Relembrando outros algoritmos

- Insertion sort (único desses *in loco*)
  - Operações da ordem de n*(n-1)/2
- Merge sort (merge *in loco* com mais operações)
  - Operações da ordem de $n*\log_2 n$
- Count sort (exige vetor de contadores)
  - Operações da ordem de n + "intervalo de valores"
- Radix sort (exige listas)
  - Operações da ordem de $n * \log_{10} n$

Heapsort

- Ordenação baseada na estrutura de dados *heap*
  - A *heap* é um vetor que pode ser visto como uma árvore binária completa
  - Cada nó da árvore equivale a um valor no vetor

Primitivas da Heap

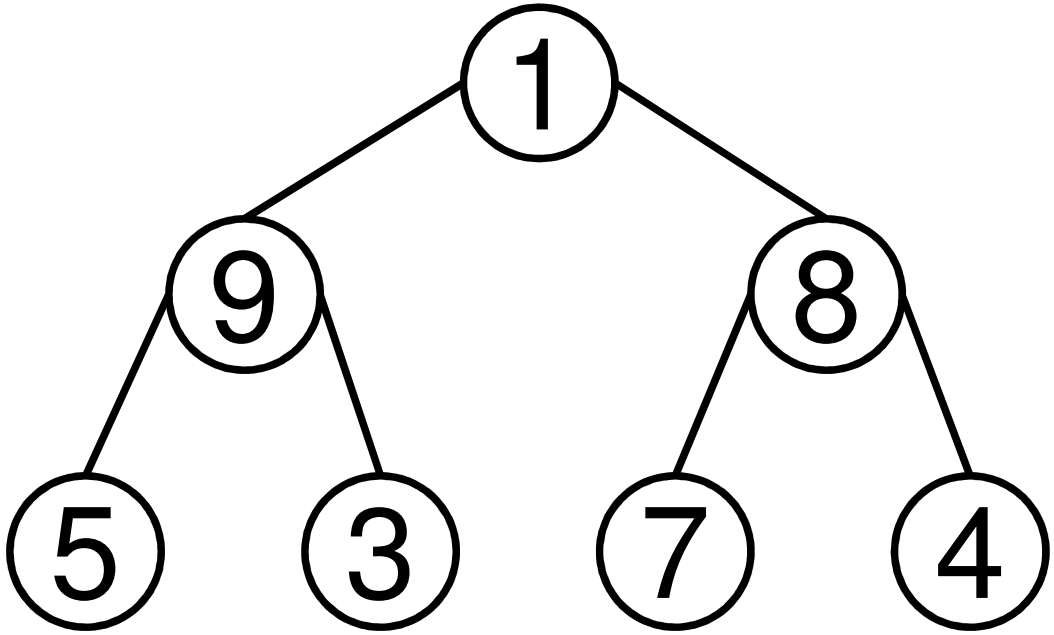
- length(V)
  - número de elementos no vetor V
- heap-size(V)
  - número de elementos da *heap* armazenados no vetor V

Vetor Heap – primitivas de árvore

- parent(i)
  - return i/2
- left(i)
  - return 2i
- right(i)
  - return 2i+1

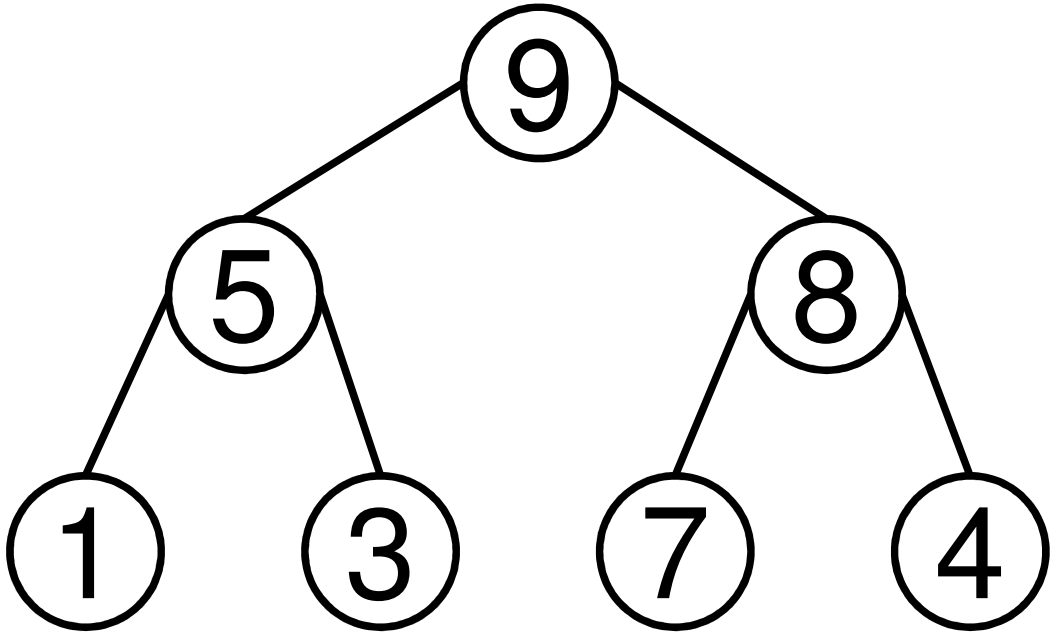
Heap como um vetor

1	9	8	5	3	7	4



Propriedade da Heap

- Para cada nó i que não seja a raiz, então:
  - V[parent(i)] >= V[i]
- Exemplo:



Primitiva Heapify

- Função para manter a propriedade da heap

```
function heapify(V, i)

   l = left(i)

   r = right(i)

   if l<=heap-size(V) and V[l]>V[i] then largest = l
     else largest = i end

   if r<=heap-size(V) and V[r]>V[largest]
     then largest=r end

   if largest != i then
        V[i], V[largest] = V[largest], V[i]
        heapify(V, largest)
     end
  end
```

