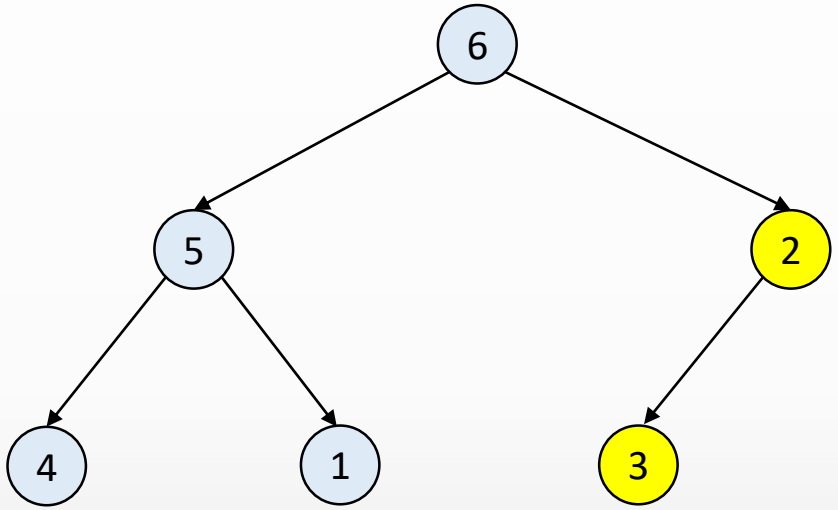

```
ekle(1)
```



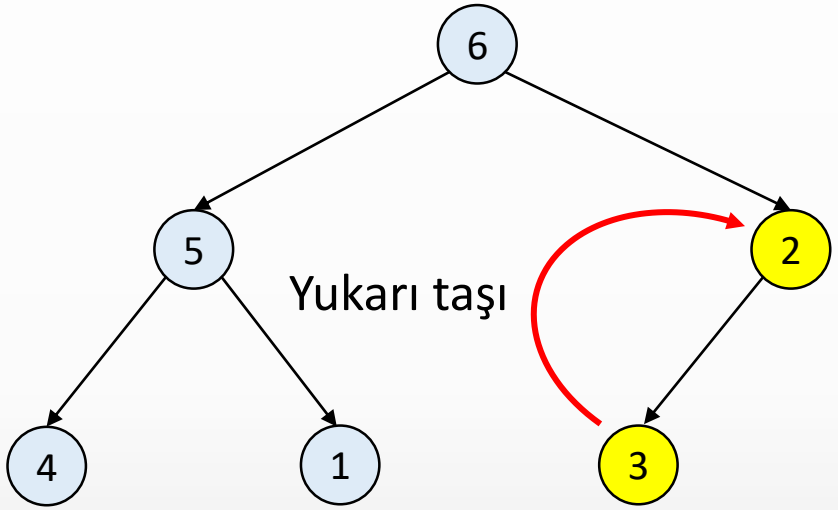

```
ekle(3)
```



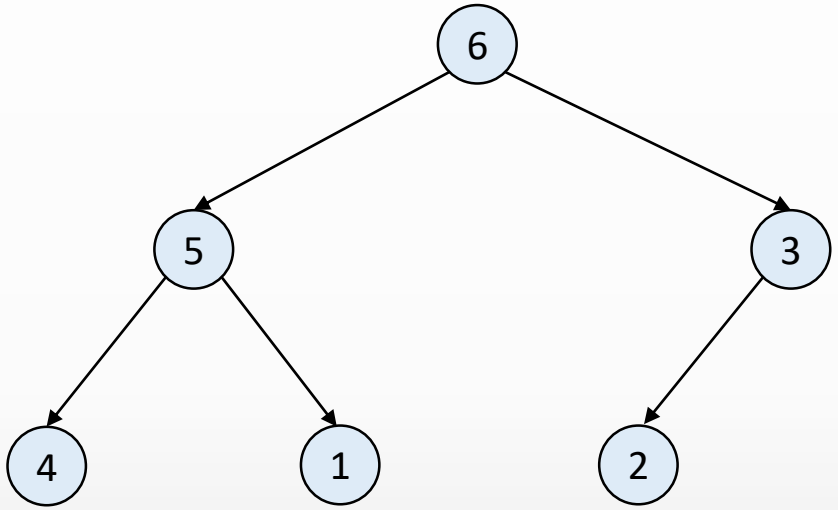

```
ekle(3)
```



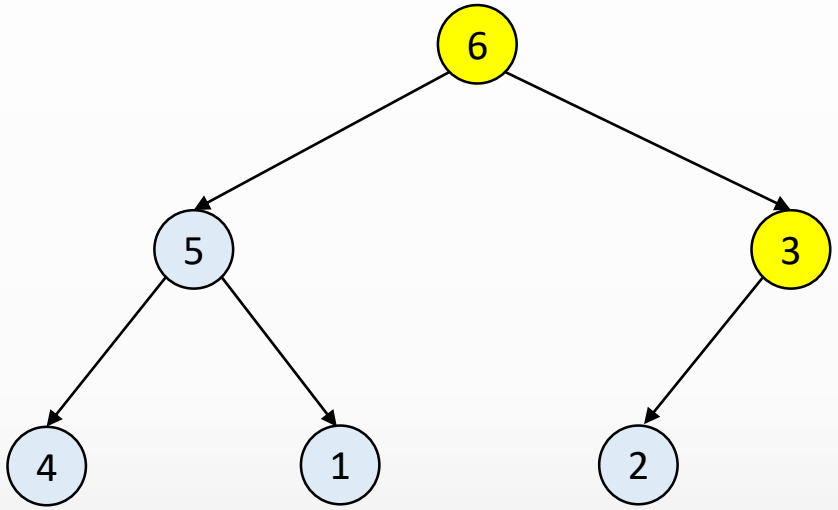

```
ekle(3)
```



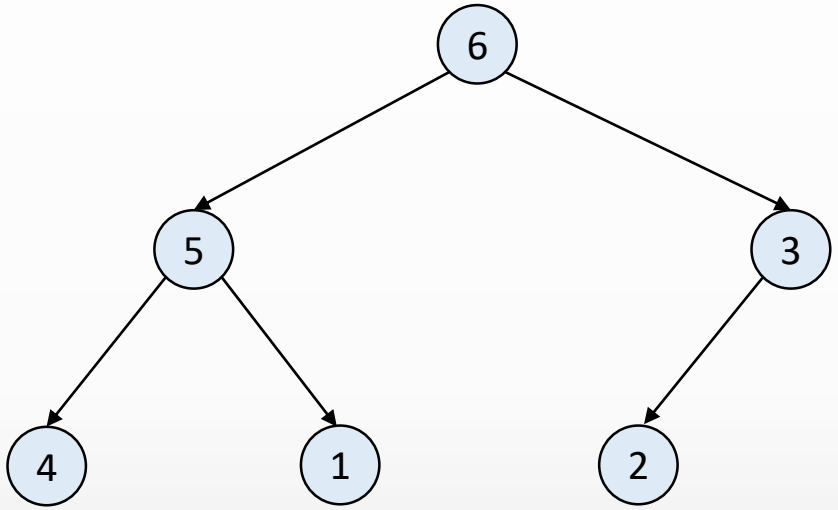

```
ekle(3)
```




```
ekle(3)
```




```
ekle(3)
```




```
ekle(3)
```

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;
  }
}
```

	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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

n = 0
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	null	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)
```

```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[]

```
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[]

```
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[]

```
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[]

```
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;
  }
}
```

	4	null	null
0	1	2	3

heap[]

```
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;
  }
}
```

	4	null	null
0	1	2	3

heap[]

n = 1
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;
  }
}
```

	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 = 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;
  }
}
```

	4	5	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;
  }
}
```

	4	5	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;
  }
}
```

	4	5	null
0	1	2	3

heap[]

```
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;
  }
}
```

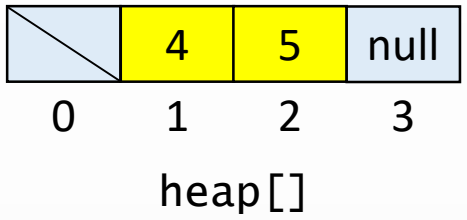
	4	5	null
0	1	2	3

heap[]

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

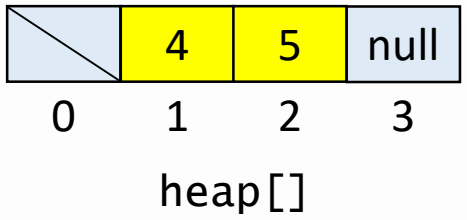
```
ekle(5)
```

```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;
  }
}
```



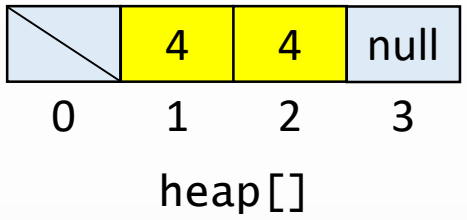

```
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


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;
  }
}
```



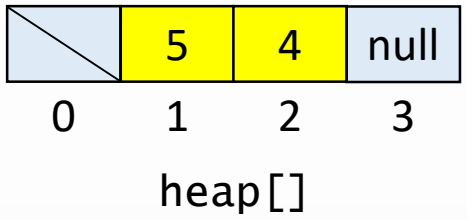

```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


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;
  }
}
```



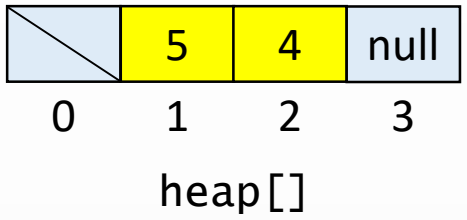

```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


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;
  }
}
```



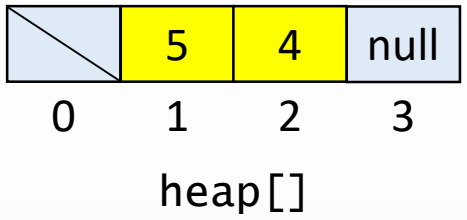

```
gecici = 5
k/2 = 1
k = 2
x = 5
n = 2
heap.length = 4


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;
  }
}
```

	5	4	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;
  }
}
```

	5	4	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;
  }
}
```

	5	4	null
0	1	2	3

heap[]

```
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;
  }
}
```

	5	4	null
0	1	2	3

heap[]

```
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 = 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 = 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;
  }
}
```

	5	4	2
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;
  }
}
```

	5	4	2
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;
  }
}
```

	5	4	2
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;
  }
}
```

	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;
  }
}
```

	5	4	2
0	1	2	3

heap[]

```
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;
  }
}
```

	5	4	2
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;
  }
}
```

	5	4	2
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;
  }
}
```

	5	4	2
0	1	2	3

heap[]

n = 3
heap.length = 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;
  }
}
```

	5	4	2
0	1	2	3

heap[]

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;
  }
}
```

	5	4	2
0	1	2	3

heap[]

```
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
0	1	2	3

heap[]

```
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
0	1	2	3

heap[]

```
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)
```

```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[]

```
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;
  }
}
```

	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)
```

```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[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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)
```

```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 = 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;
  }
}
```

	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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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)
```

```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)
```

```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;
  }
}
```

	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[]

```
n = 4
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	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)
```

```
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)
```

```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/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[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

	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;
  }
}
```

	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;
  }
}
```

	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[]

```
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;
  }
}
```

	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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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;
  }
}
```

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

heap[]

```
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)
```

```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/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[]

```
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[]

```
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;
  }
}
```

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

heap[]

n = 6
heap.length = 8

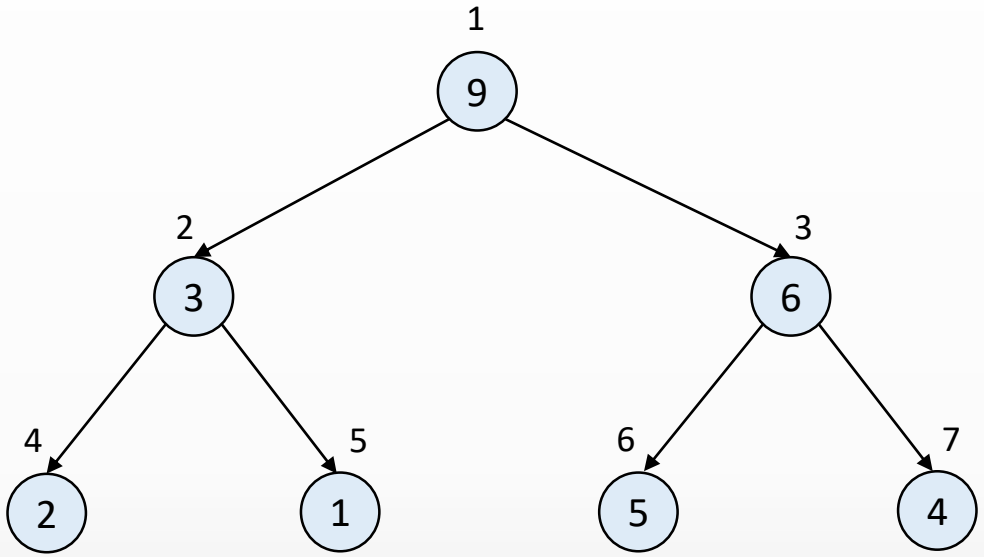
```
public void ekle(int x) {
  if (n == heap.length - 1) {
    buyut(2 * heap.length);
  }
  n++;
  heap[n] = x;
  yuzdur(n);
}

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

Yukarıdan Aşağıya Heap Ağacına Dönüştürme

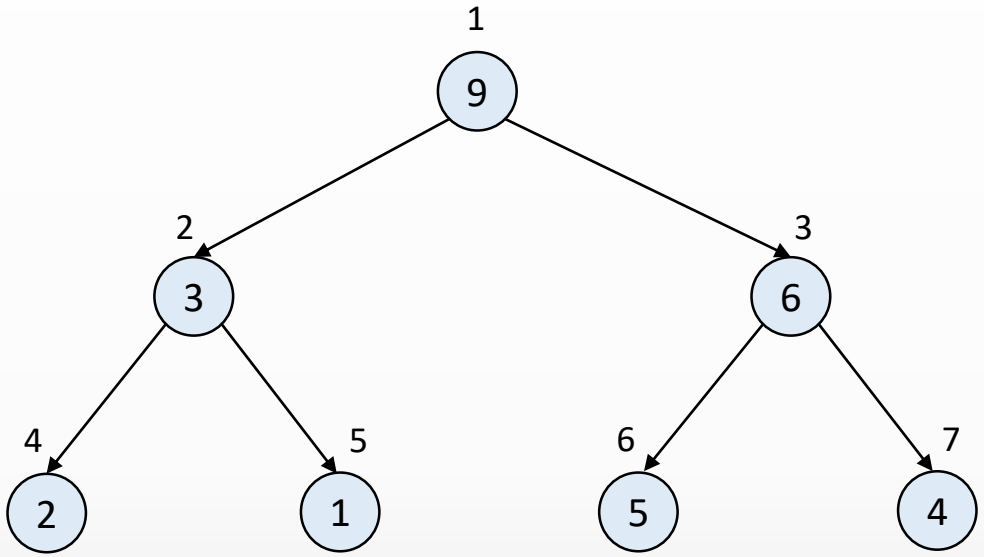
- Max heap ikili ağacının her bir düğümünün değeri, çocuklarının değerlerinden büyük olma özelliğini taşır.
- Ancak heap ağacında bir eleman silindikten sonra bu özellik bozulabilir.
- Bu nedenle elemanların yerlerini yukarıdan aşağıya değiştirerek yeniden heap ağacına dönüştürme işlemi (batır - sink) gerçekleştirilir (top-down heapify).

silMax()



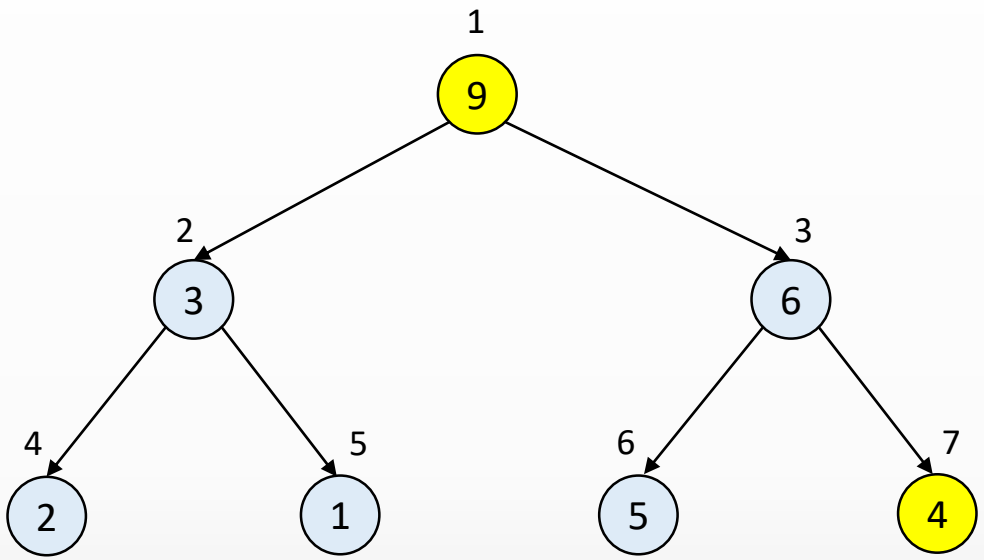
silMax()

max = 9



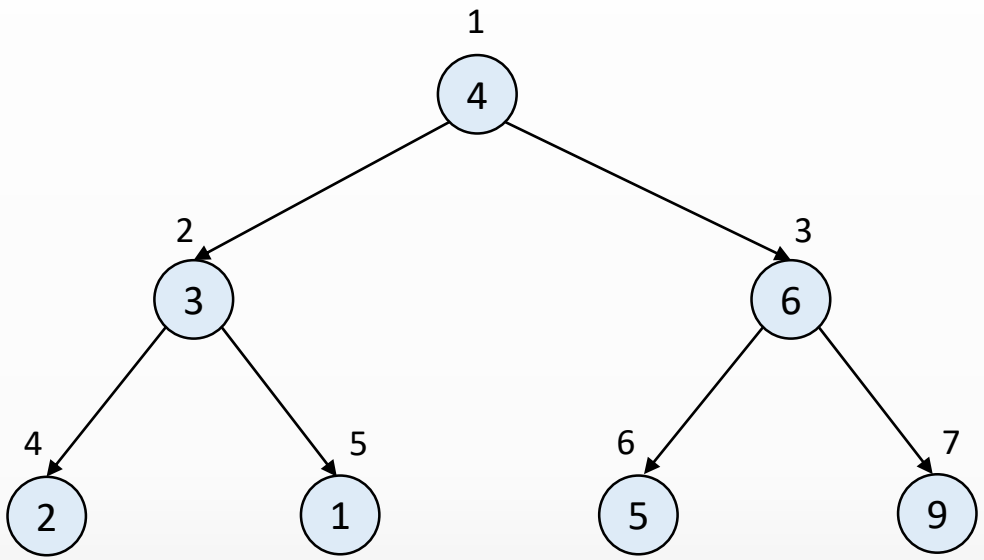
silMax()

max = 9



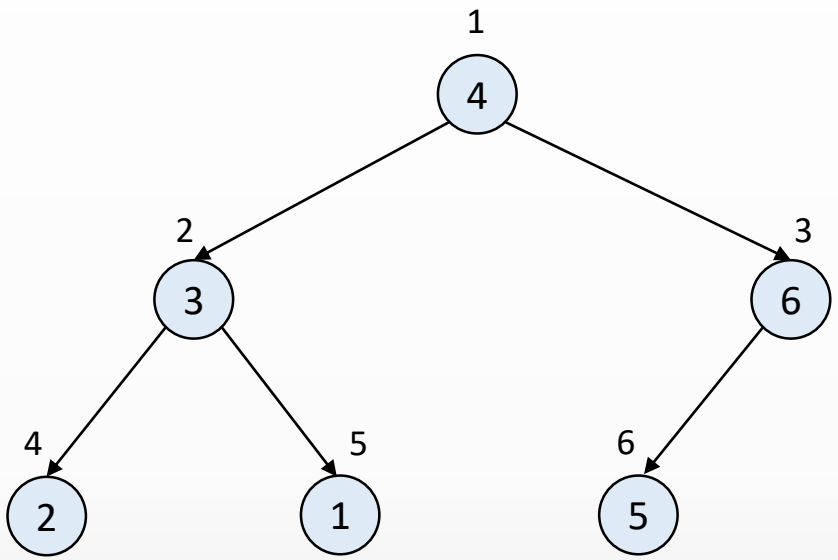
silMax()