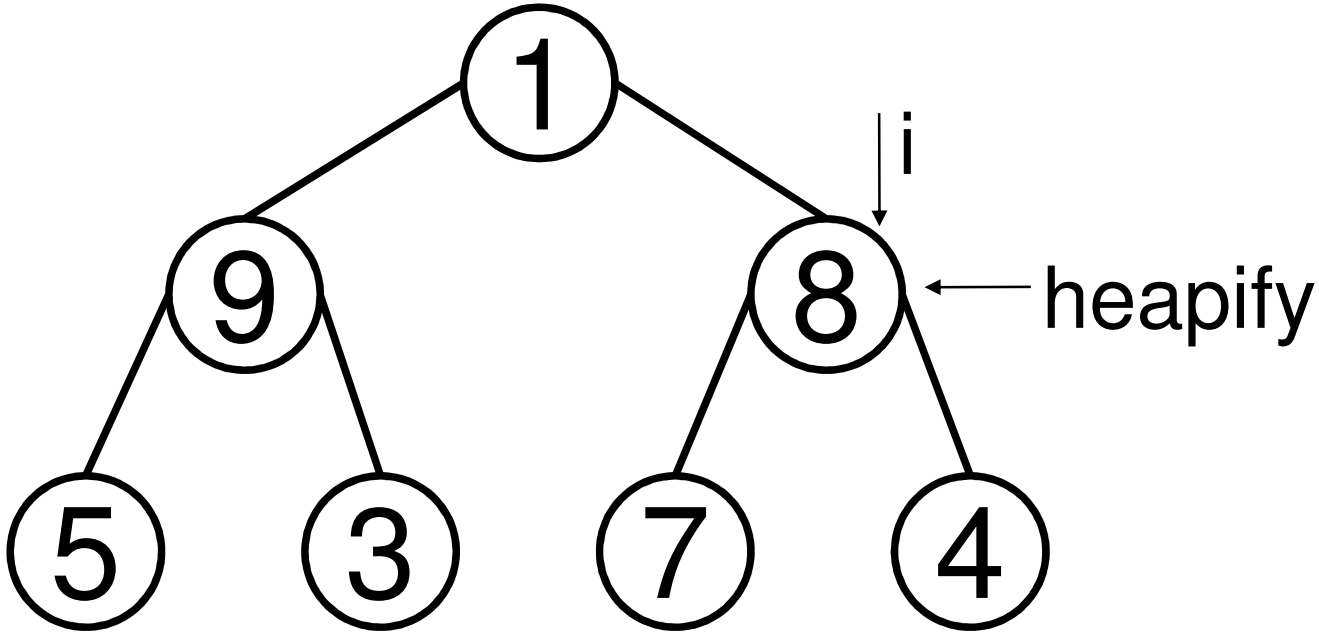
Construindo uma Heap

- Primitiva que constrói uma *heap* a partir de um vetor de N elementos

```
function build_heap(V)
    heap-size(V) = length(V) -- #V
    for i=length(V)/2,1,-1 do
        heapify(V,i)
    end
end
```