max = 9



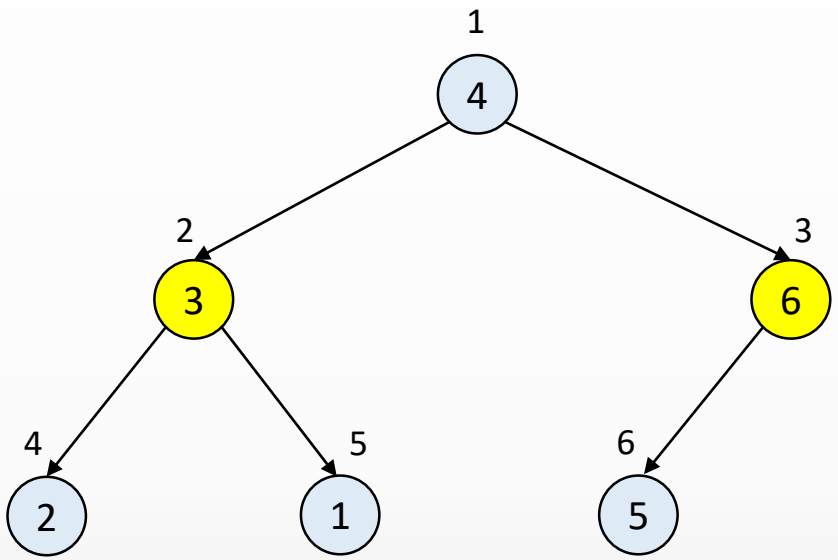
silMax()

max = 9



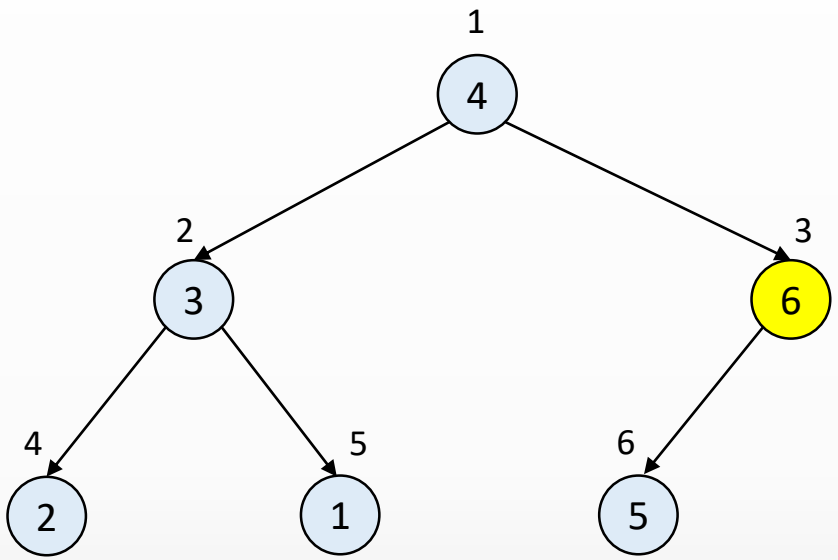
silMax()

max = 9



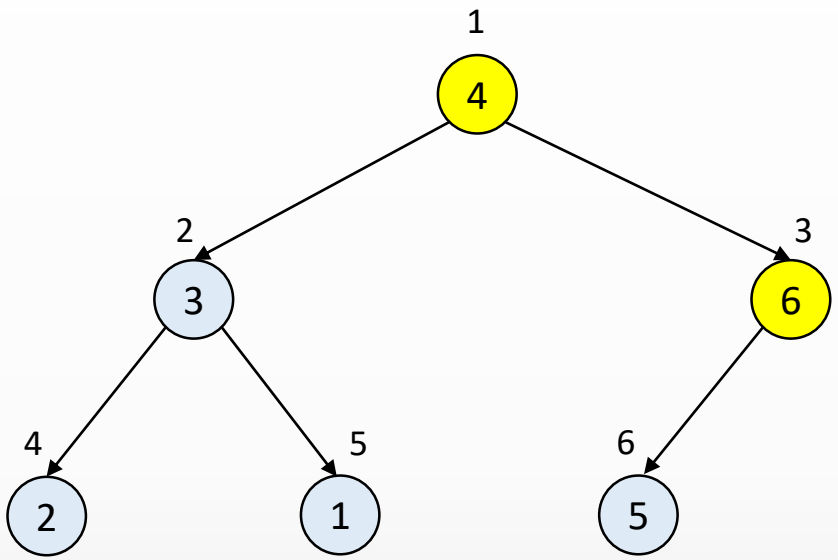
silMax()

max = 9



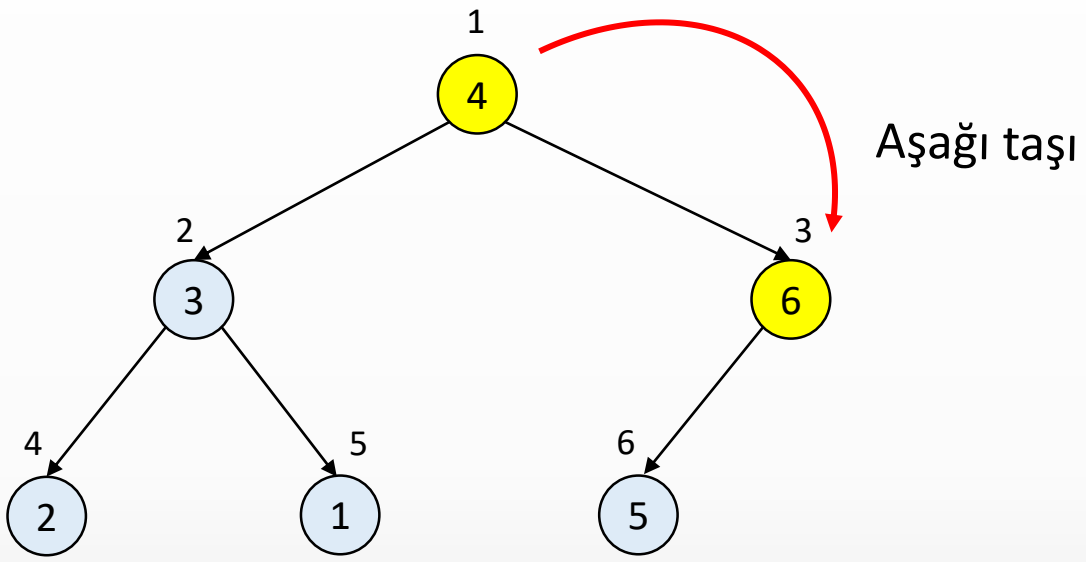
silMax()

max = 9



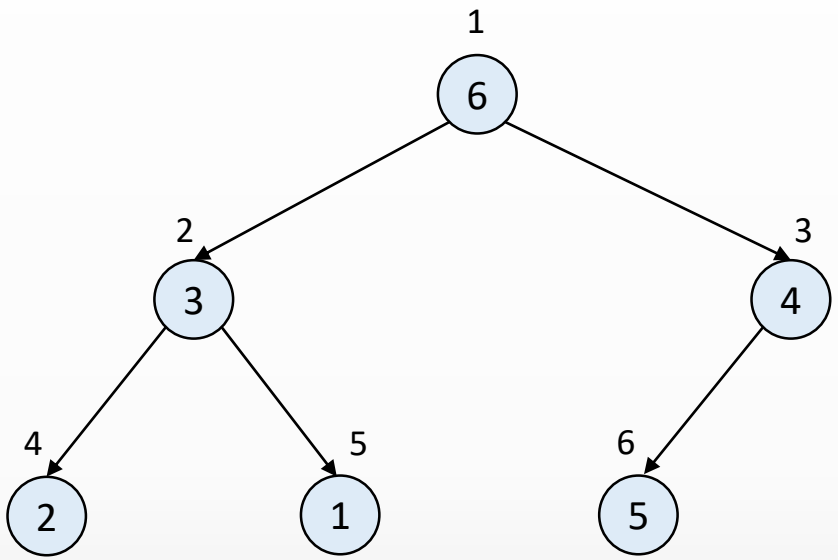
silMax()

max = 9



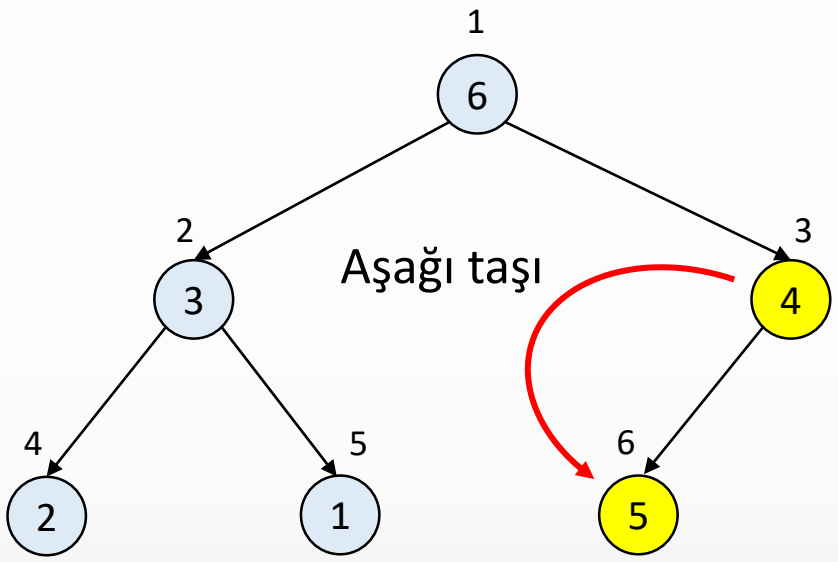
silMax()

max = 9



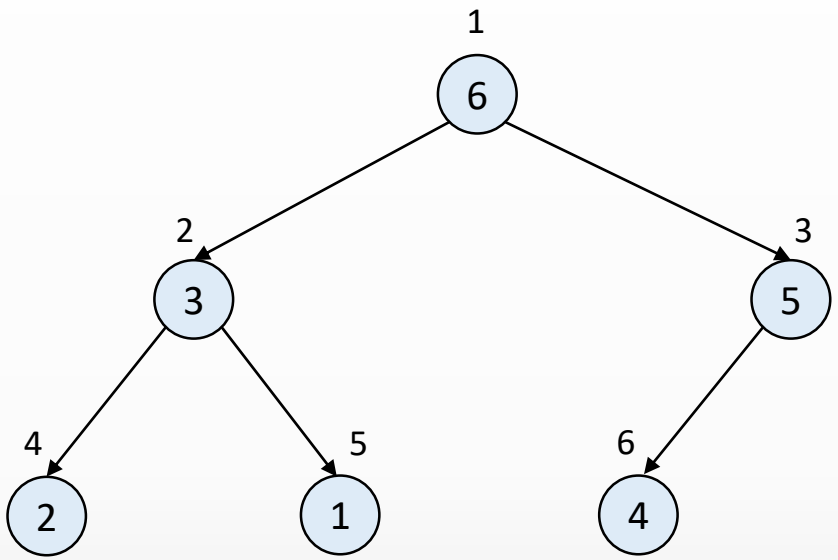
silMax()

max = 9

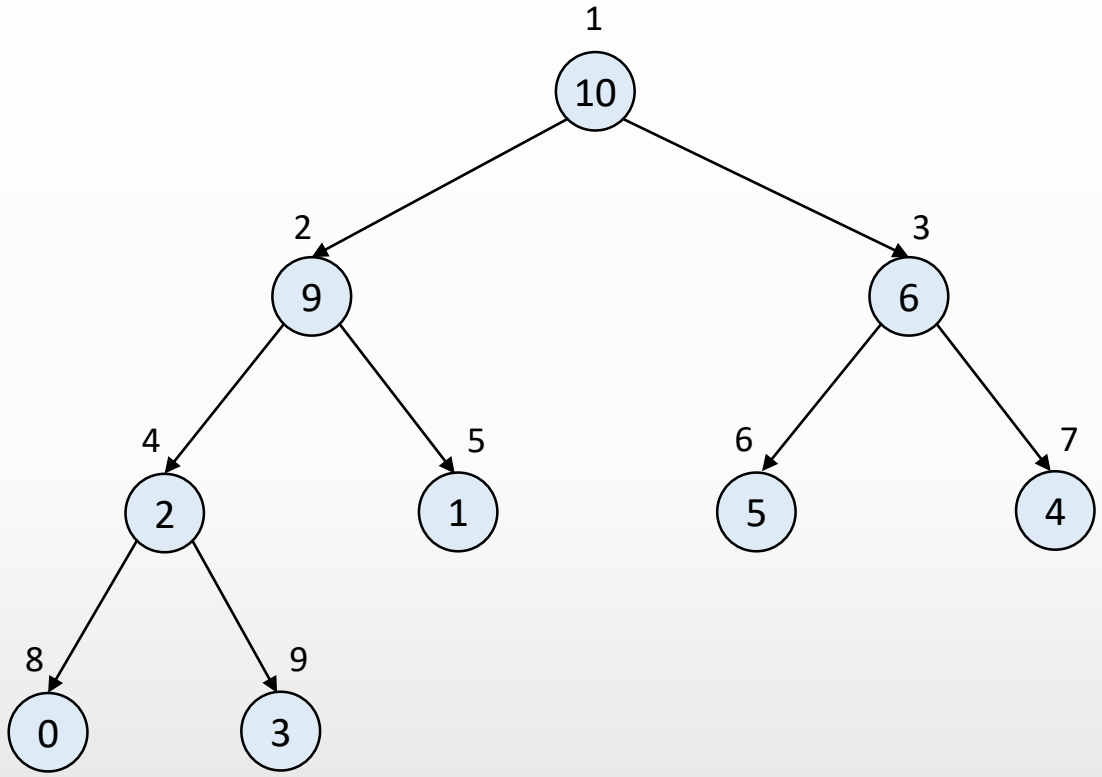


silMax()

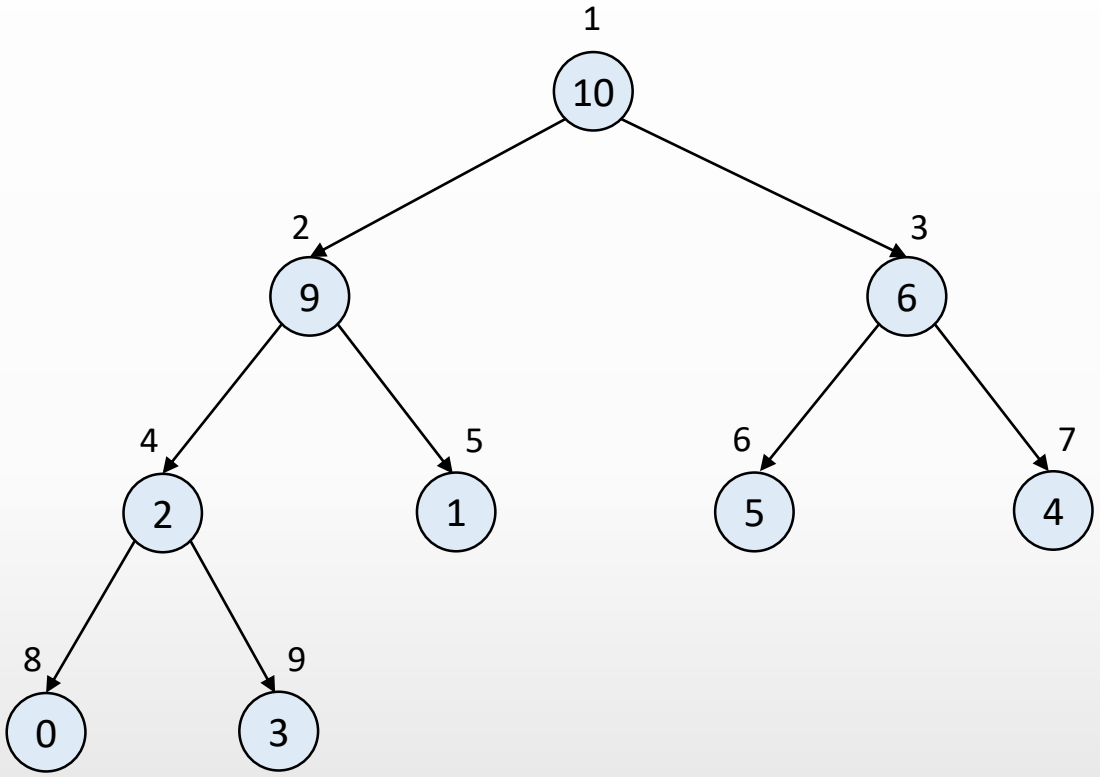
max = 9



Yukarıdan Aşağıya Heap Ağacına Dönüştürme

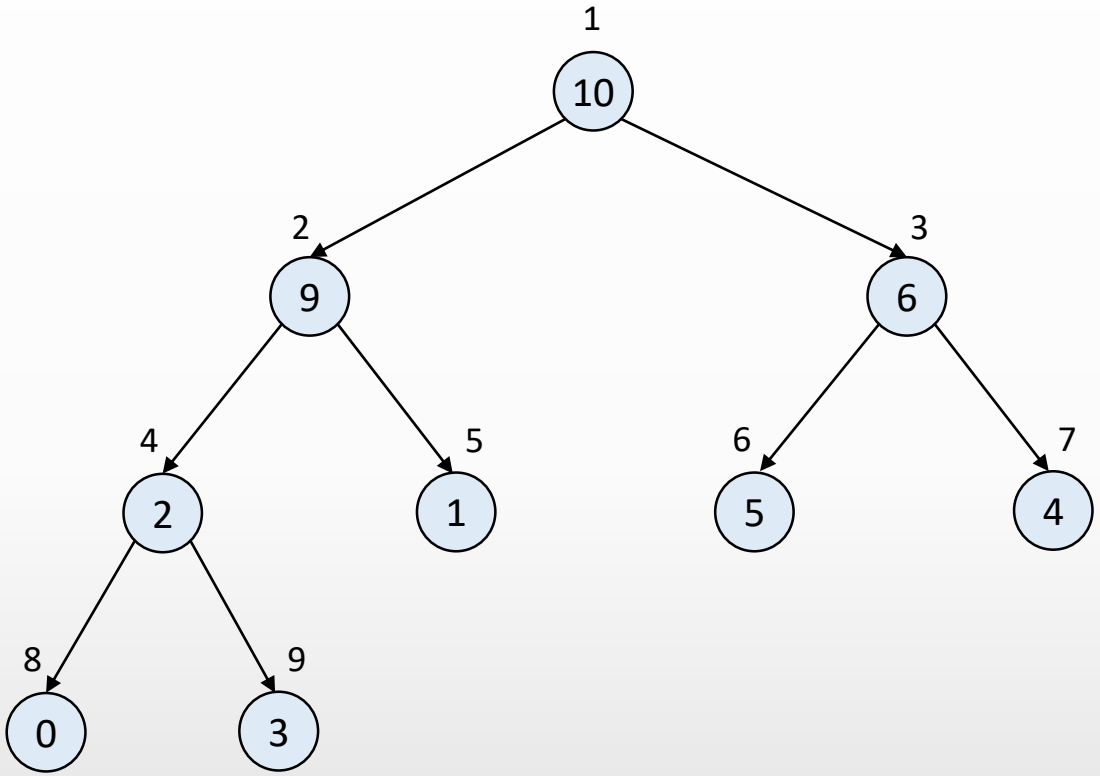

1
10
2
9
3
6
4
2
5
1
6
5
7
4
8
0
9
3

silMax()