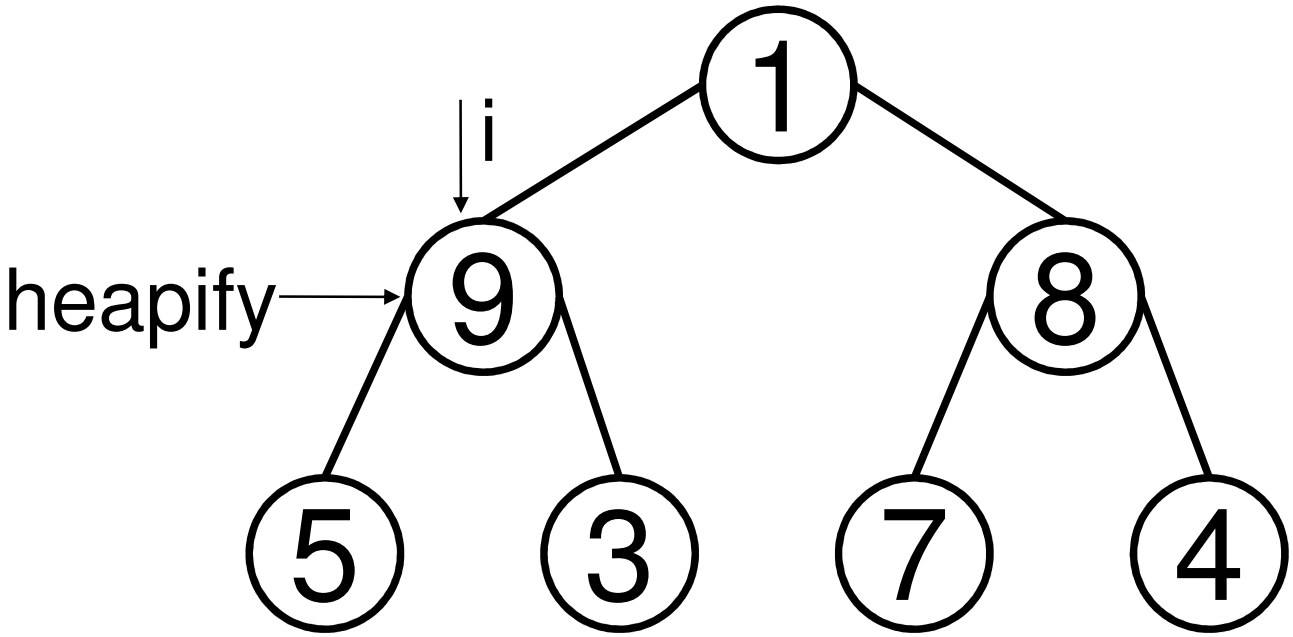
Construindo uma Heap

1	9	8	5	3	7	4



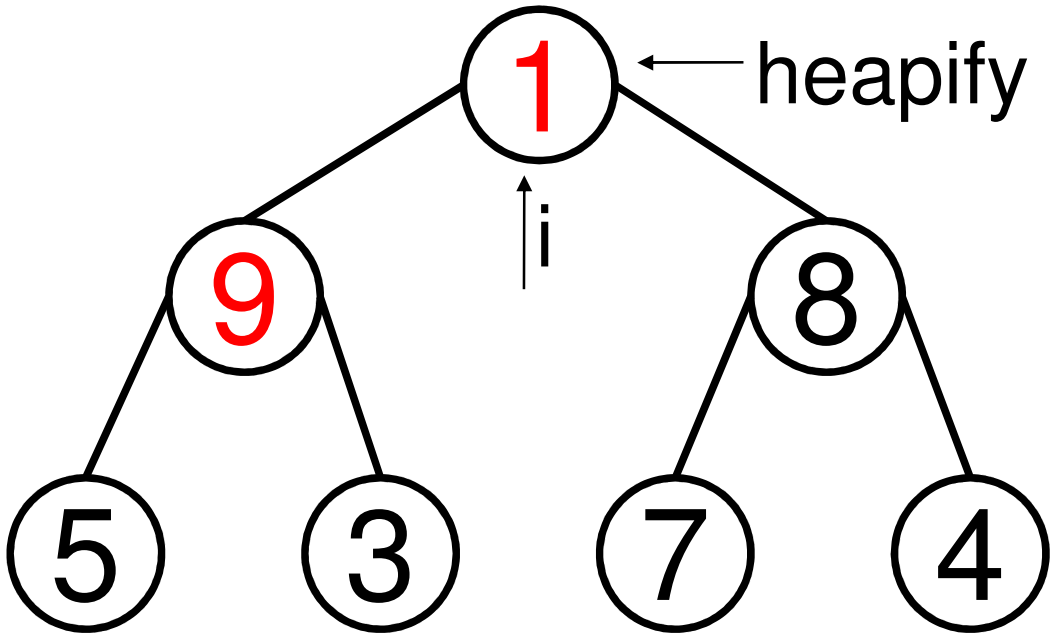
Construindo uma Heap

1	9	8	5	3	7	4



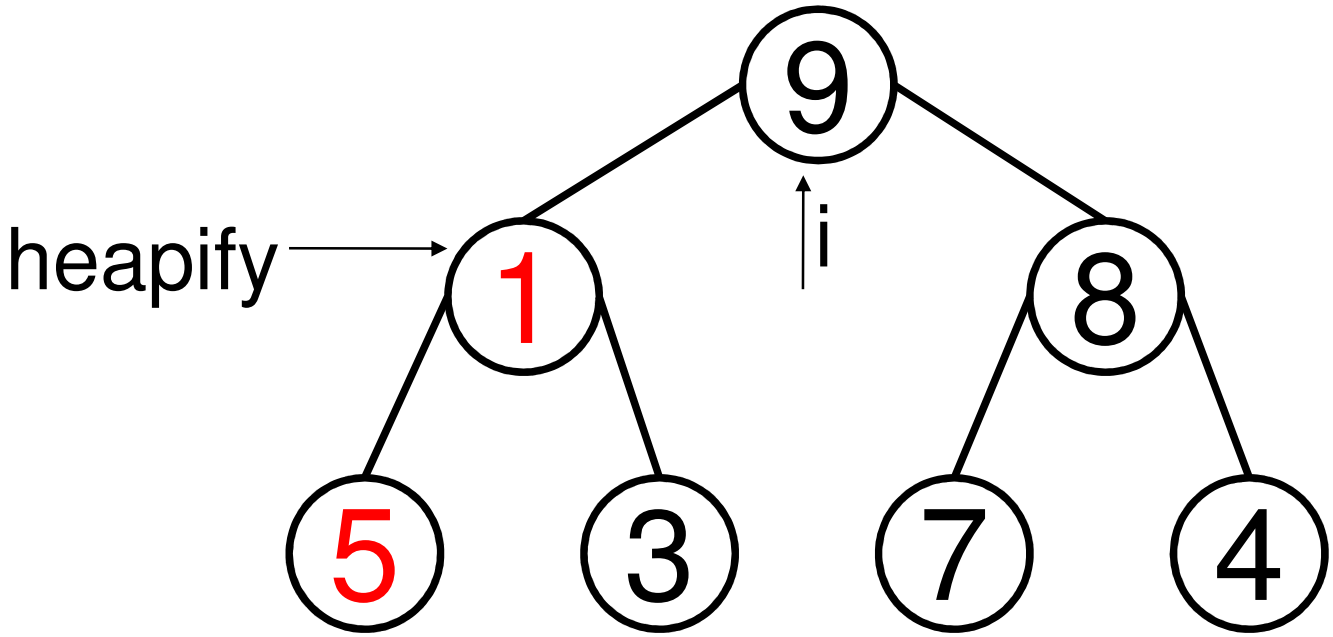
Construindo uma Heap

1	9	8	5	3	7	4



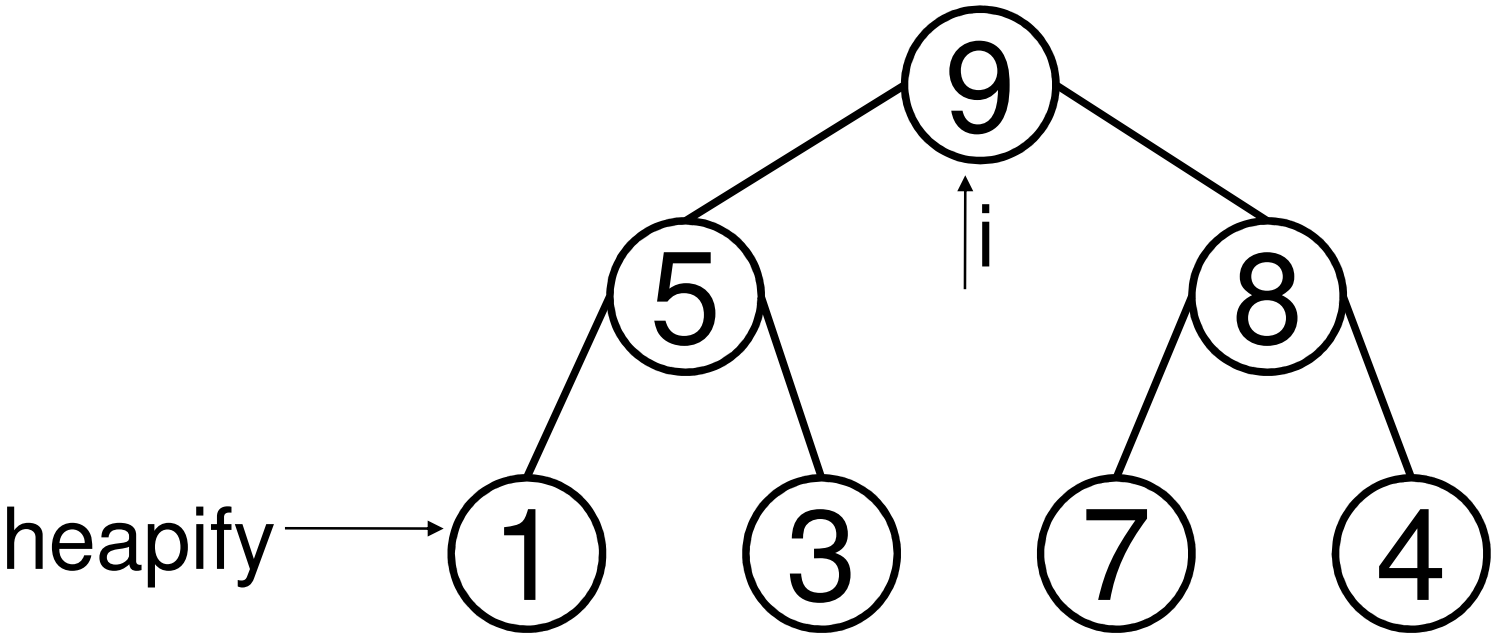
Construindo uma Heap

9	1	8	5	3	7	4



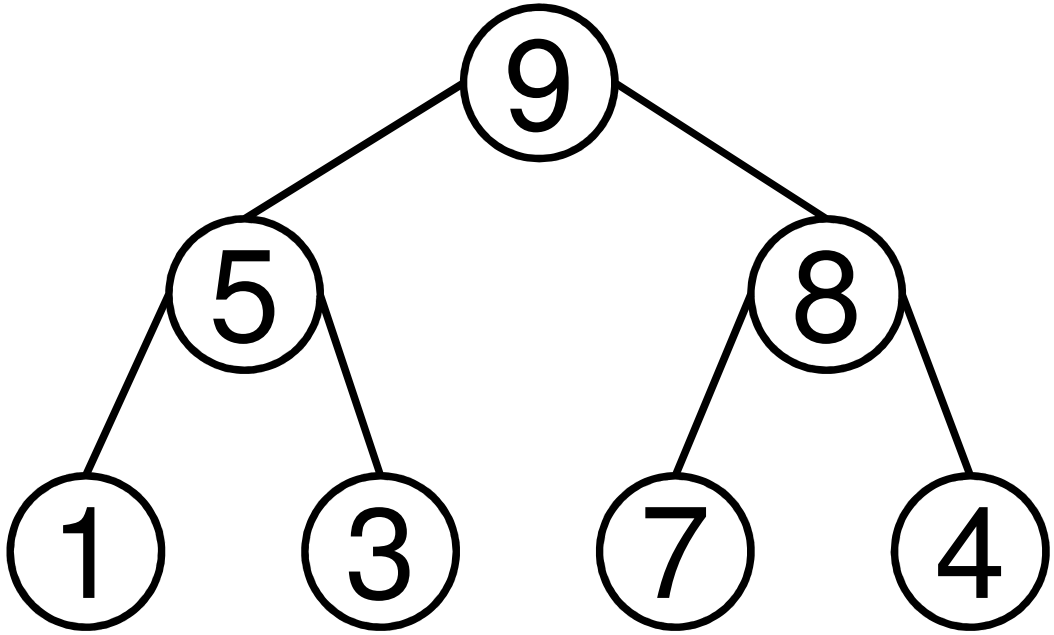
Construindo uma Heap

9	5	8	1	3	7	4



Construindo uma Heap

9	5	8	1	3	7	4

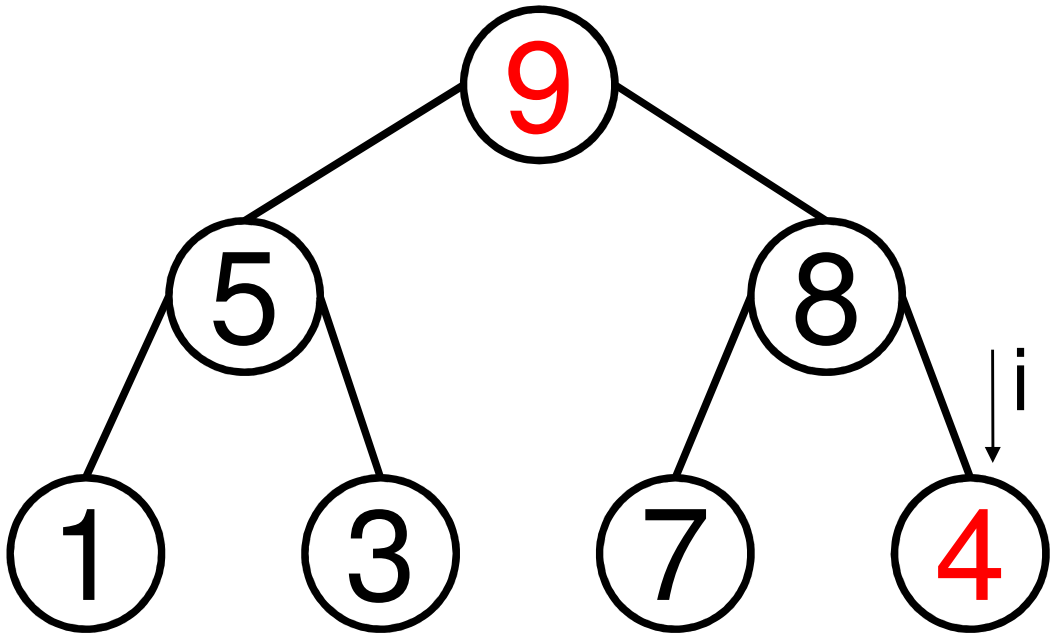


Heapsort

```
function heapsort(V)
   for i=length(V),2,-1 do
         V[1], V[i] = V[i], V[1]
         heap-size(V) = heap-size(V)-
 1
         heapify(V,1)
      end
  end
```