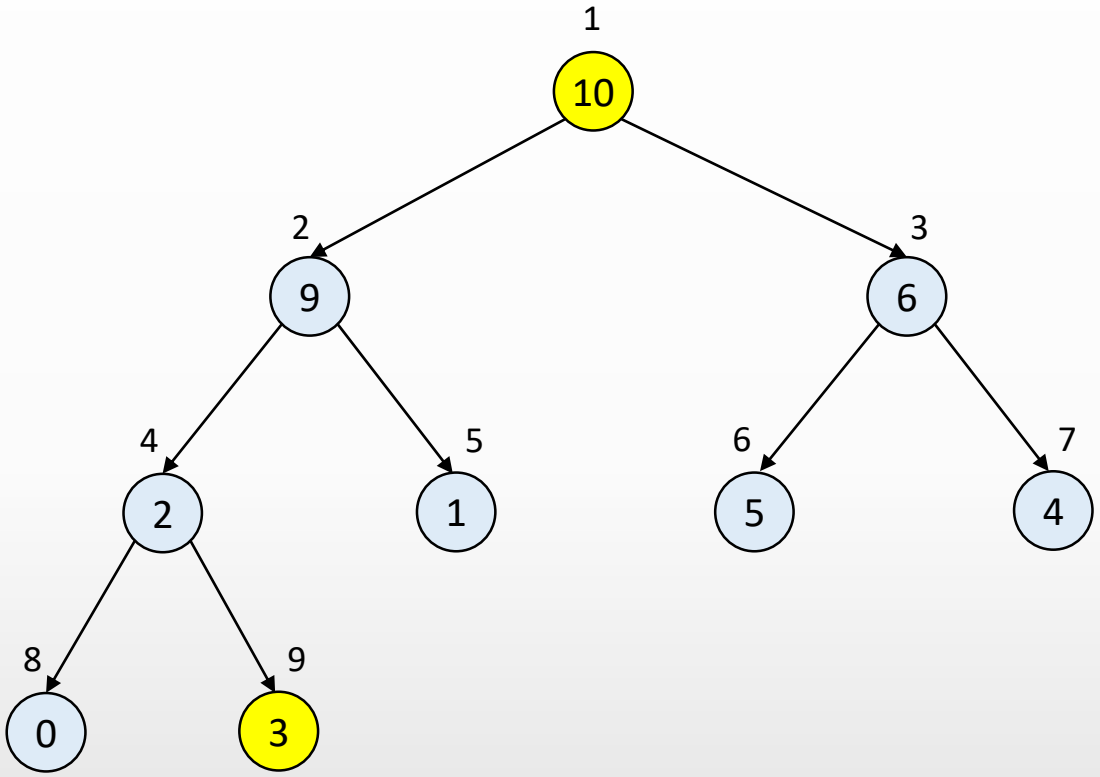
silMax()

max = 10



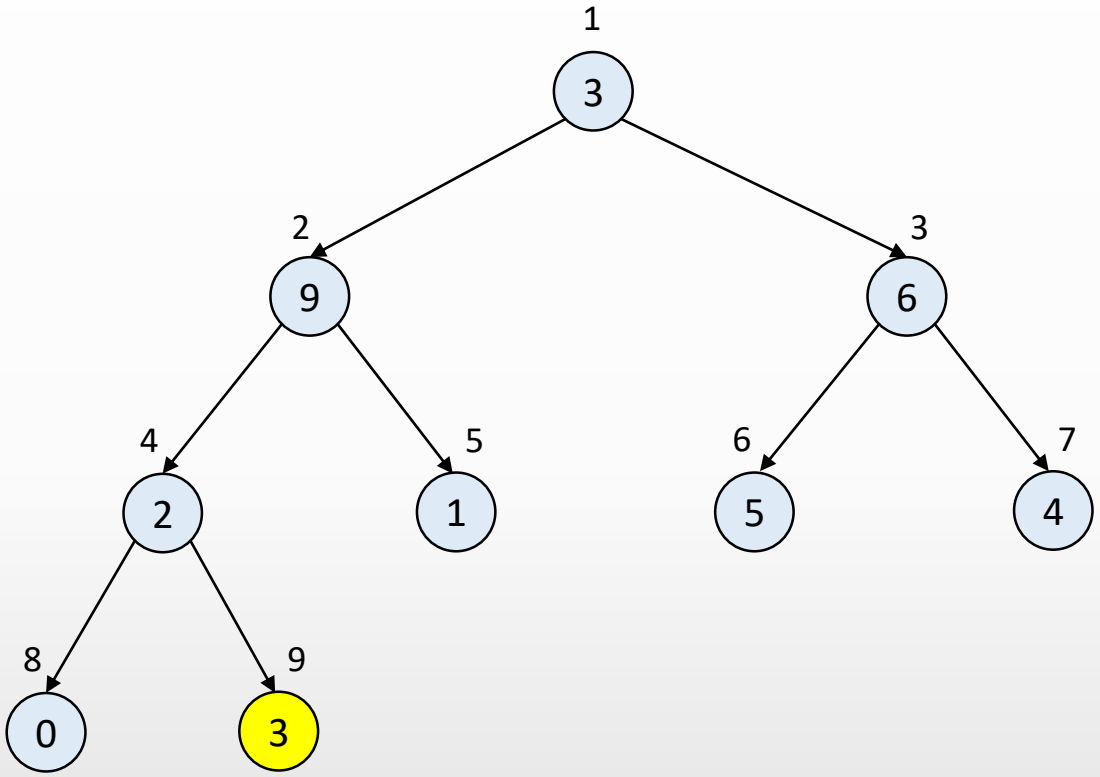
silMax()

max = 10



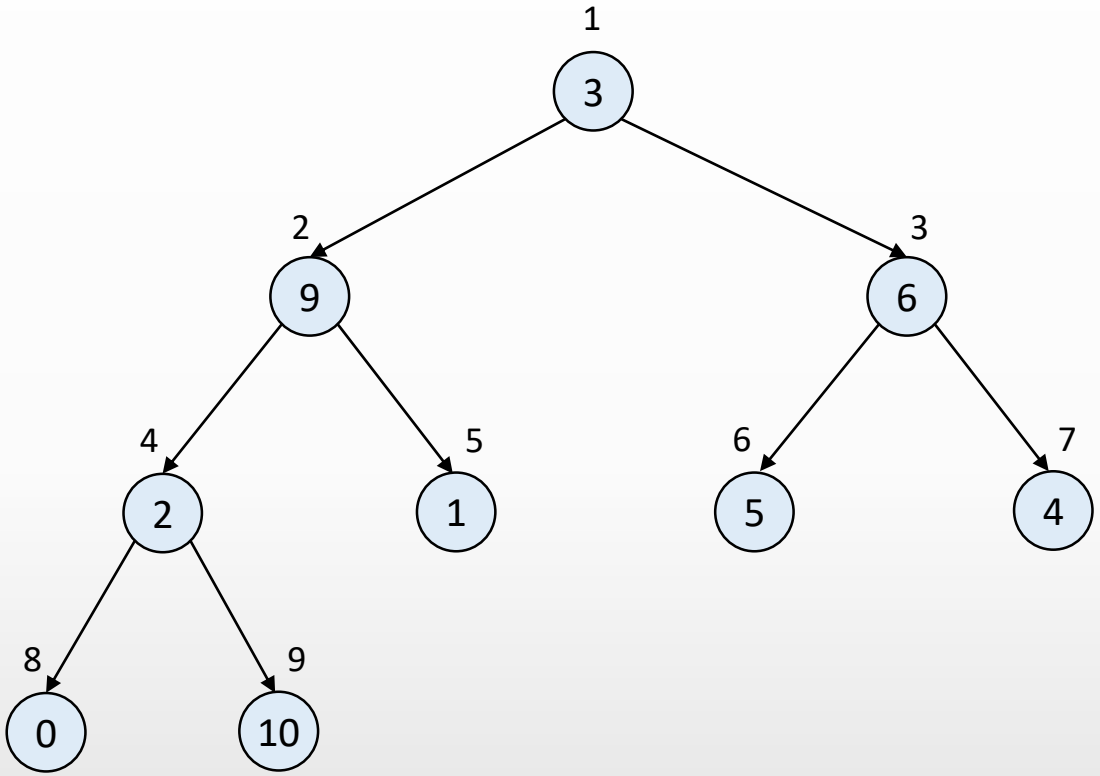
silMax()

max = 10



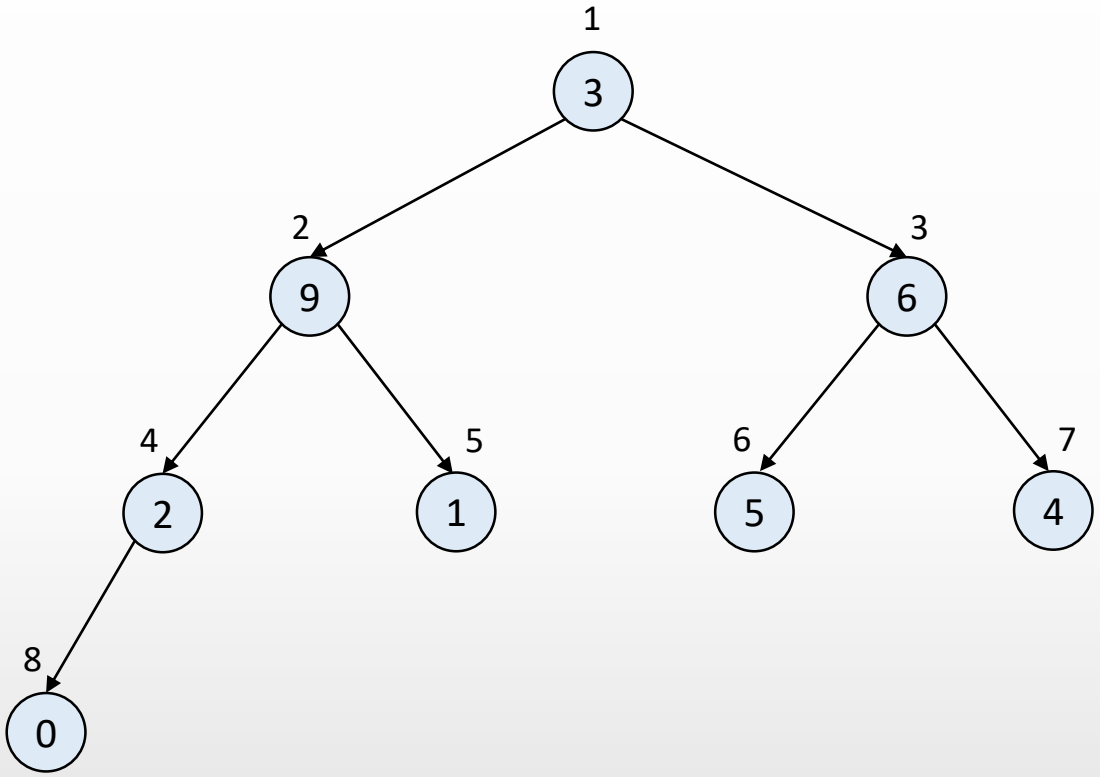
silMax()

max = 10



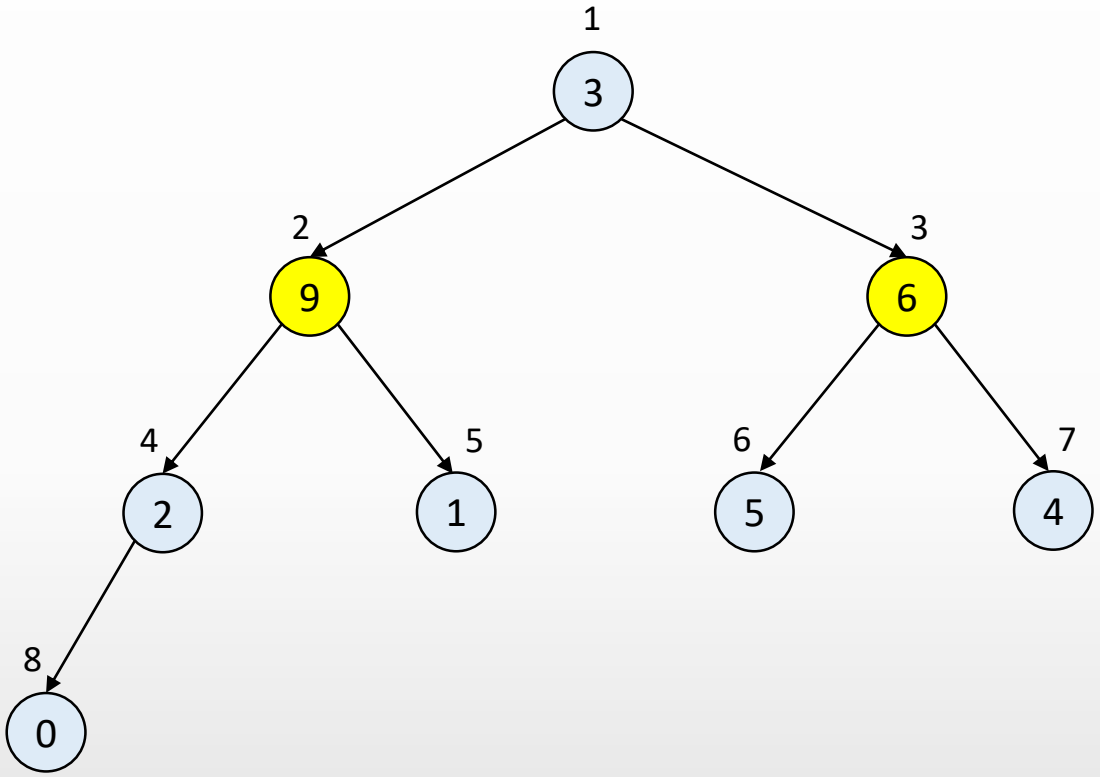
silMax()

max = 10



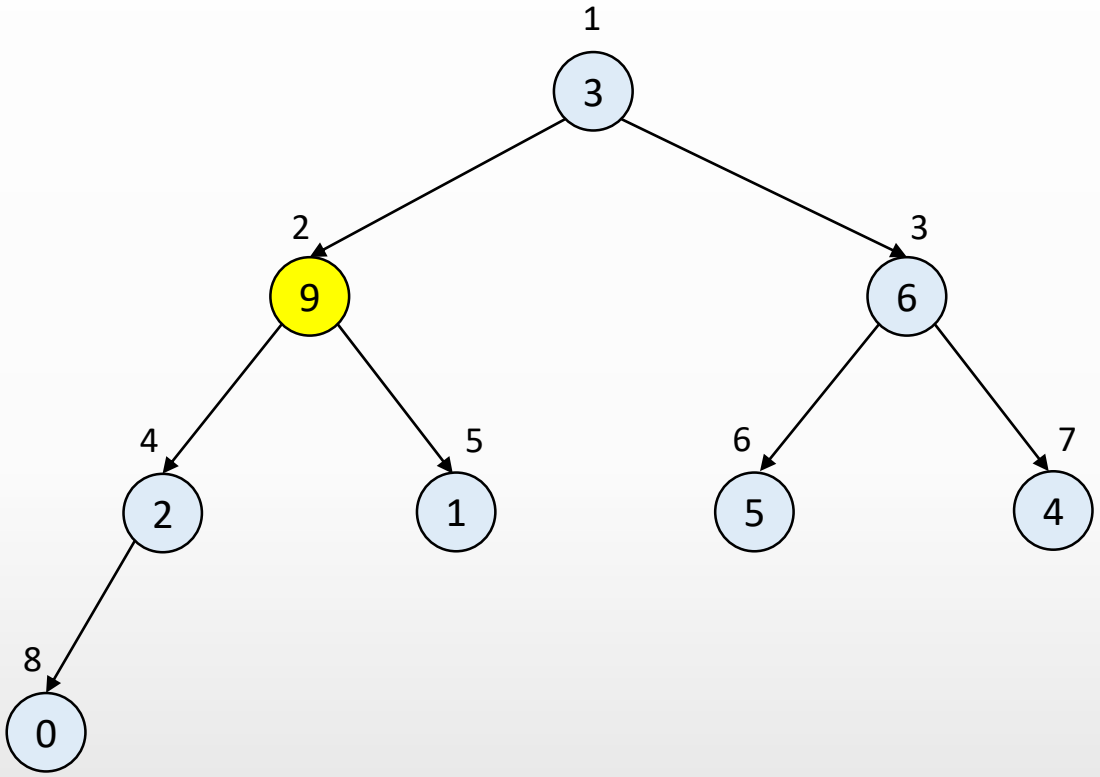
silMax()

max = 10



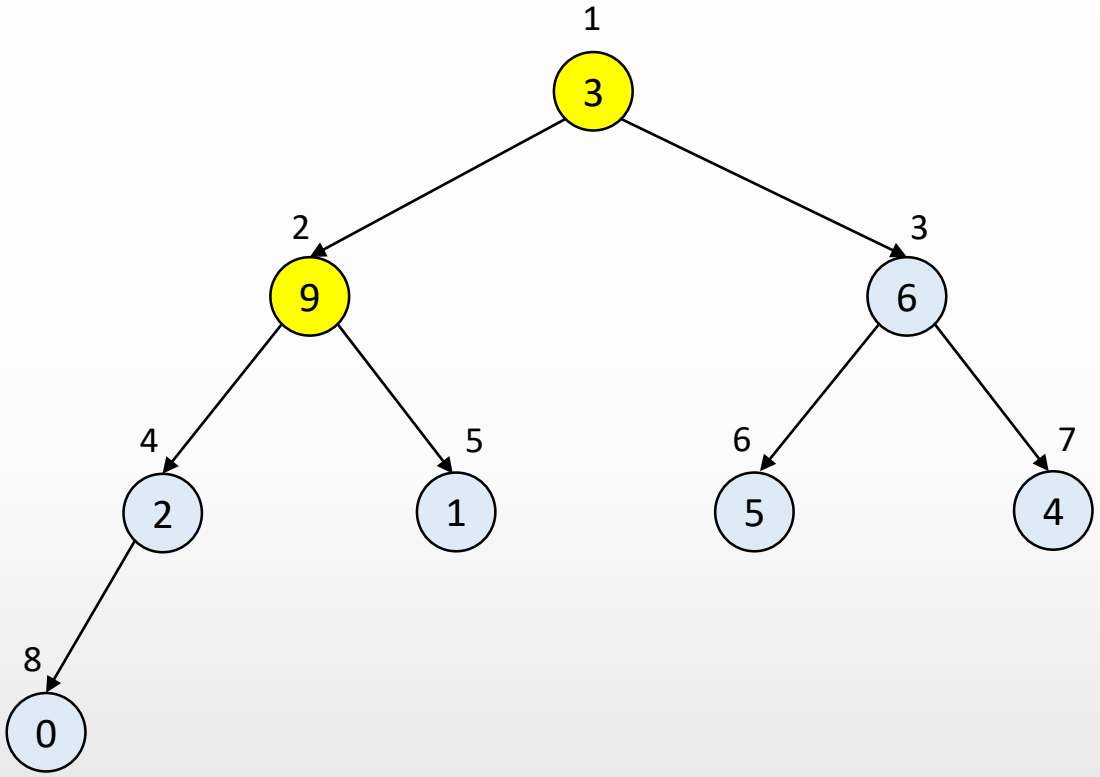
silMax()

max = 10



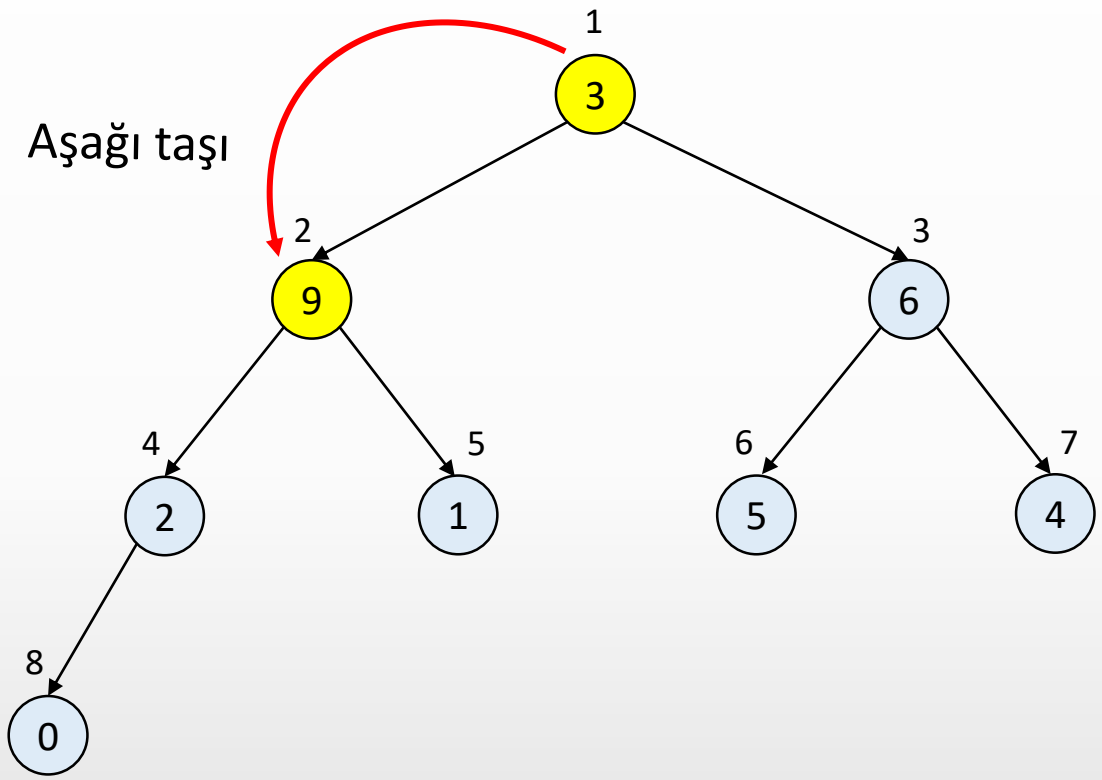
silMax()

max = 10



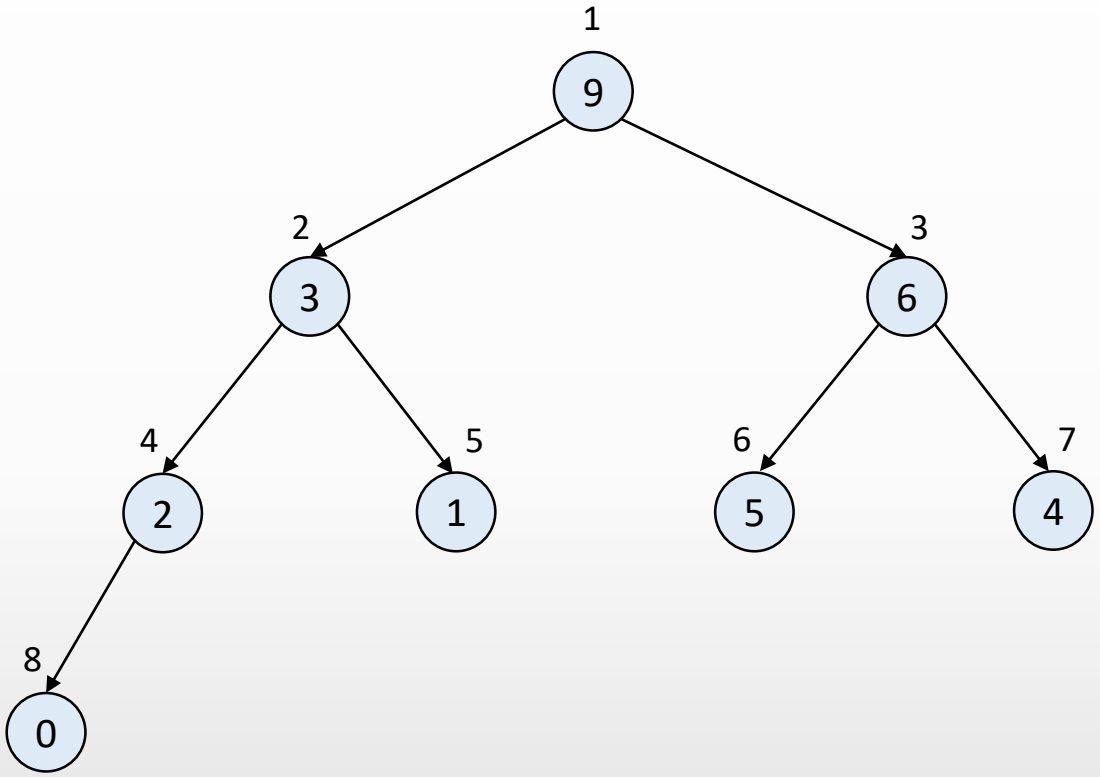
silMax()