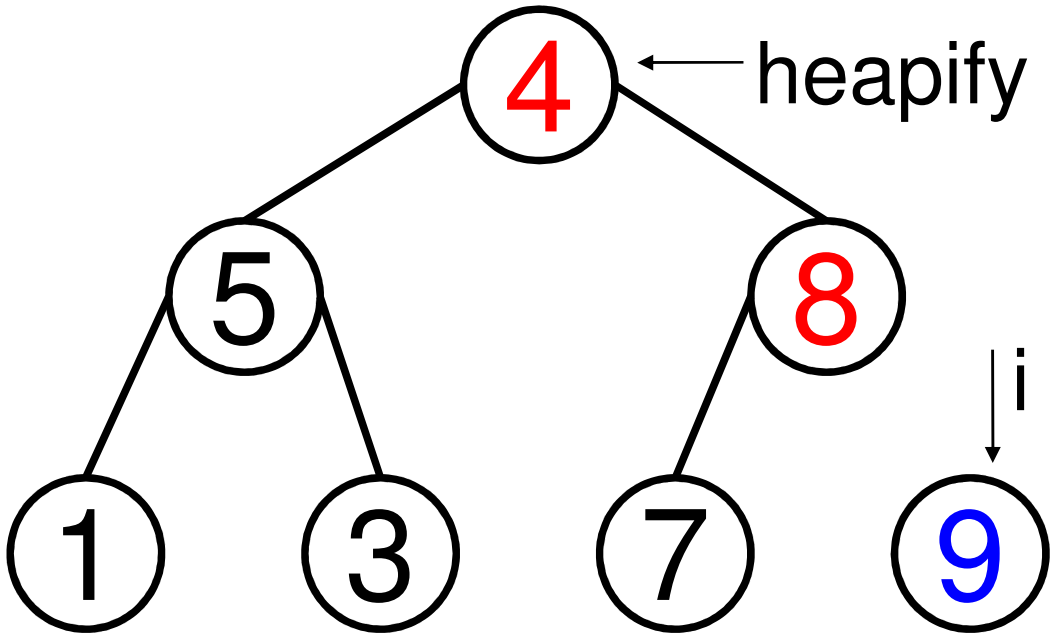
Heapsort

9	5	8	1	3	7	4



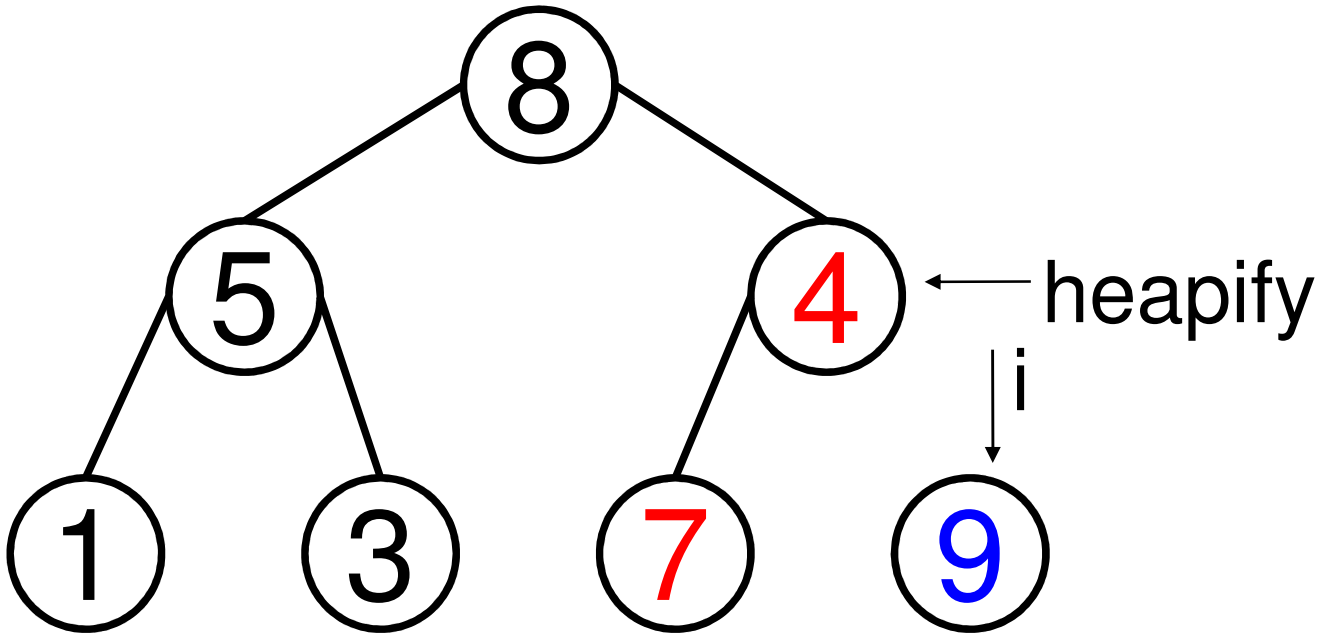
Heapsort

4	5	8	1	3	7	9



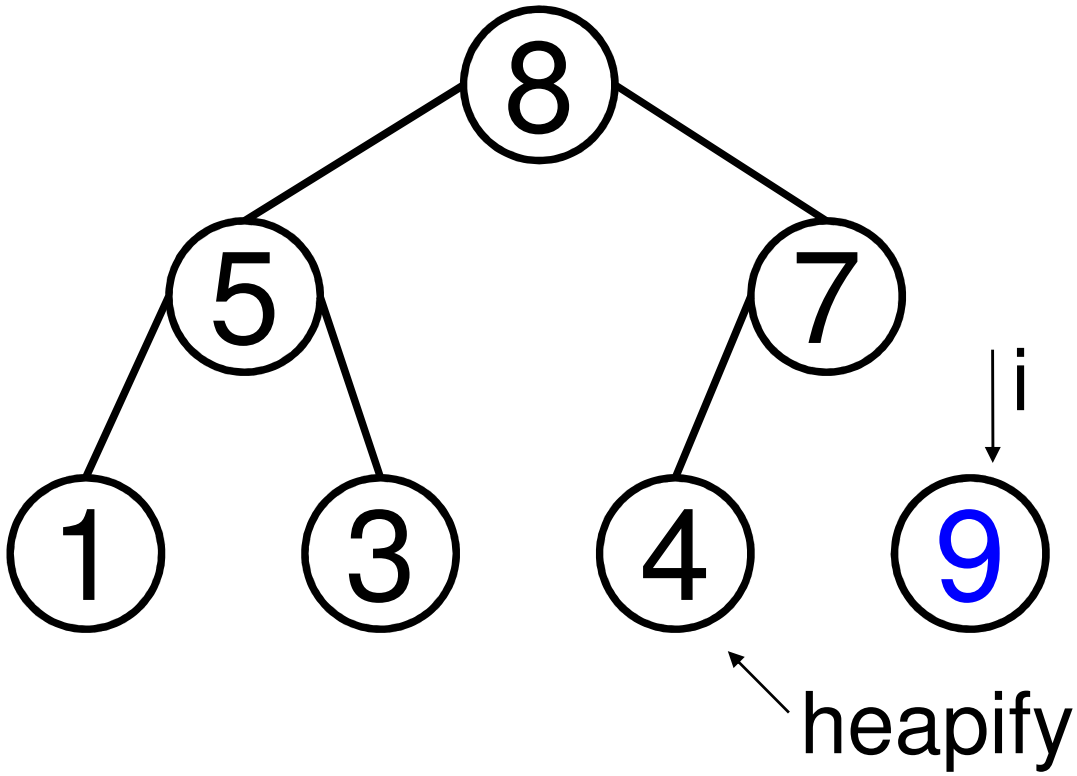
Heapsort

8	5	4	1	3	7	9



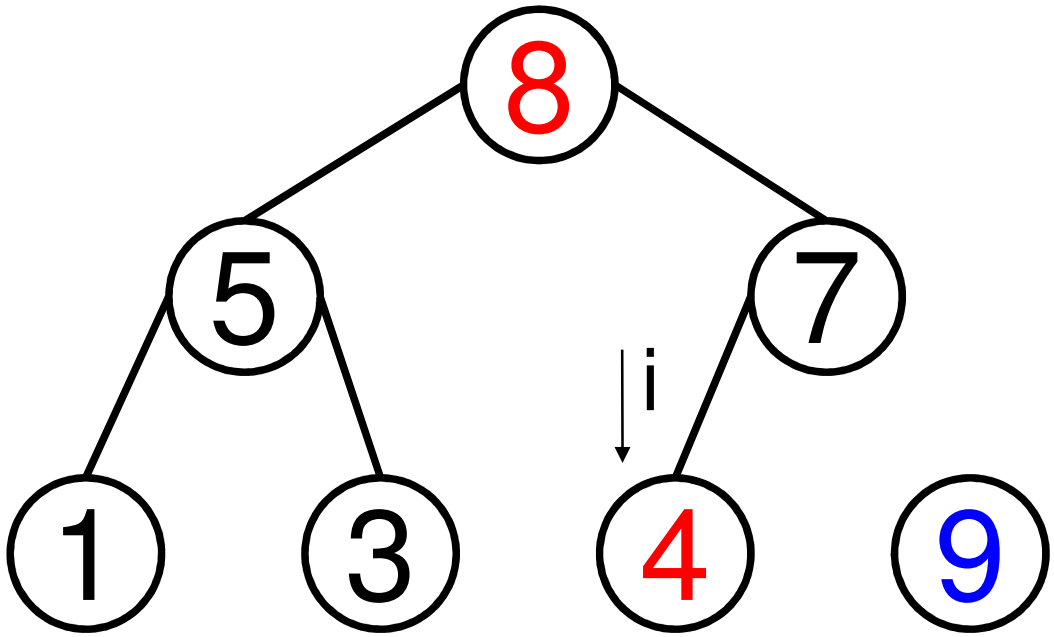
Heapsort

8	5	7	1	3	4	9



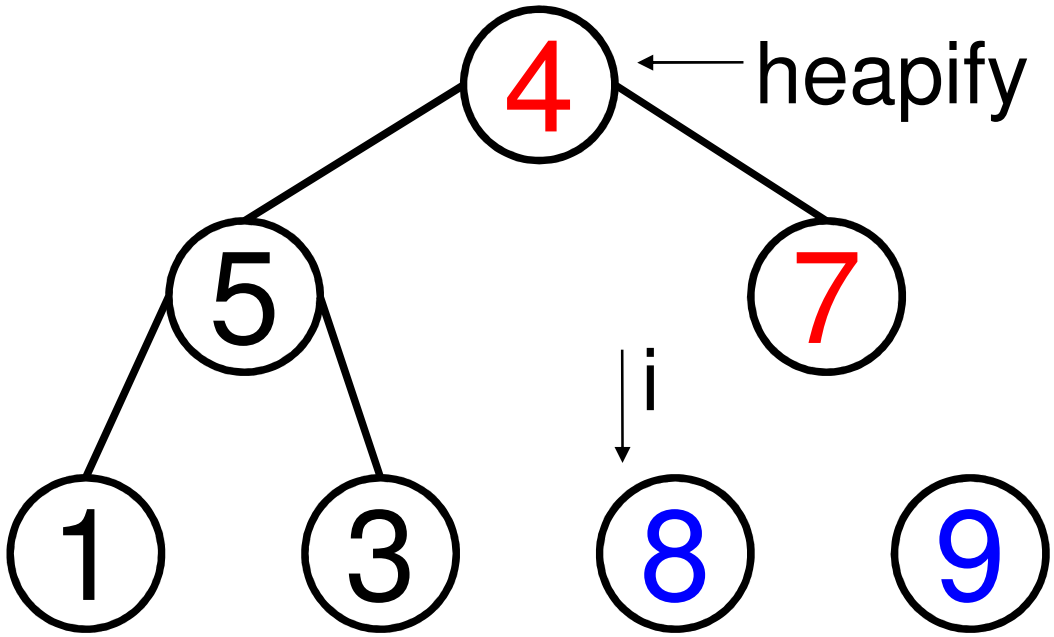
Heapsort

8	5	7	1	3	4	9



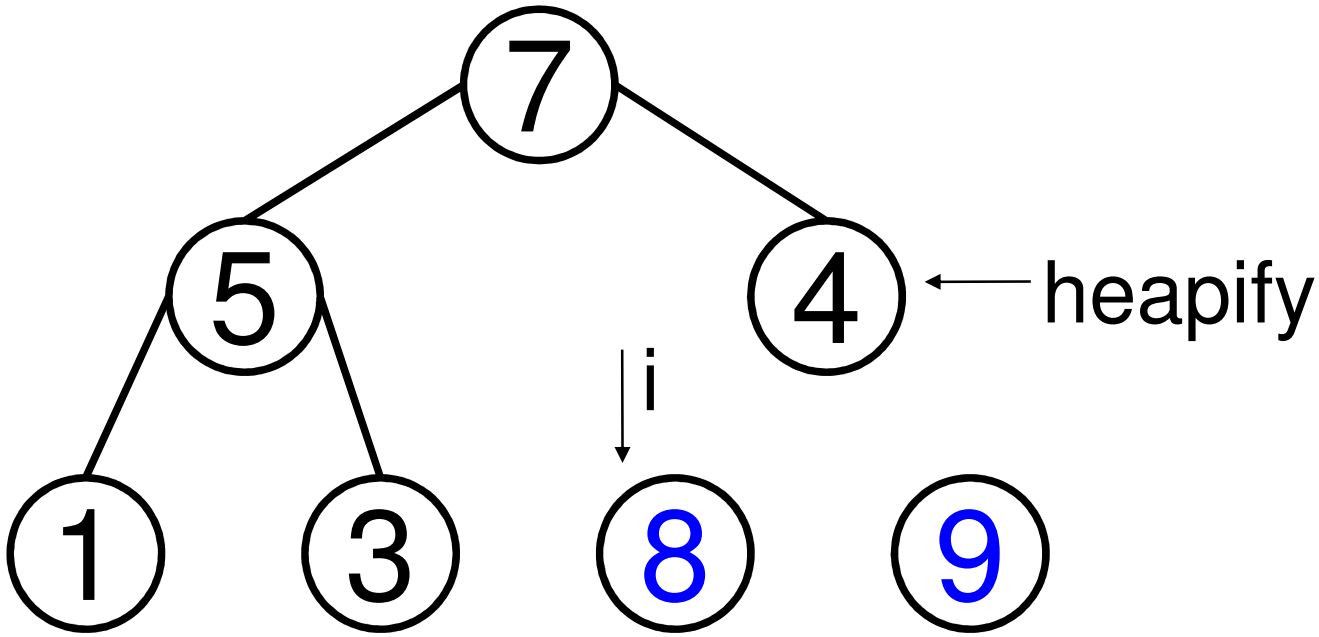