max = 10



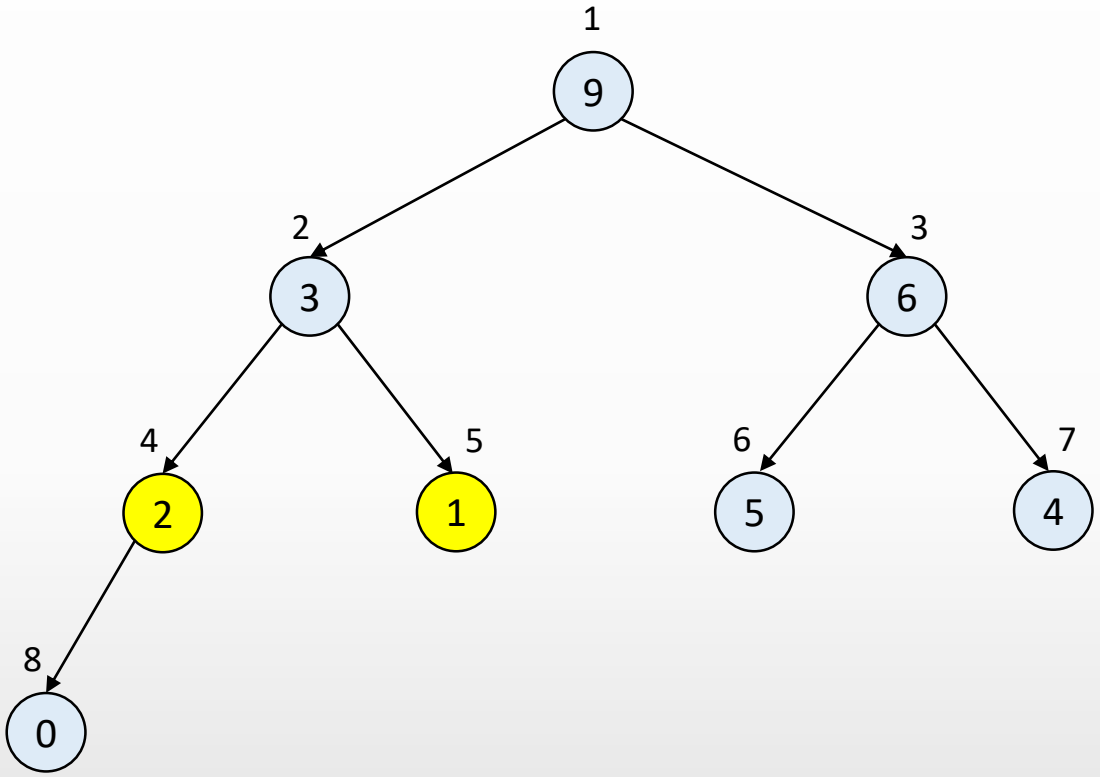
silMax()

max = 10



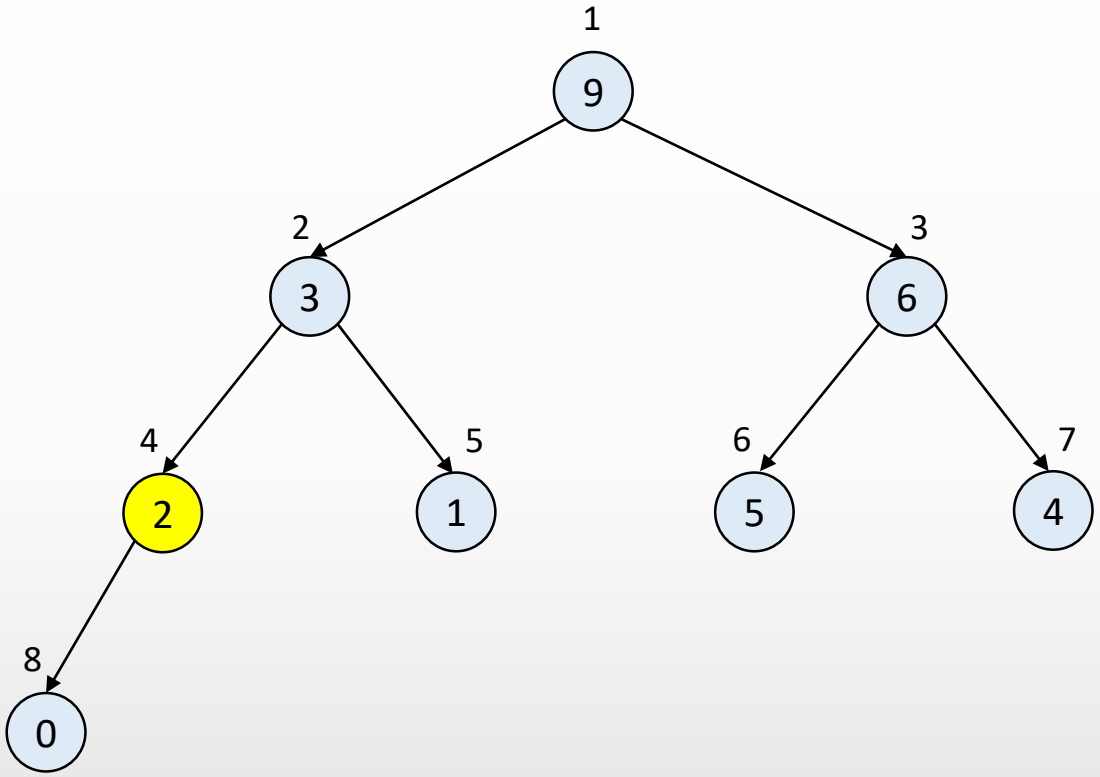
silMax()

max = 10



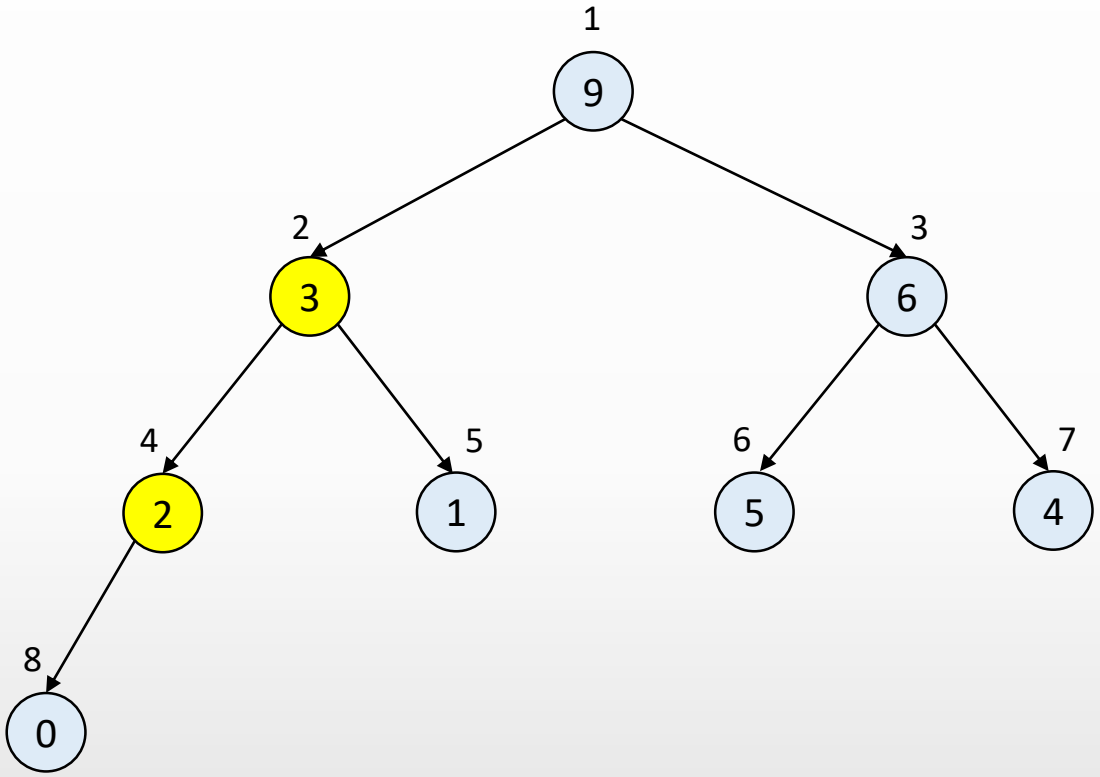
silMax()

max = 10



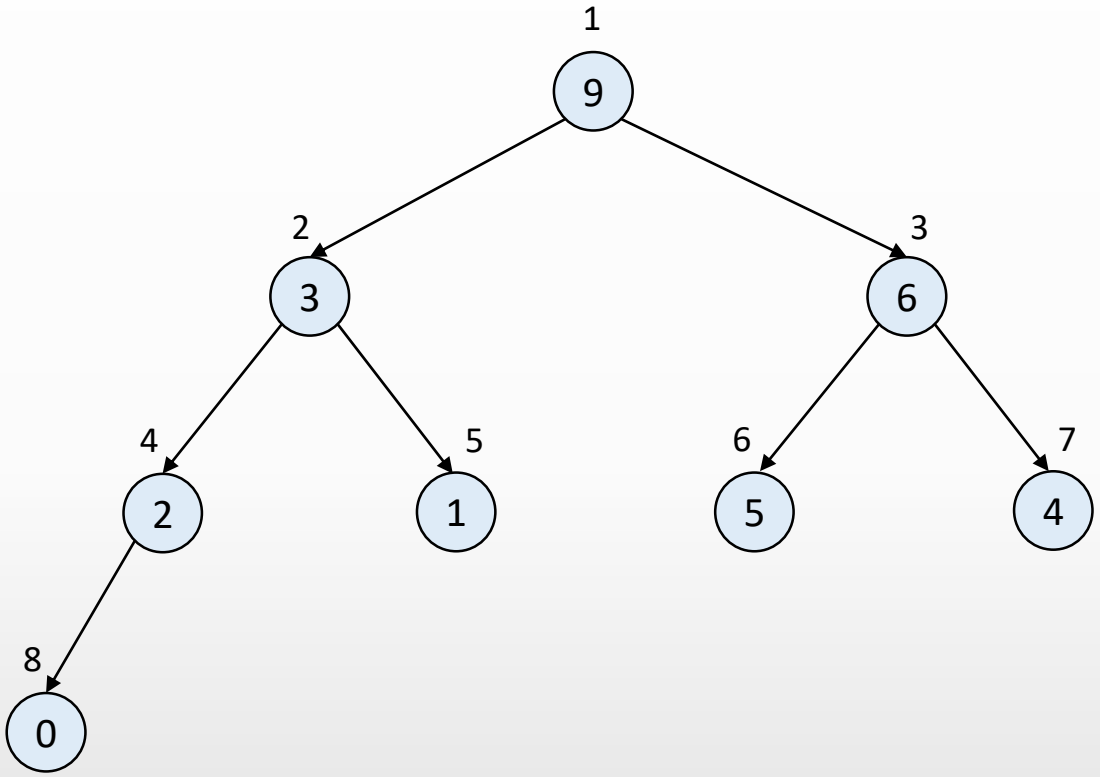
silMax()

max = 10



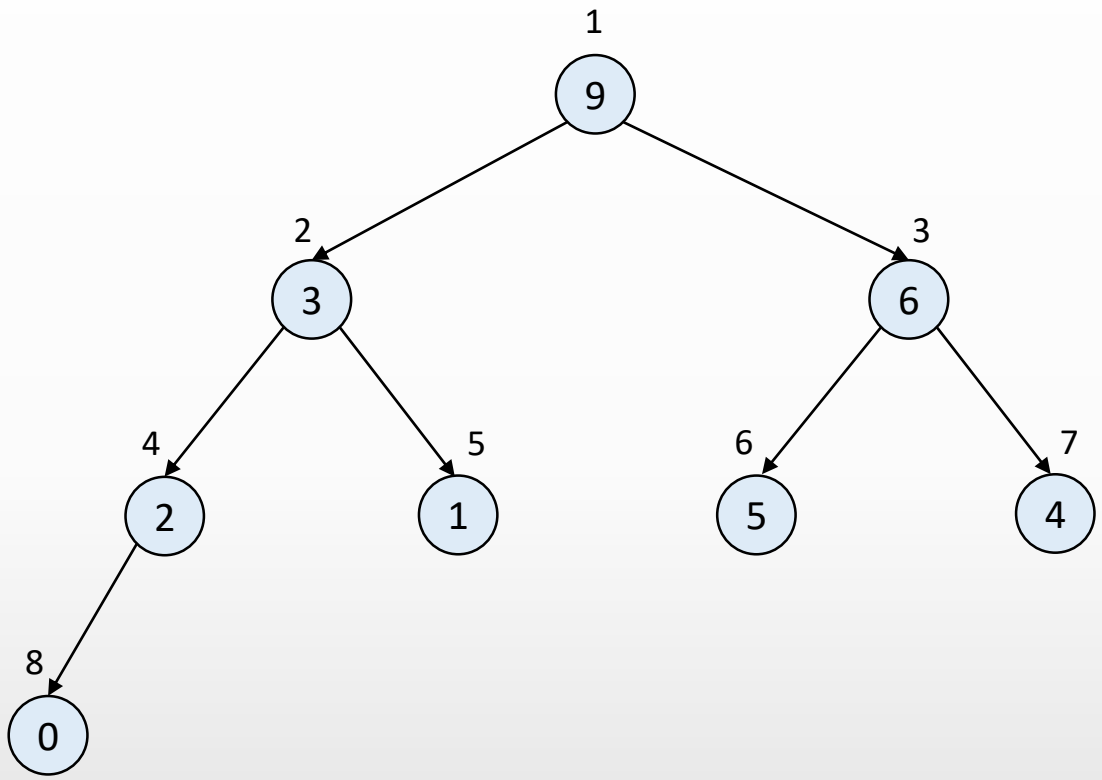
silMax()

max = 10



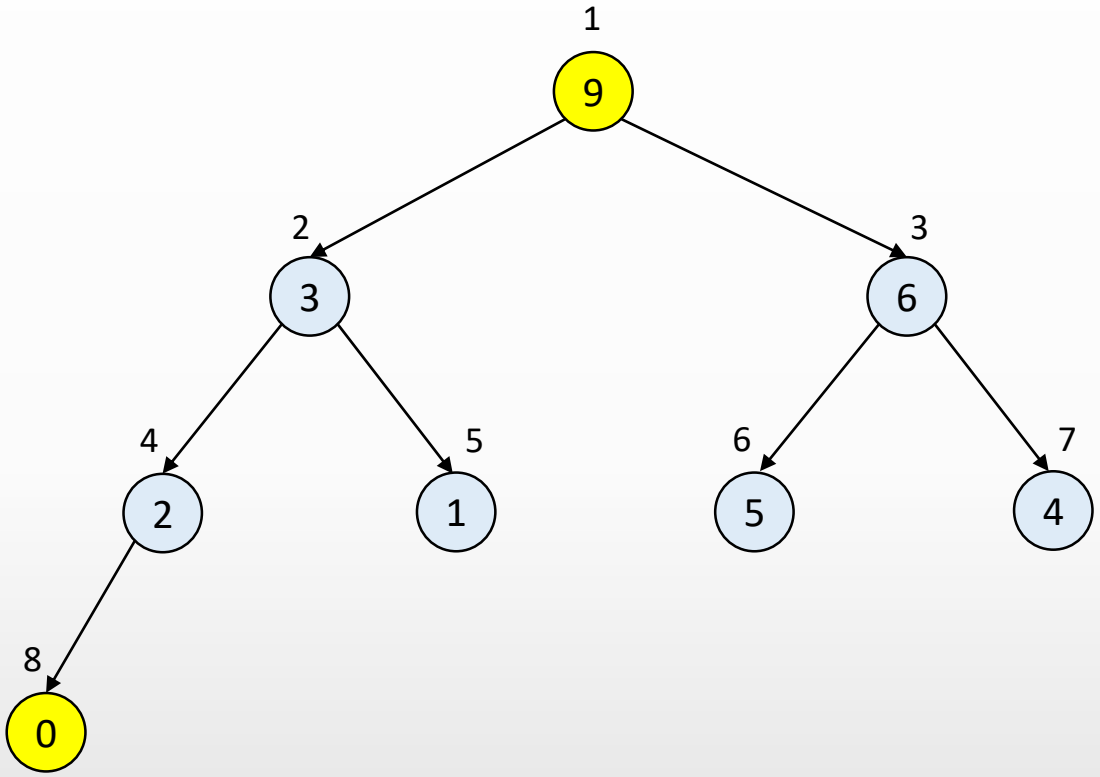
silMax()

max = 9



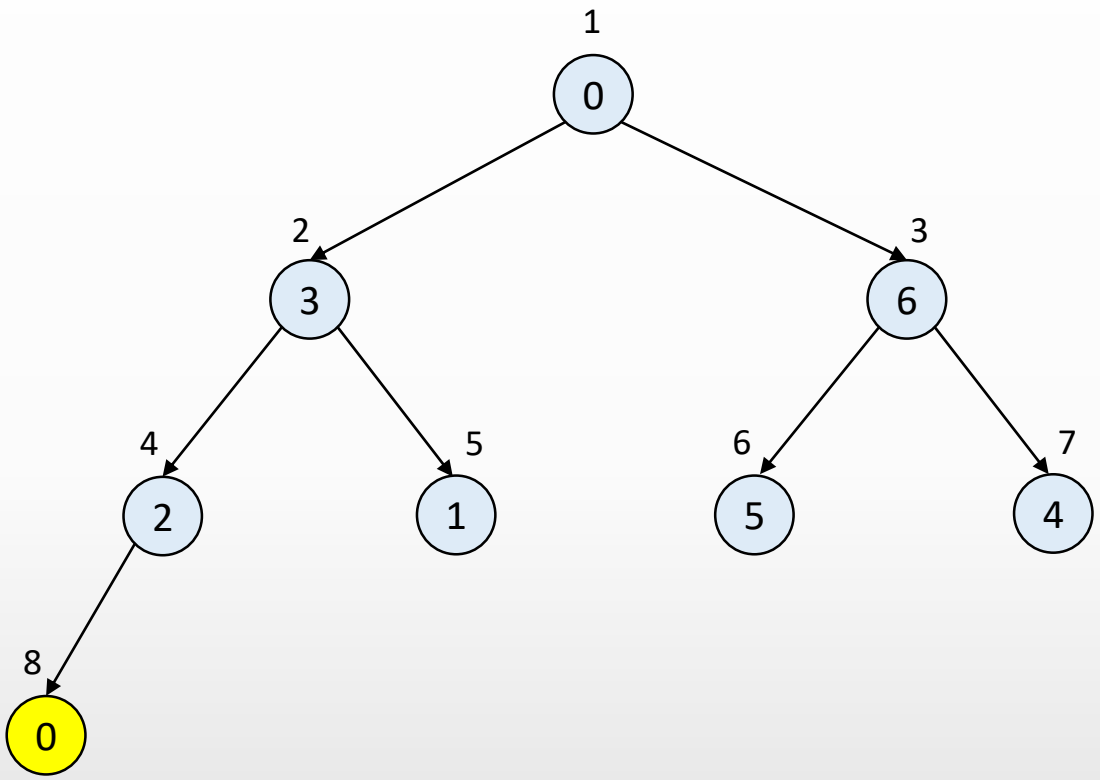
silMax()