Heapsort

4	5	7	1	3	8	9



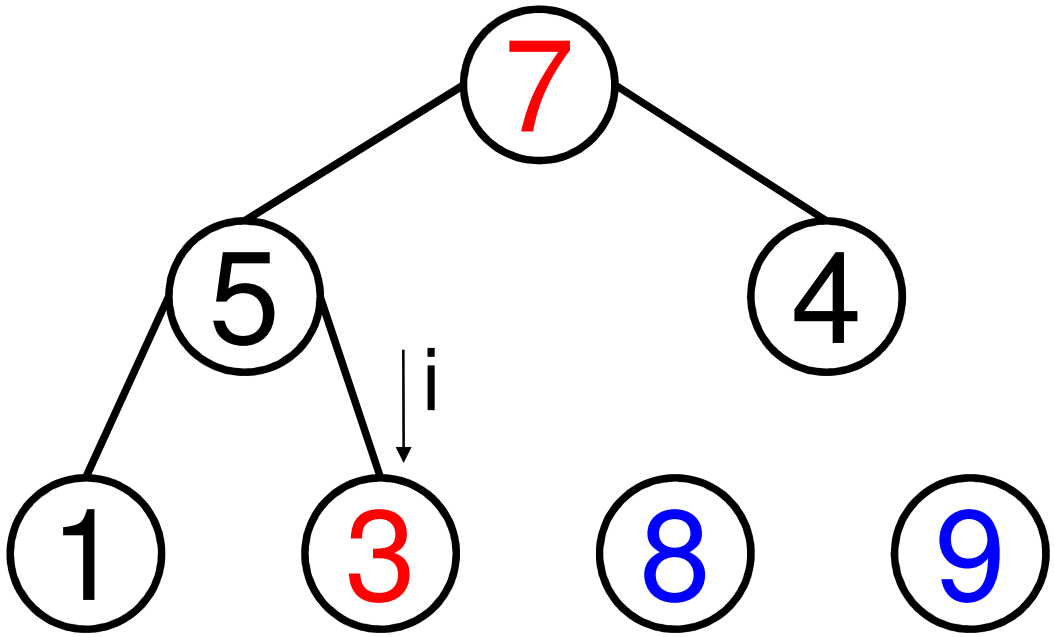
Heapsort

7	5	4	1	3	8	9



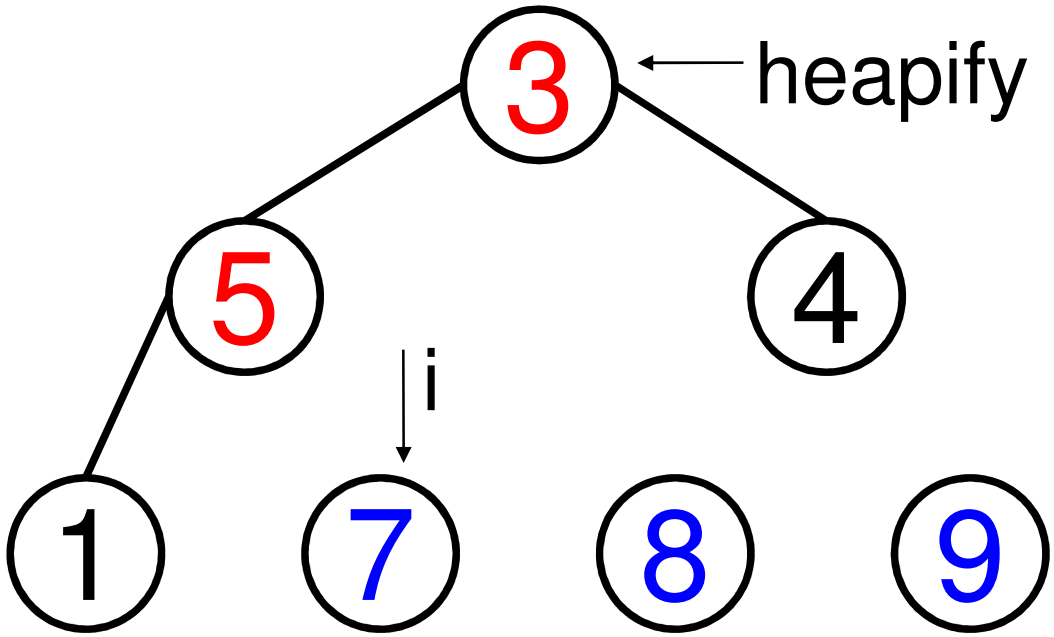
Heapsort

7	5	4	1	3	8	9