max = 9



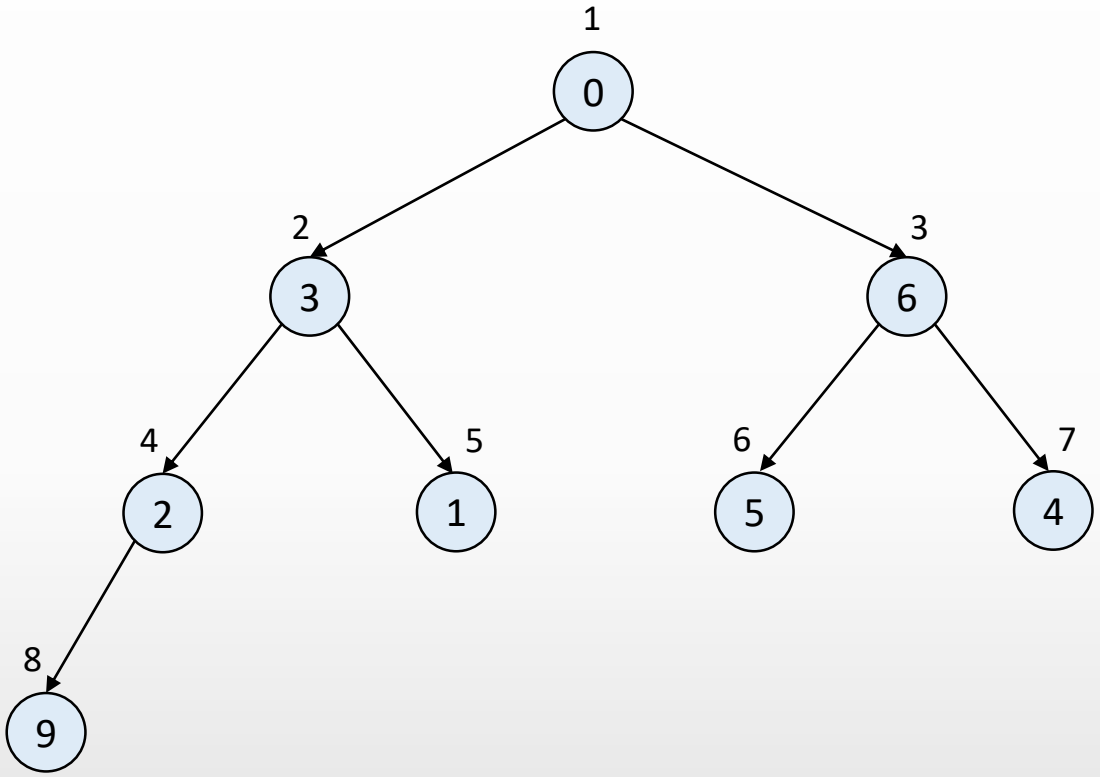
silMax()

max = 9



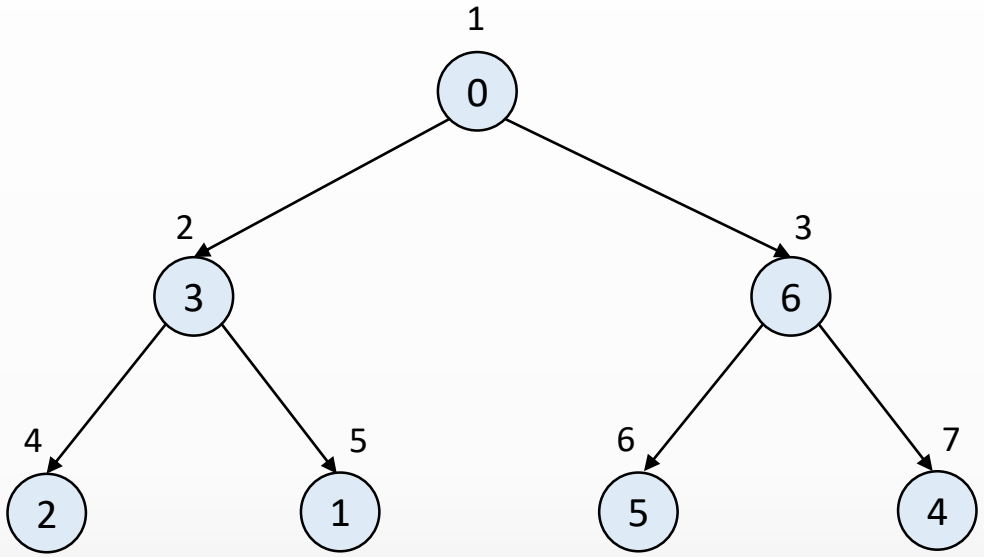
silMax()

max = 9



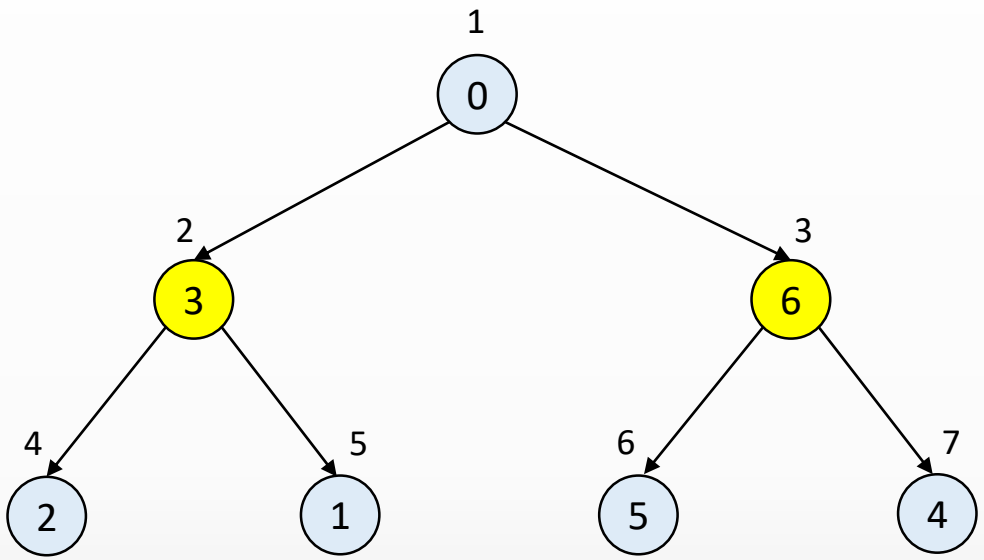
silMax()

max = 9



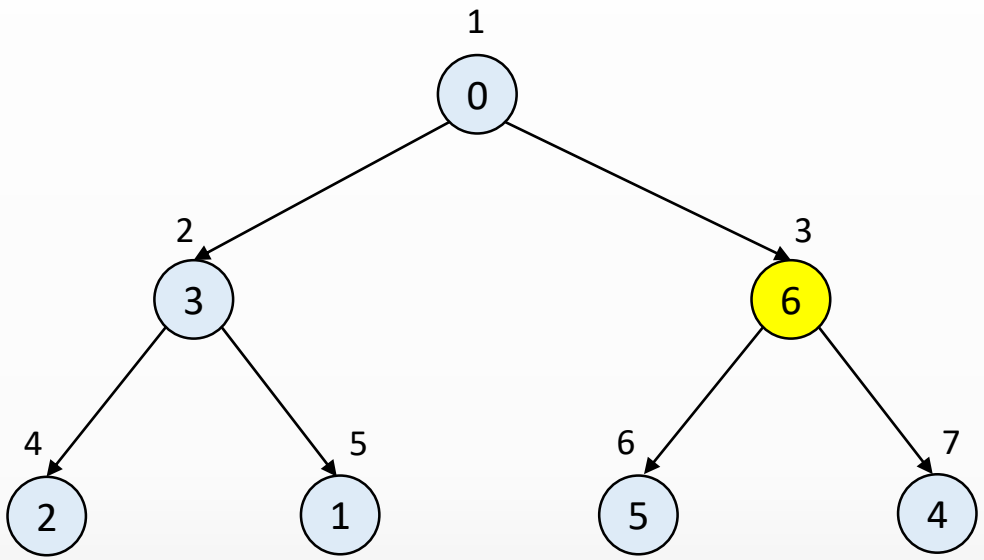
silMax()

max = 9



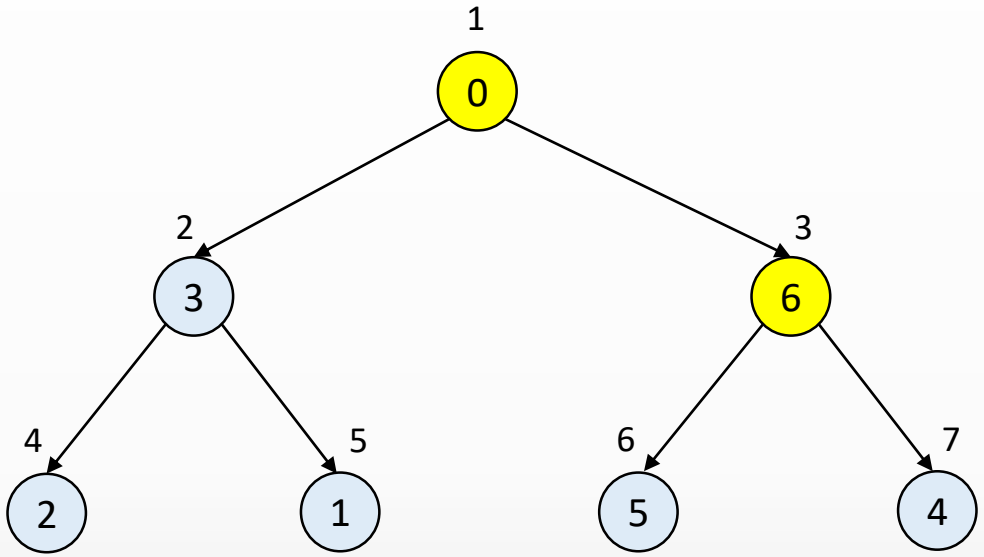
silMax()

max = 9



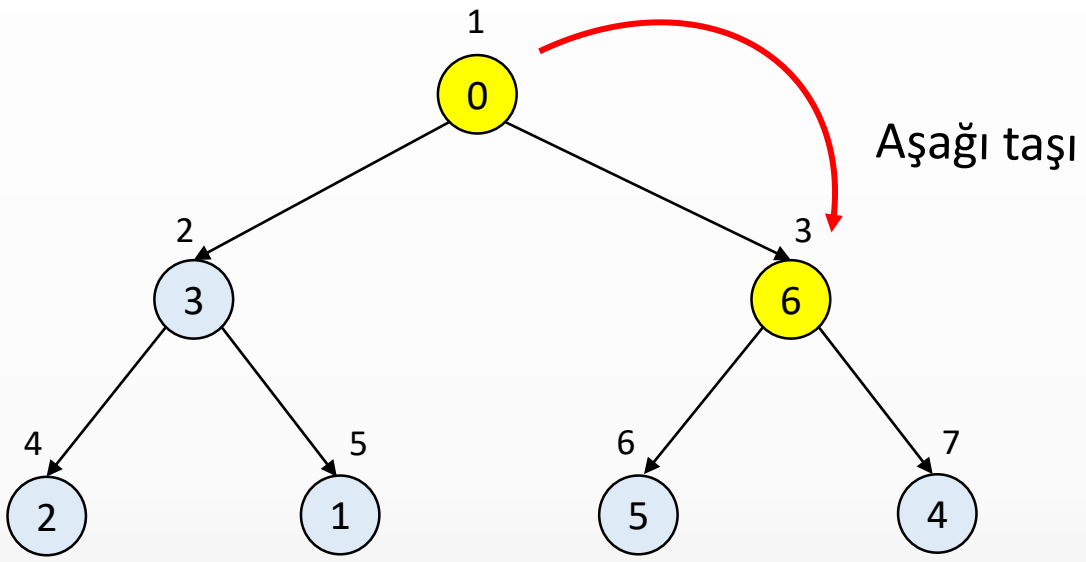
silMax()

max = 9



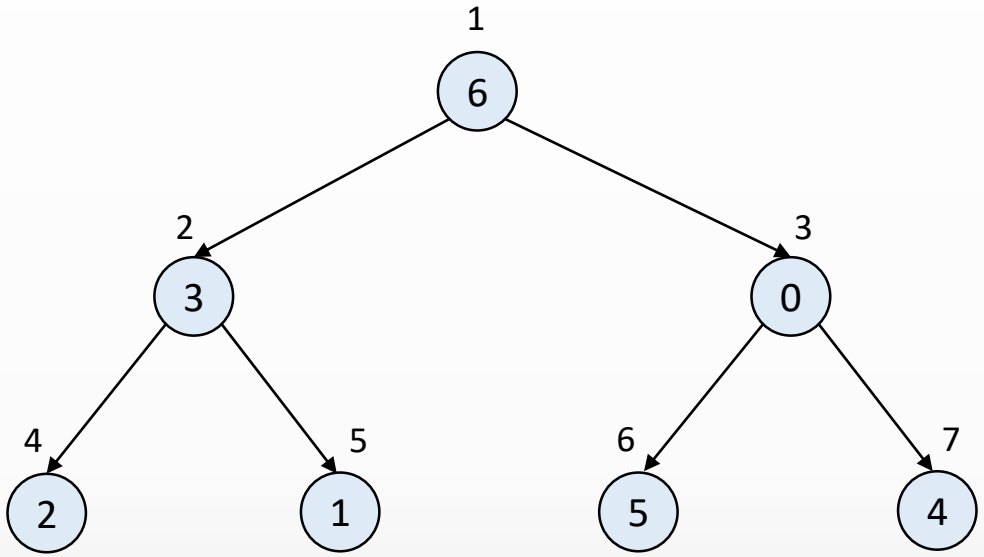
silMax()

max = 9



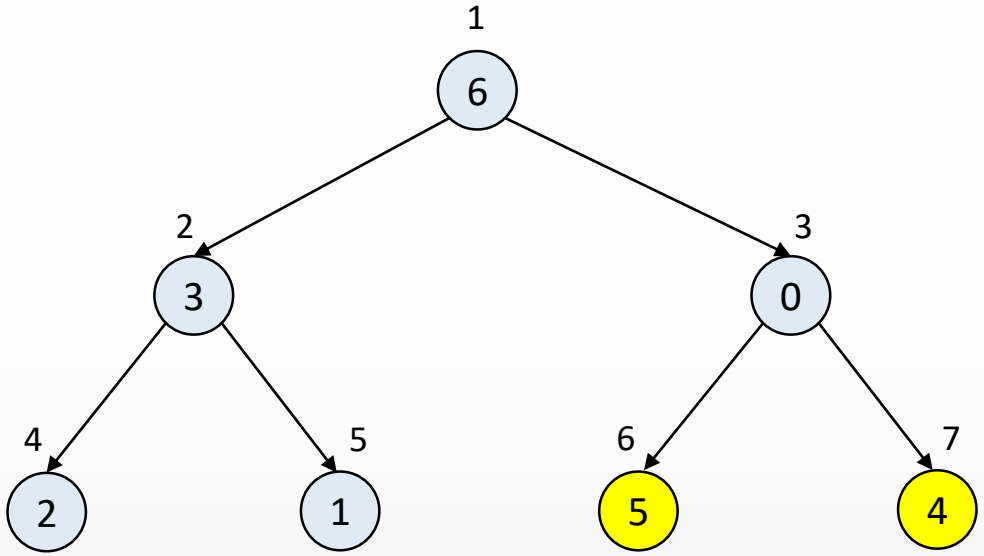
silMax()

max = 9



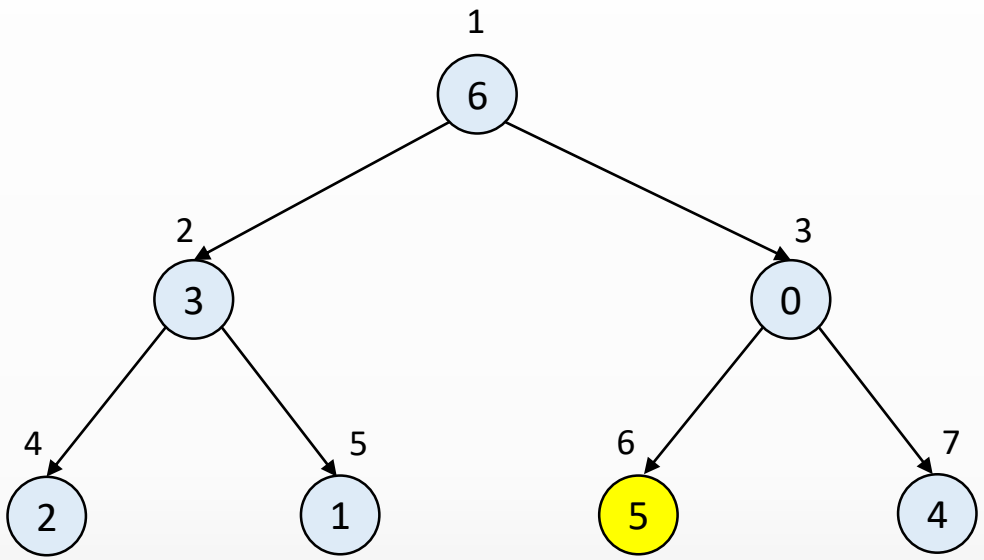
silMax()

max = 9



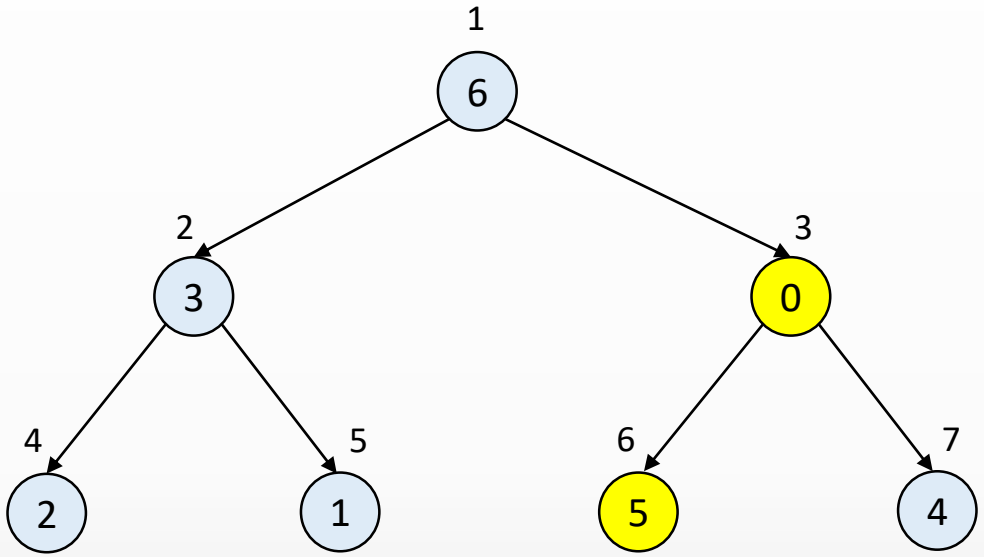
silMax()

max = 9



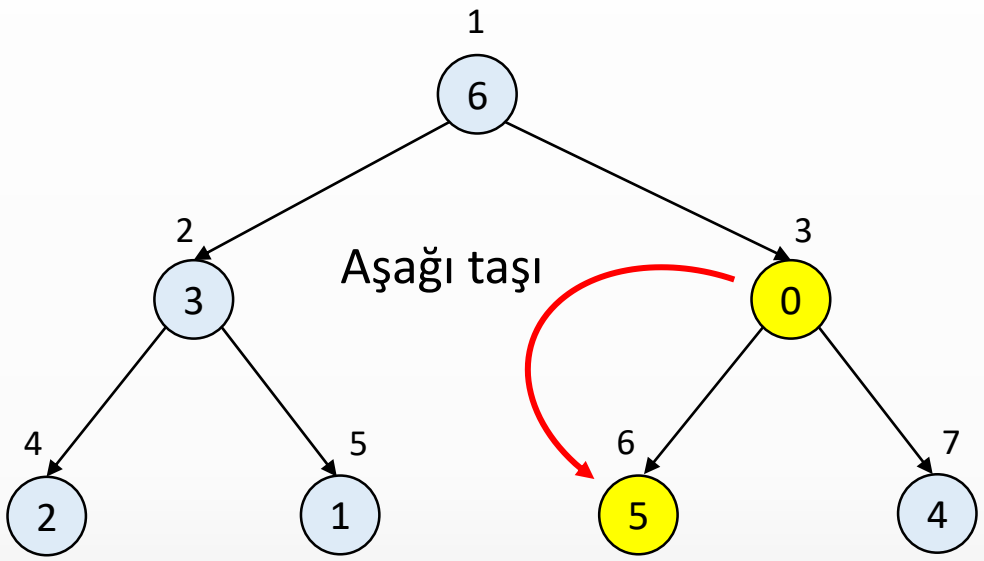
silMax()

max = 9



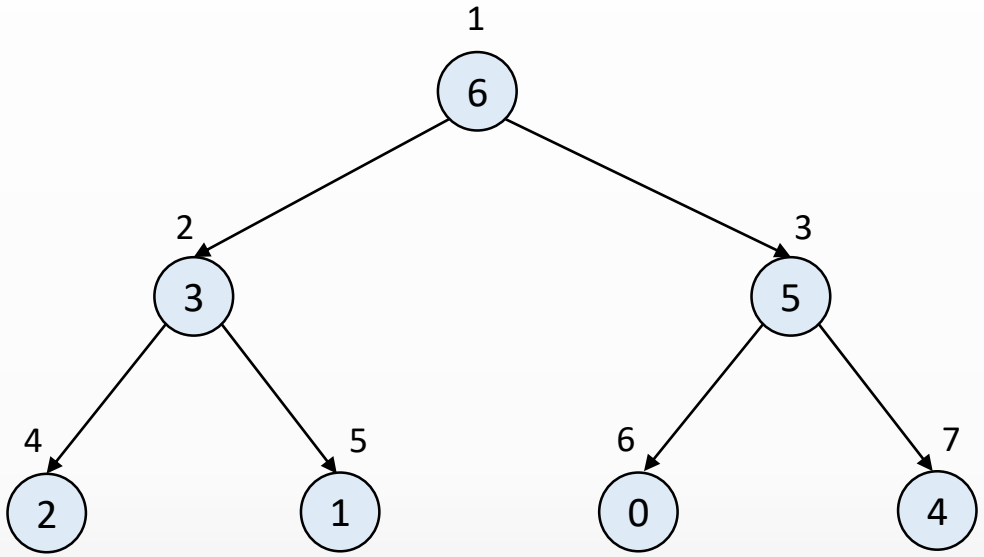
silMax()

max = 9



silMax()

max = 9

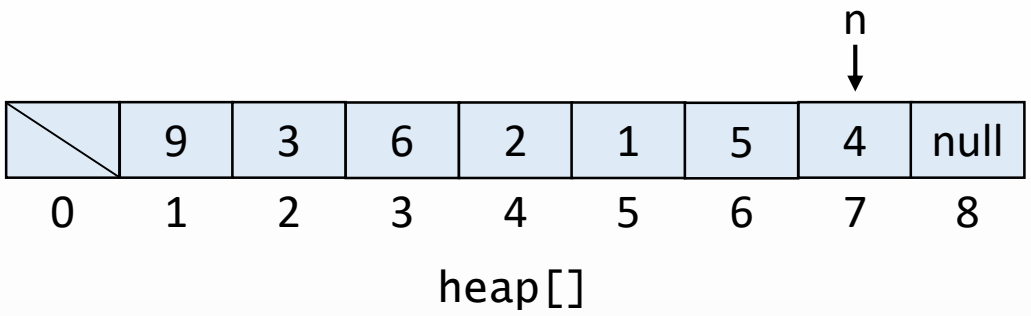


Max Heap Ağacında En Büyük Elemanı Silme

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

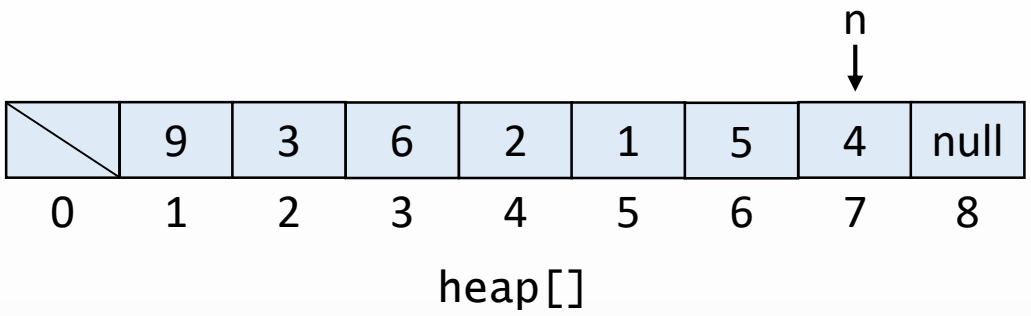
heap[]

```
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

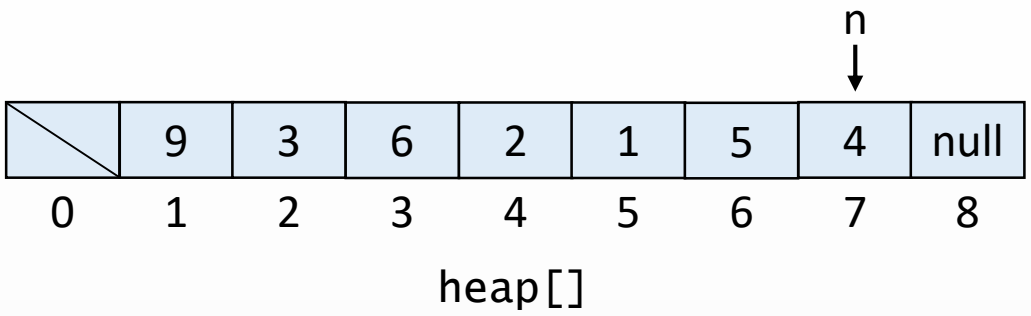
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



n = 7

silMax()

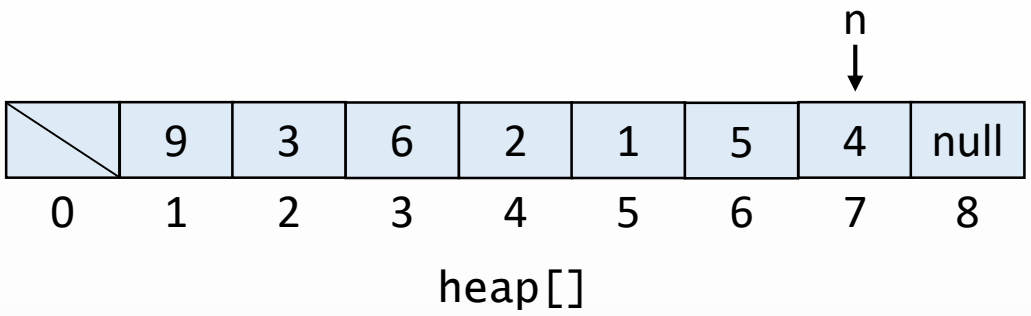
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
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

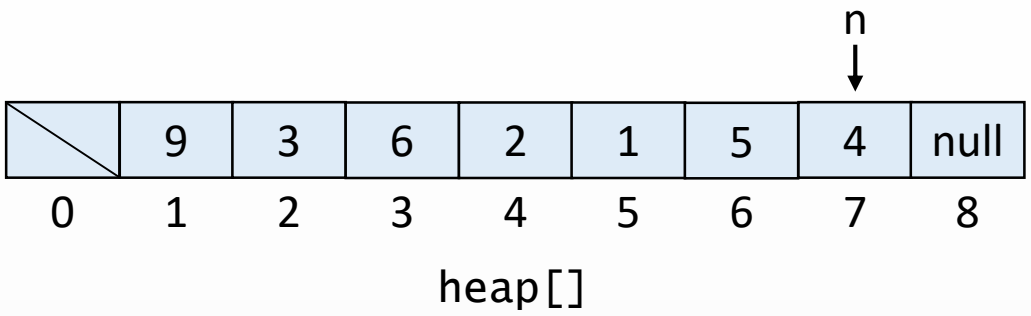
silMax()



```
max = 9
n = 7
```

```
silMax()
```

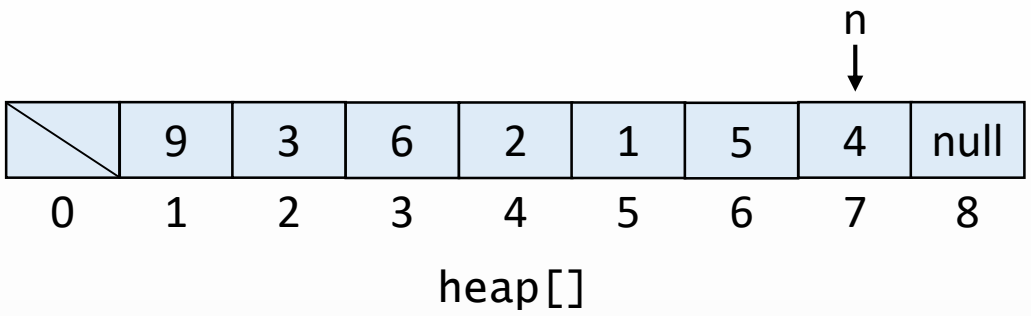
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 7
```

```
silMax()
```

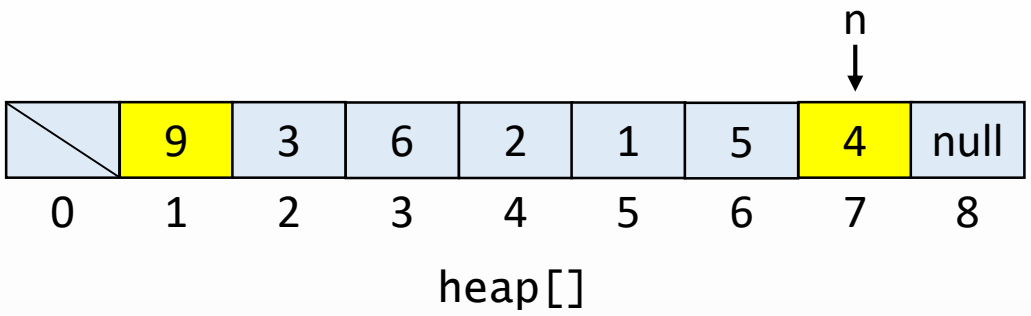
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 7
```

```
silMax()
```

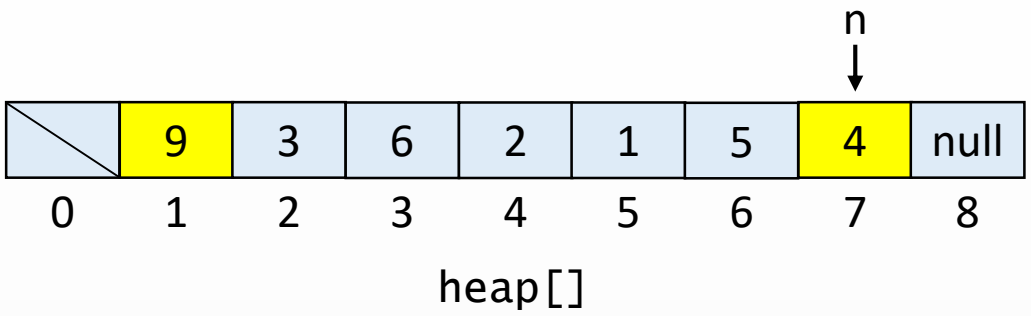
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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()
```

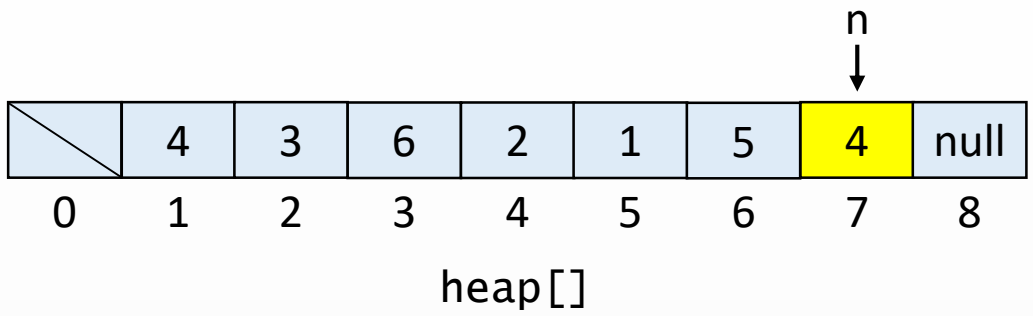
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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()
```

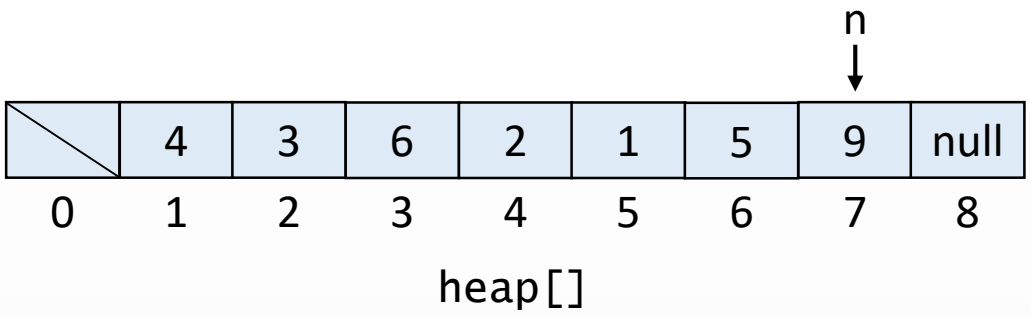
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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()
```

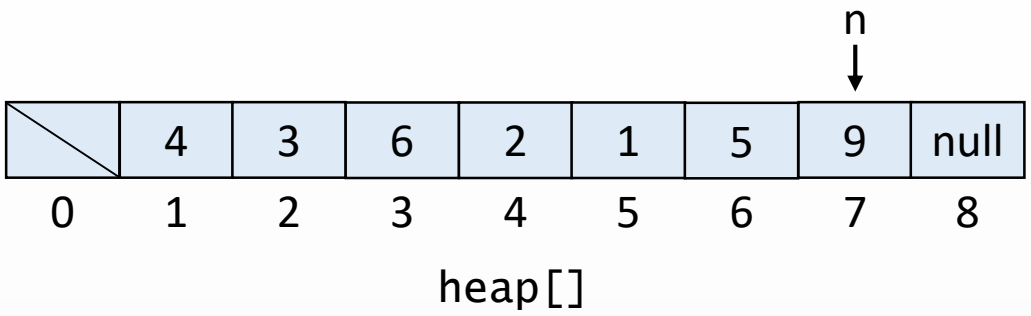
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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()
```

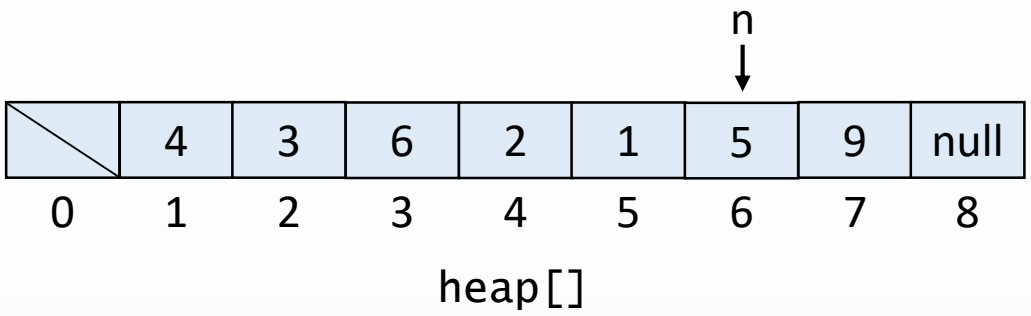
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 7
```

```
silMax()
```

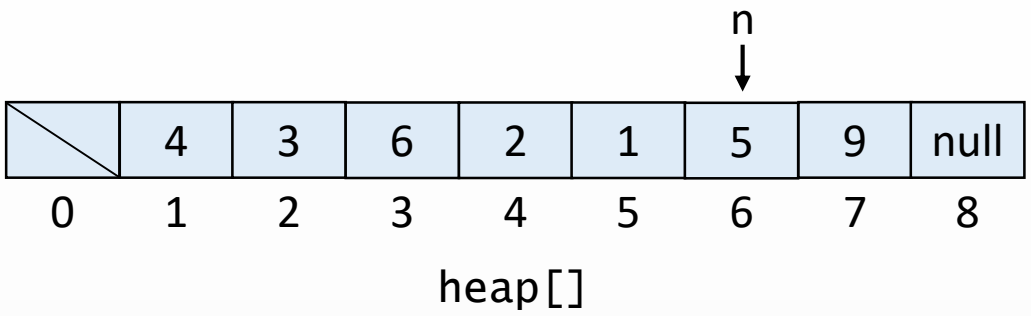
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 6
```

```
silMax()
```

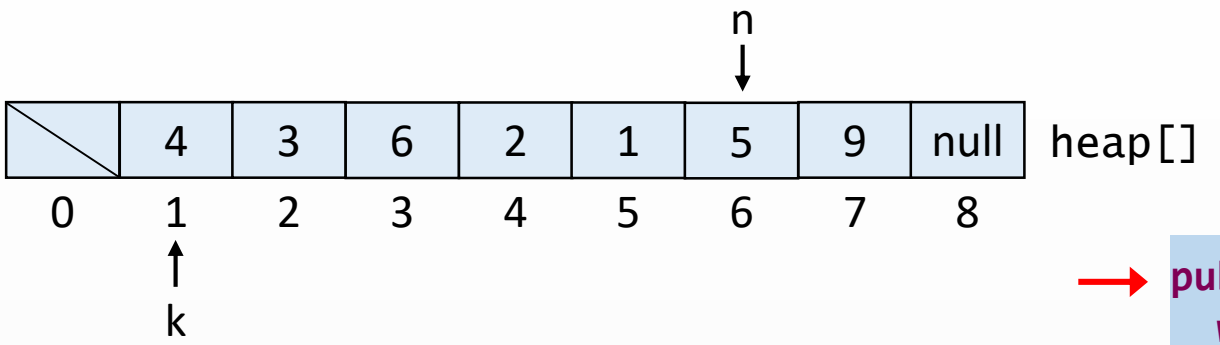
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 6
```

```
silMax()
```

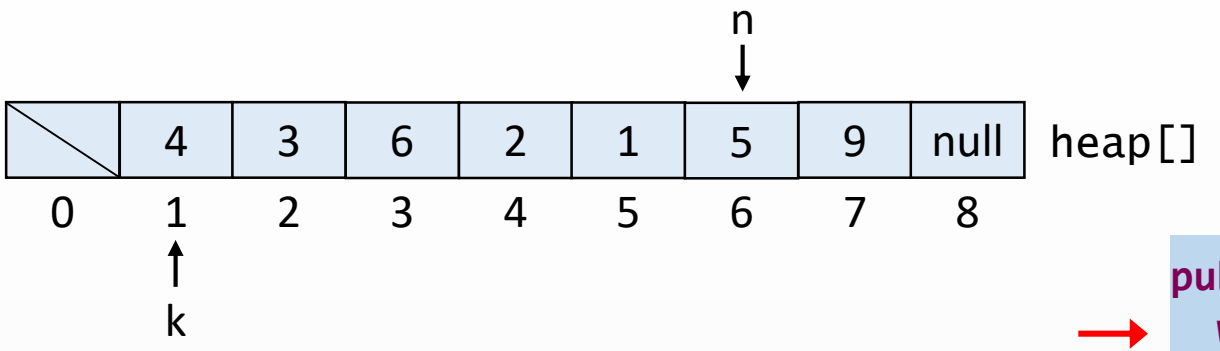
```java
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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()
```

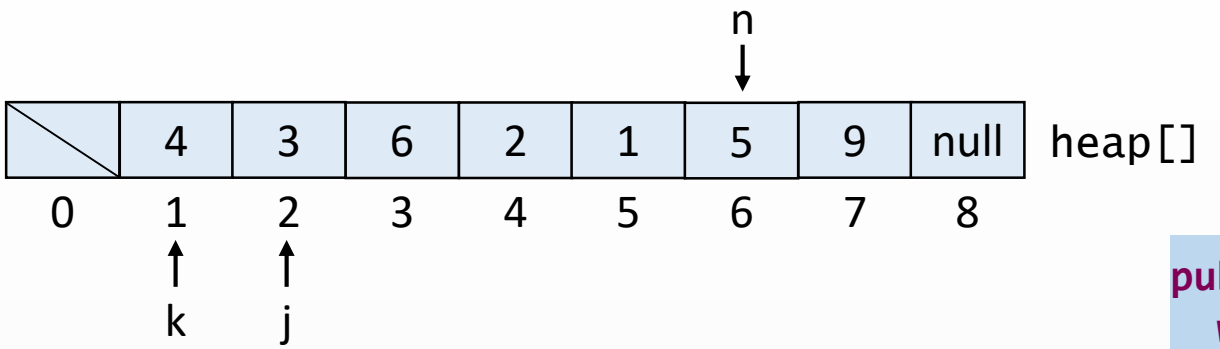
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

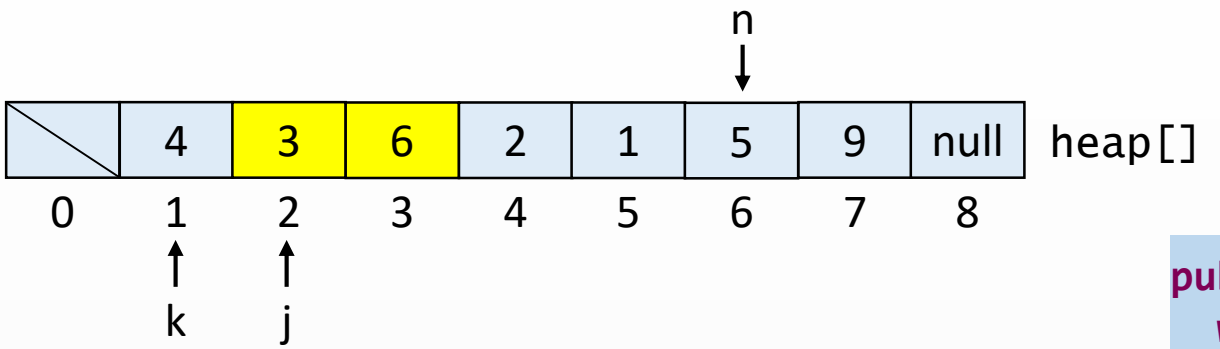
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

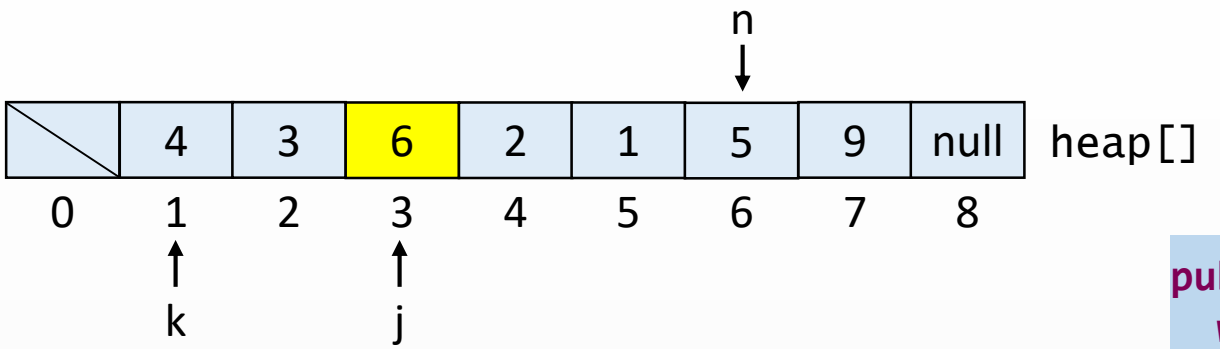
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

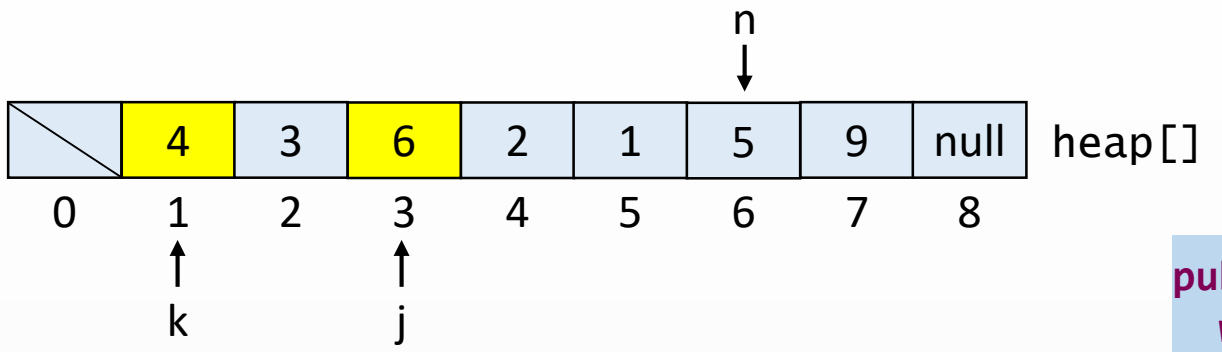
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

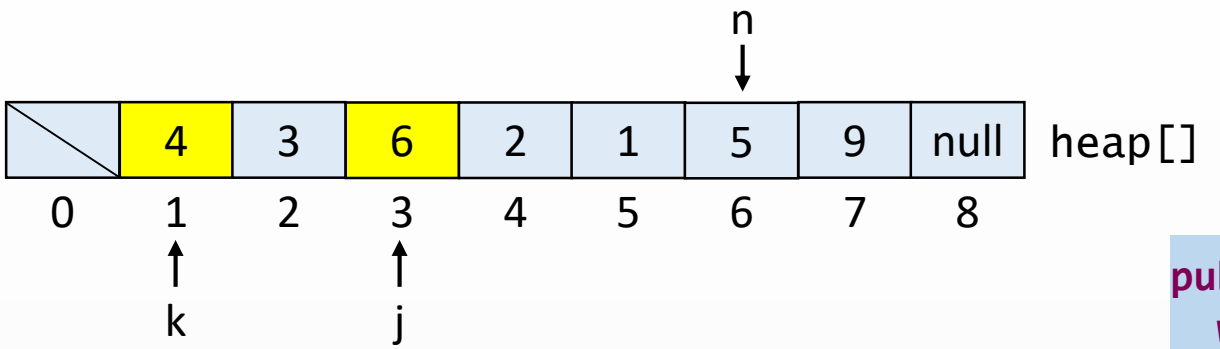
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

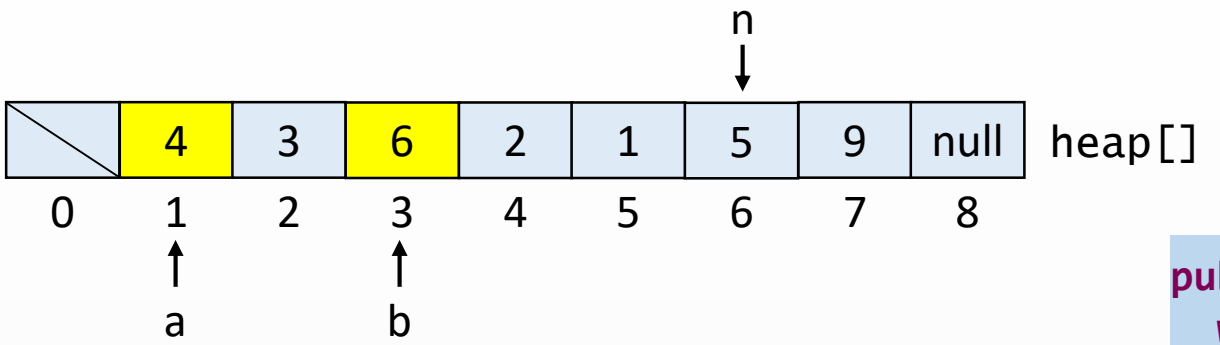
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

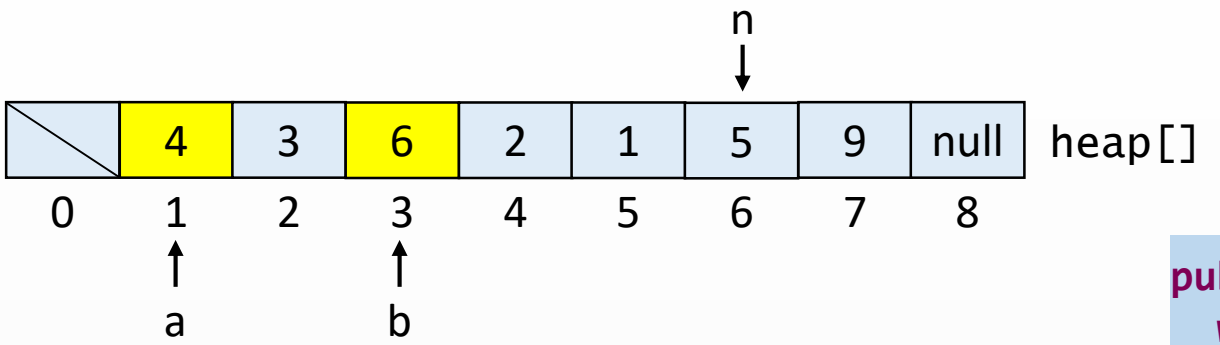
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

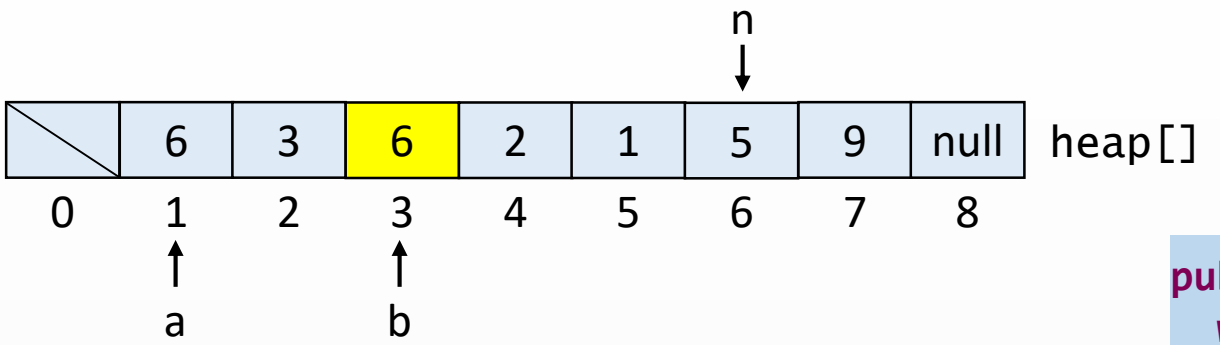
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

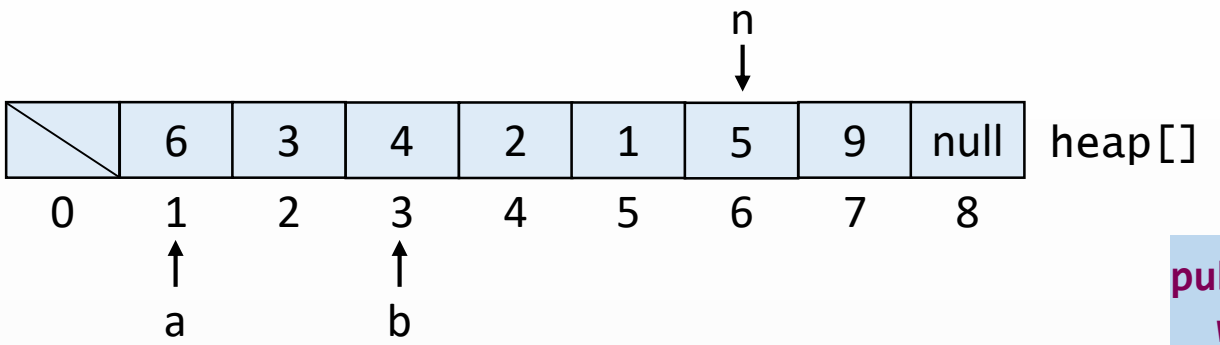
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

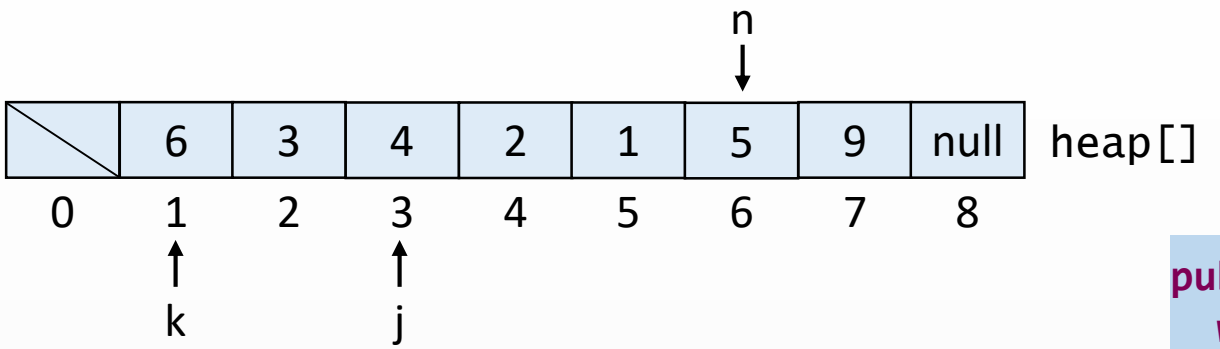
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

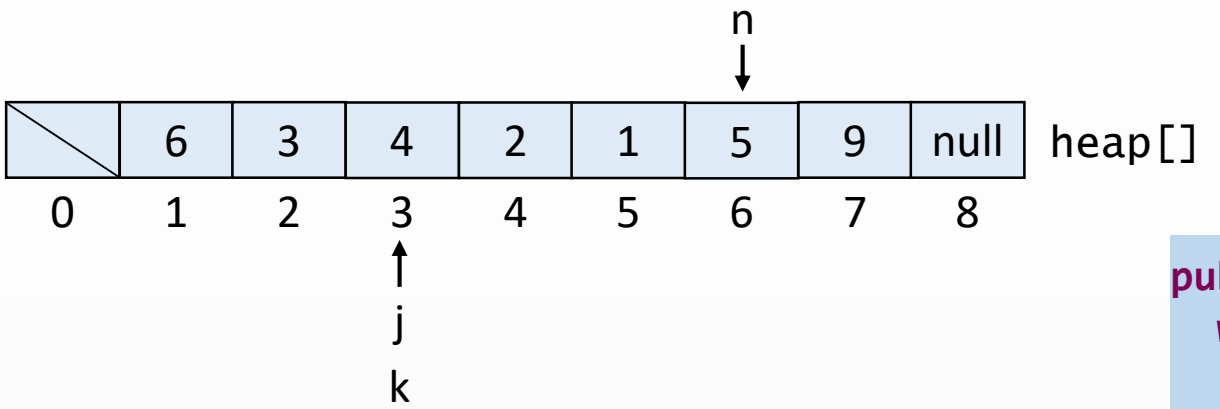
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

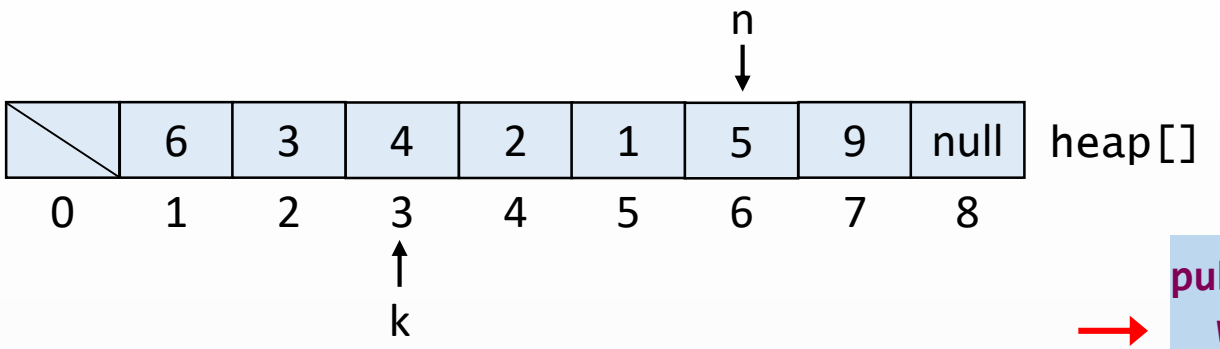
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

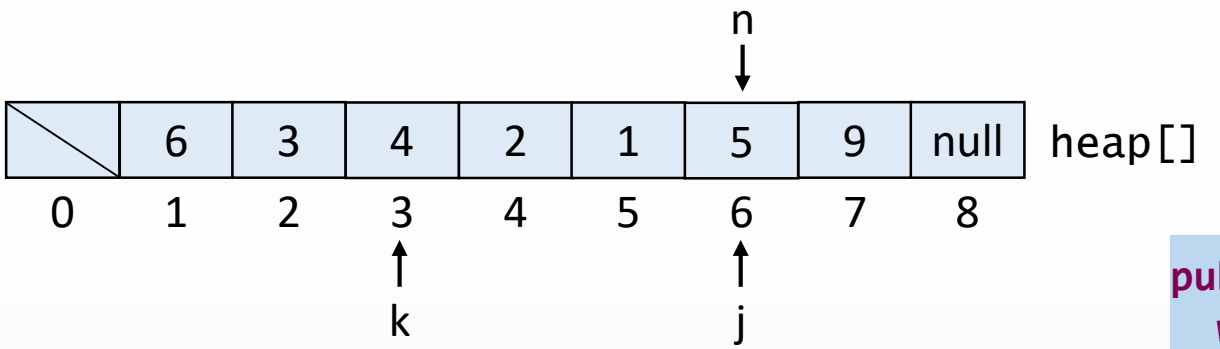
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

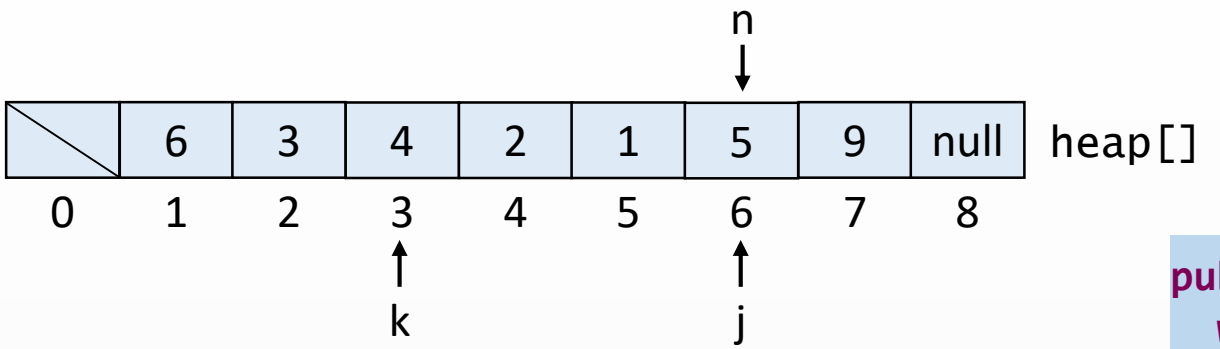
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

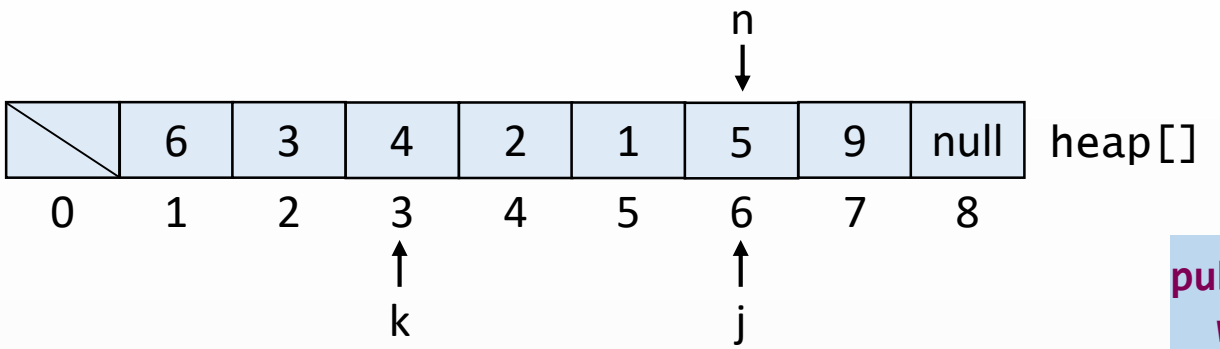
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

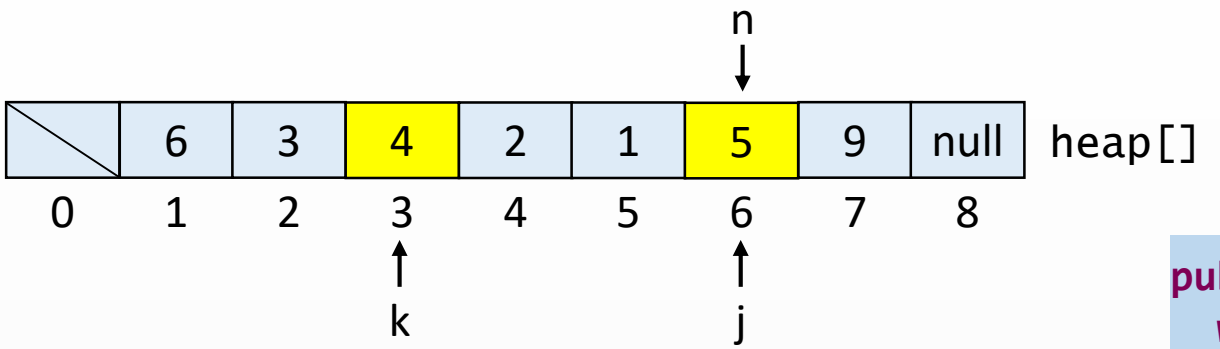
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

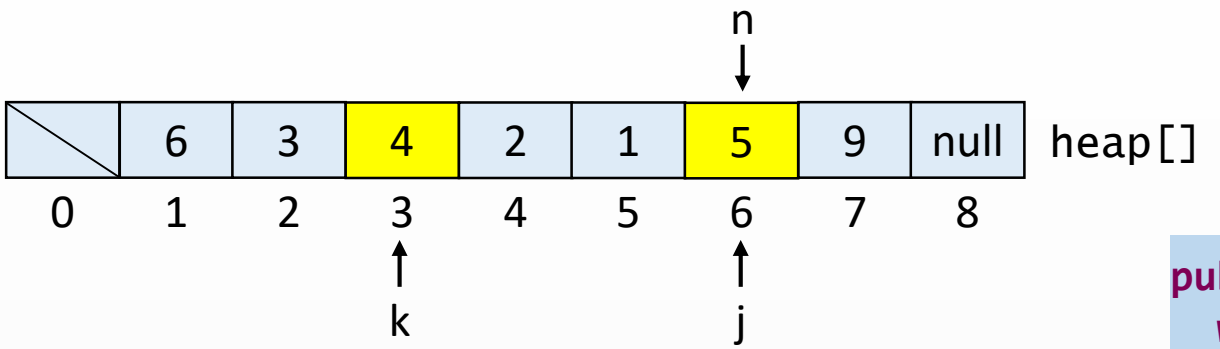
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

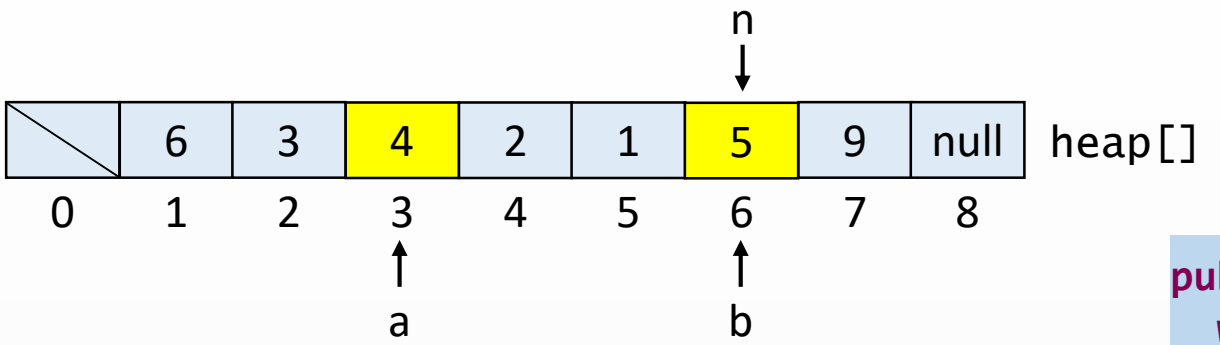
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

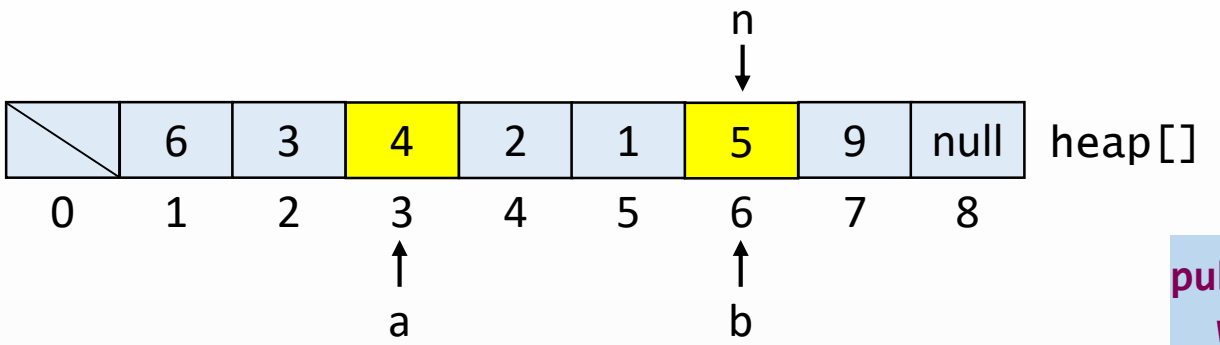
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

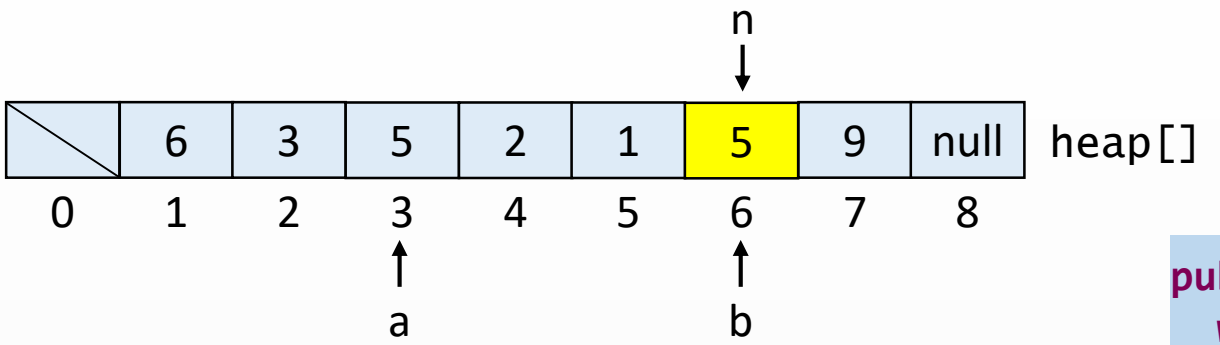
```java
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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




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

```
silMax()
```

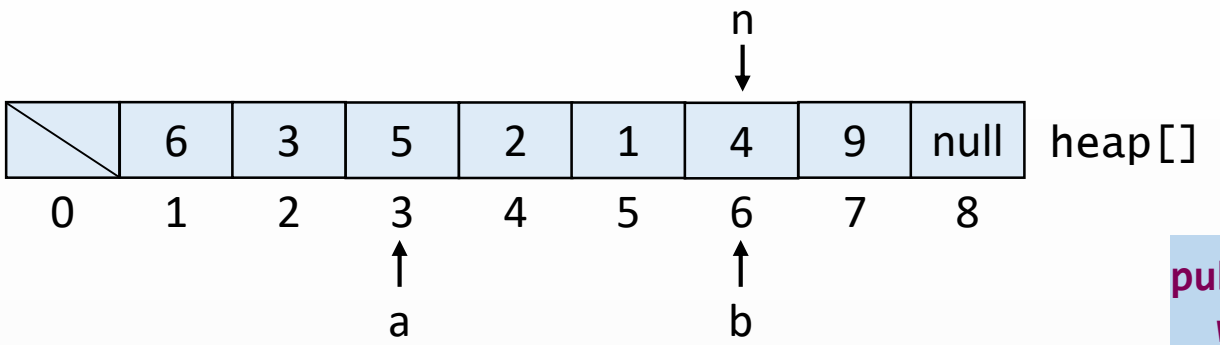
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

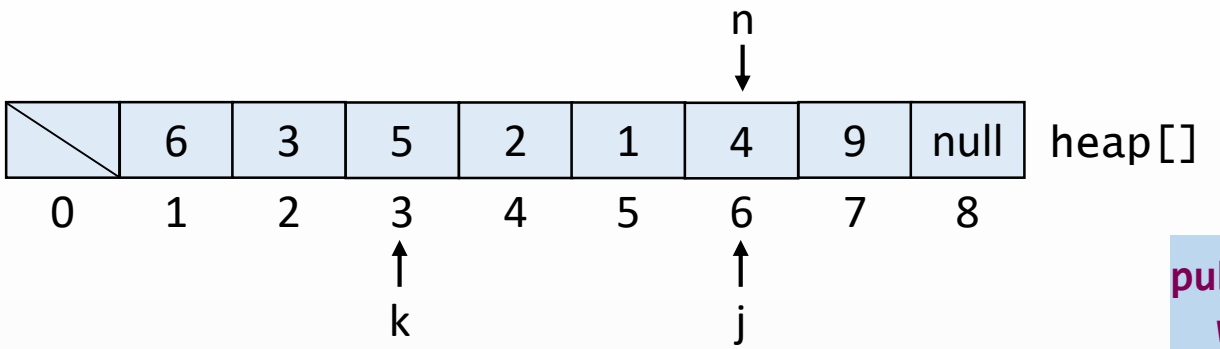
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

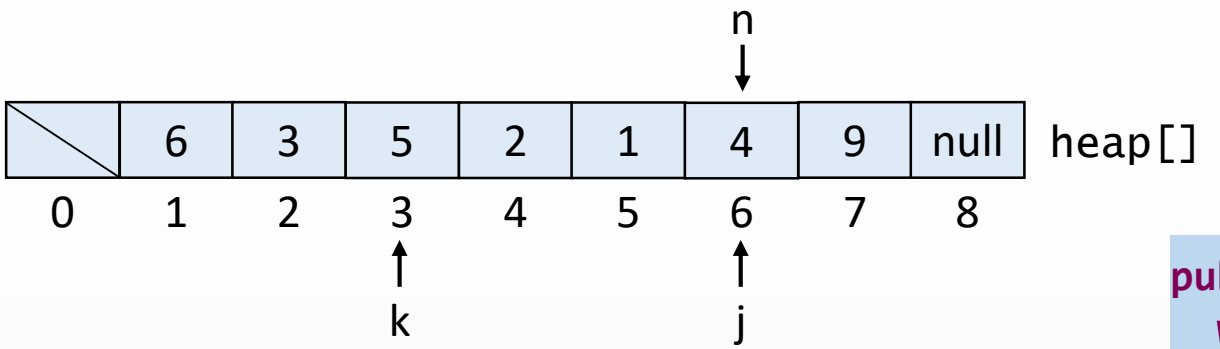
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

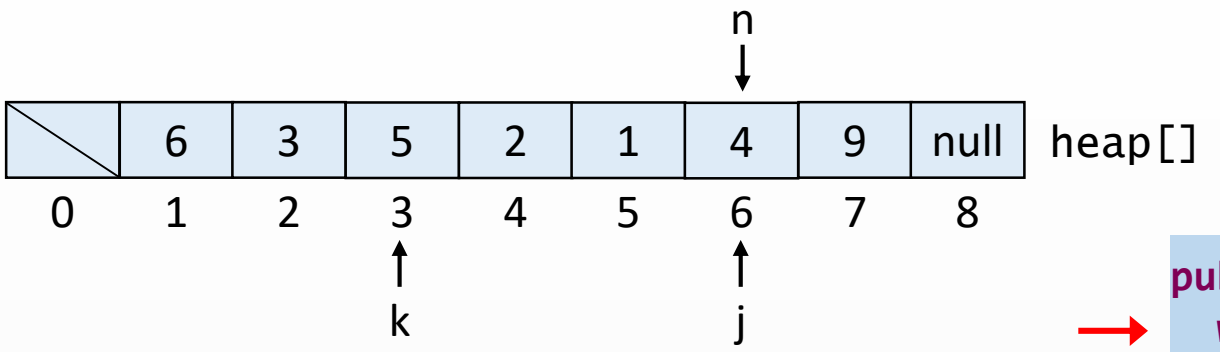
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

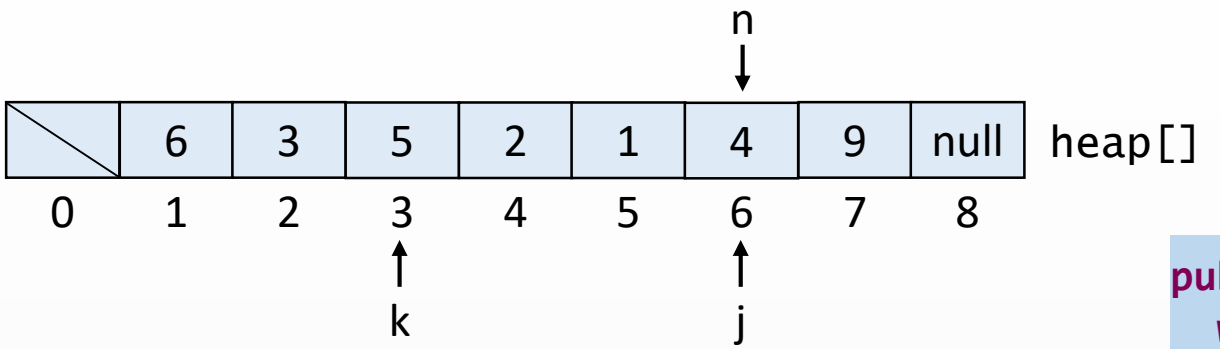
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

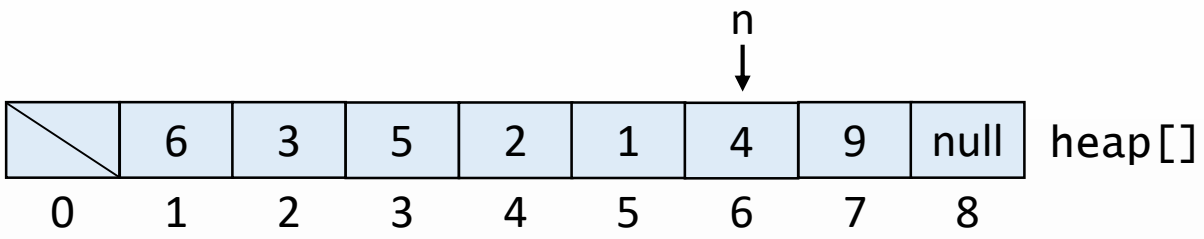
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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()
```

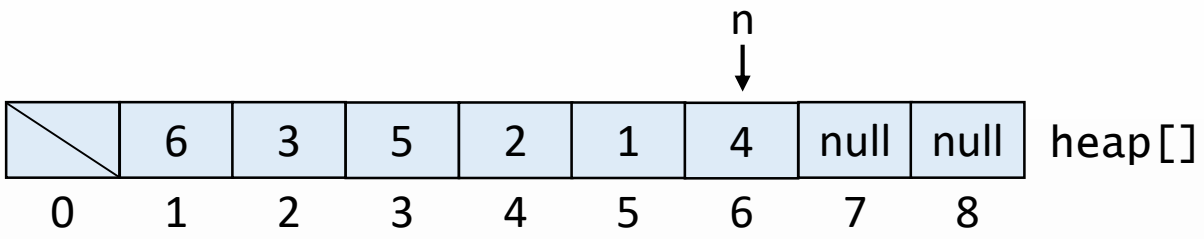
```
public void batir(int k) {
  while(2*k <= n) {
    int j = 2*k;
    if(j < n && heap[j] < heap[j+1]) {
      j++;
    }
    if(heap[k] >= heap[j]) {
      break;
    }
    yerDegistir(k, j);
    k = j;
  }
}

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



```
max = 9
n = 6
```

```
silMax()
```

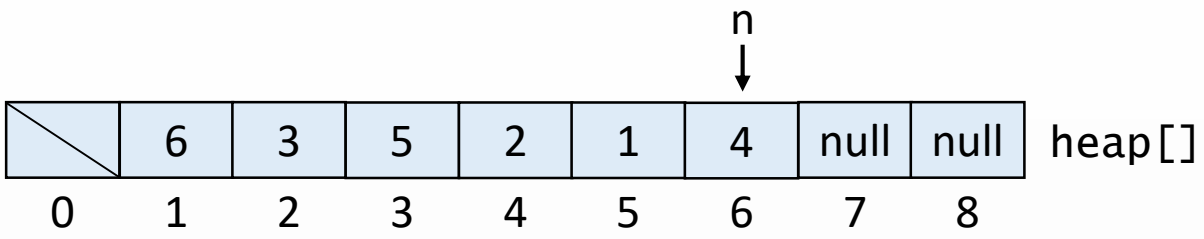
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 6
```

```
silMax()
```

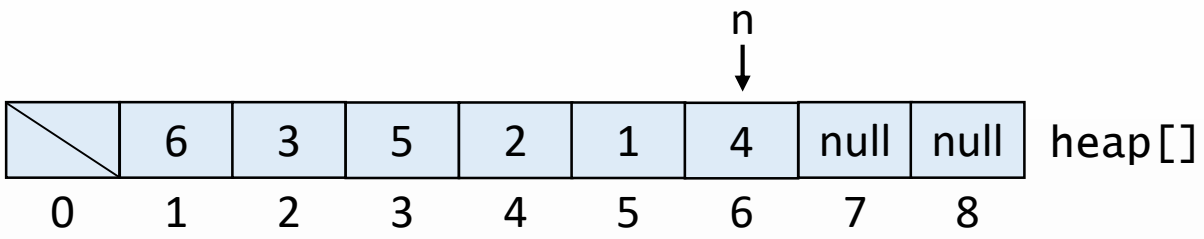
```
public int silMax() {
  int max = heap[1];
  yerDegistir(1,n);
  n--;
  batir(1);
  heap[n + 1] = null;
  if(n > 0 && (n == (heap.length - 1) / 4)) {
    kucult(heap.length / 2);
  }
  return max;
}

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



```
max = 9
n = 6
```

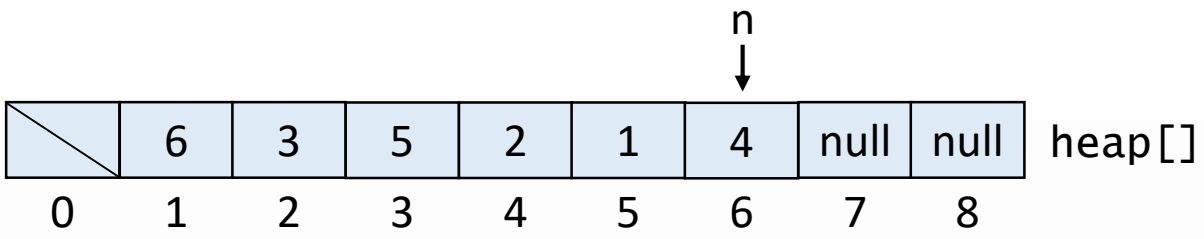
```
silMax()
```

```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;
}
```



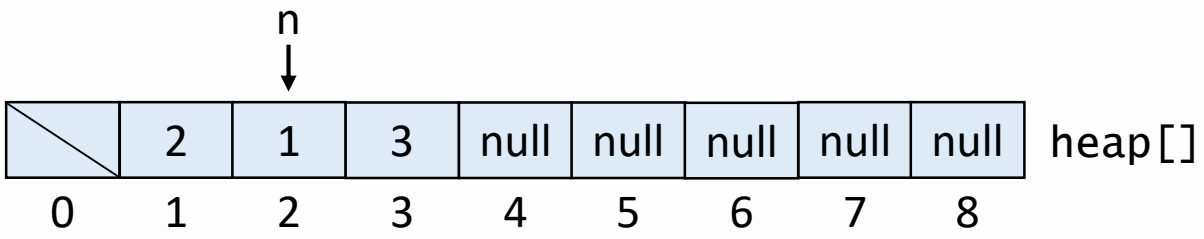
```
max = 9
n = 6

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;
}
```



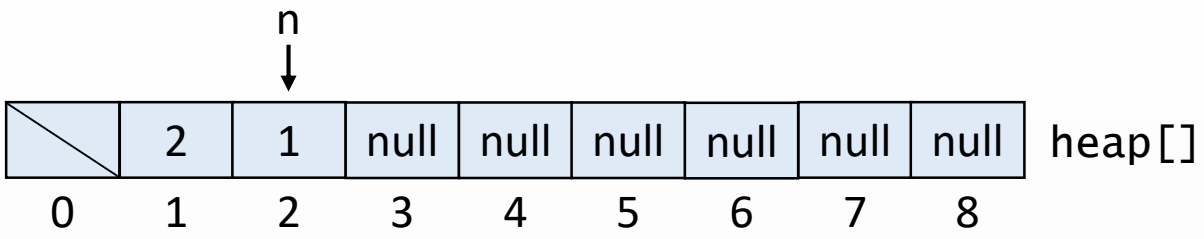
n = 6

```
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;
}
```



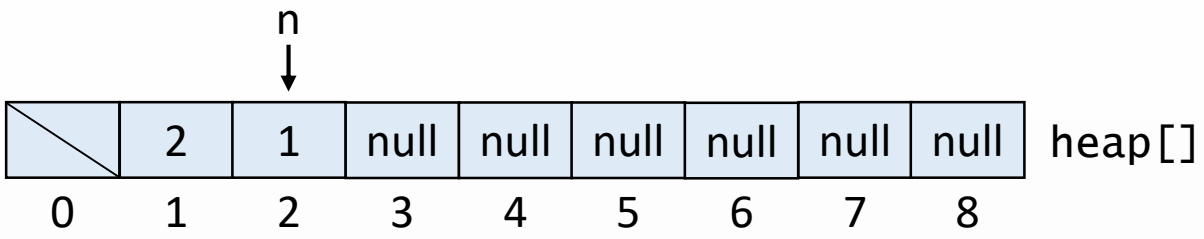
```
max = 3
n = 2
```

```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;
}
```



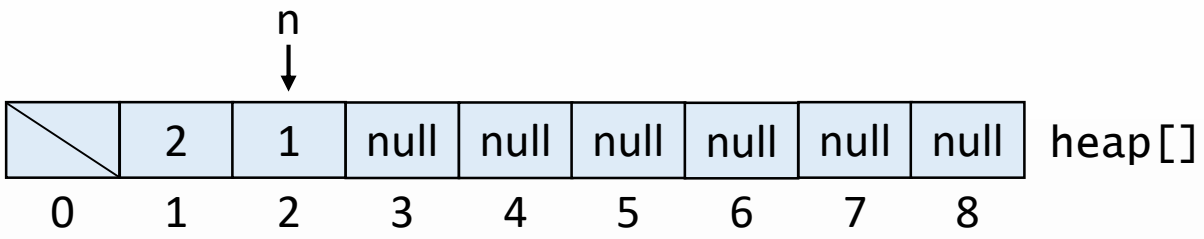
```
max = 3
n = 2
```

```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;
}
```



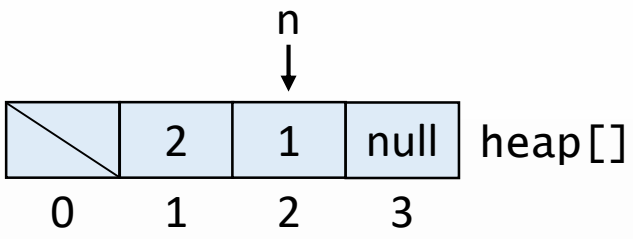
```
max = 3
n = 2
```

```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;
}
```



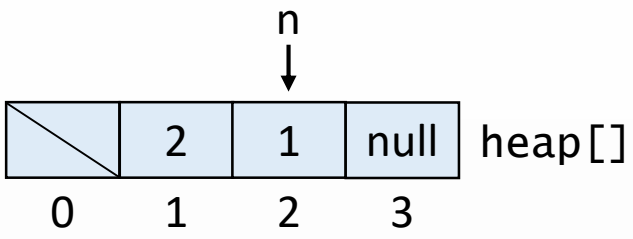
max = 3
n = 2

```
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;
}
```



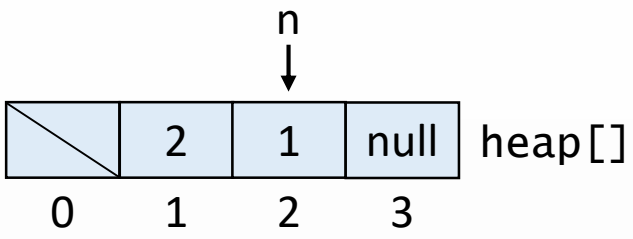
max = 3
n = 2

```
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;
}
```



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