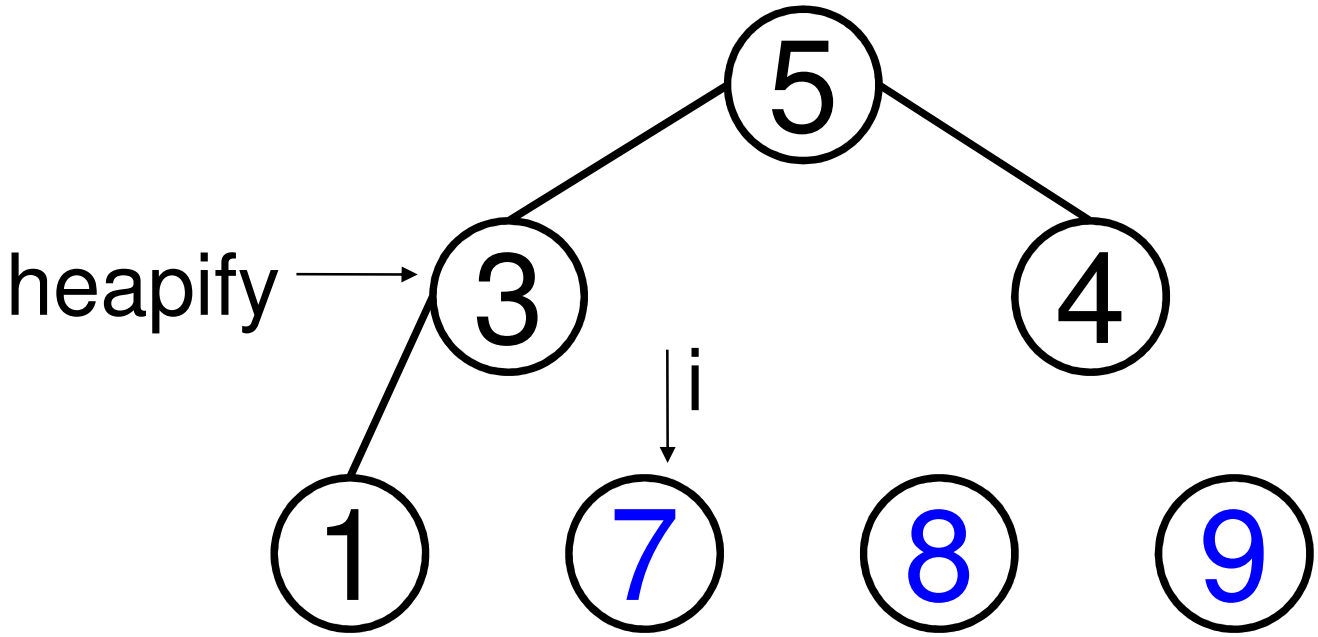
Heapsort

3	5	4	1	7	8	9



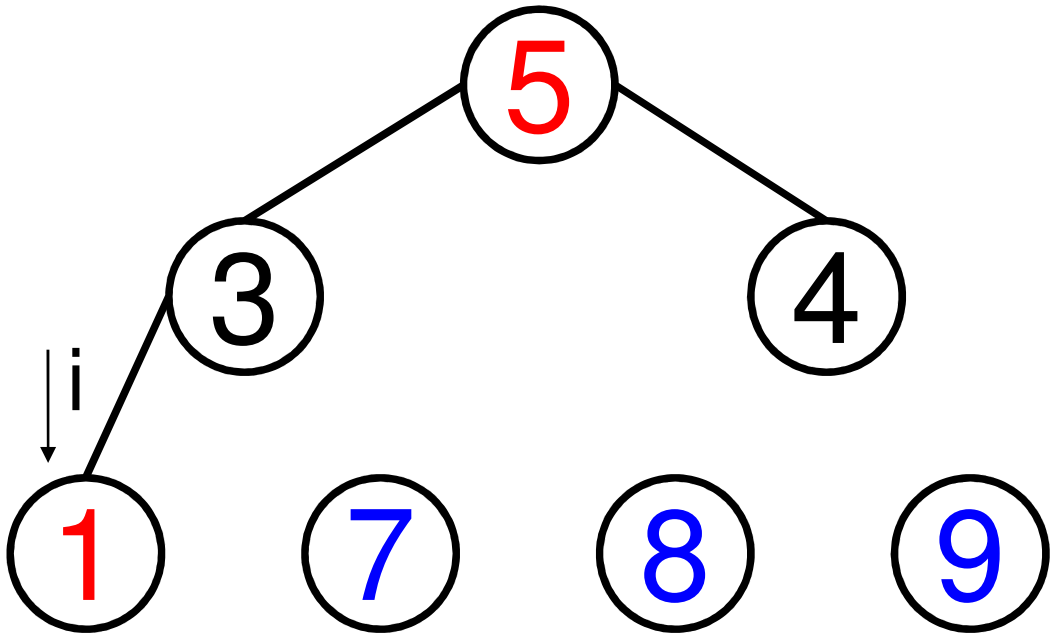
Heapsort

5	3	4	1	7	8	9



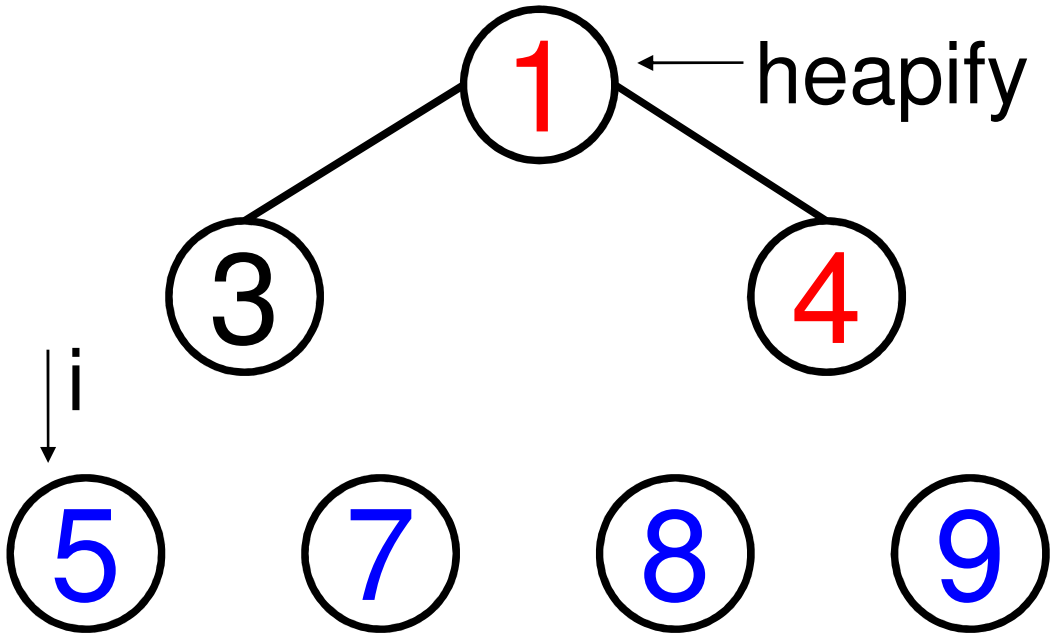
Heapsort

5	3	4	1	7	8	9



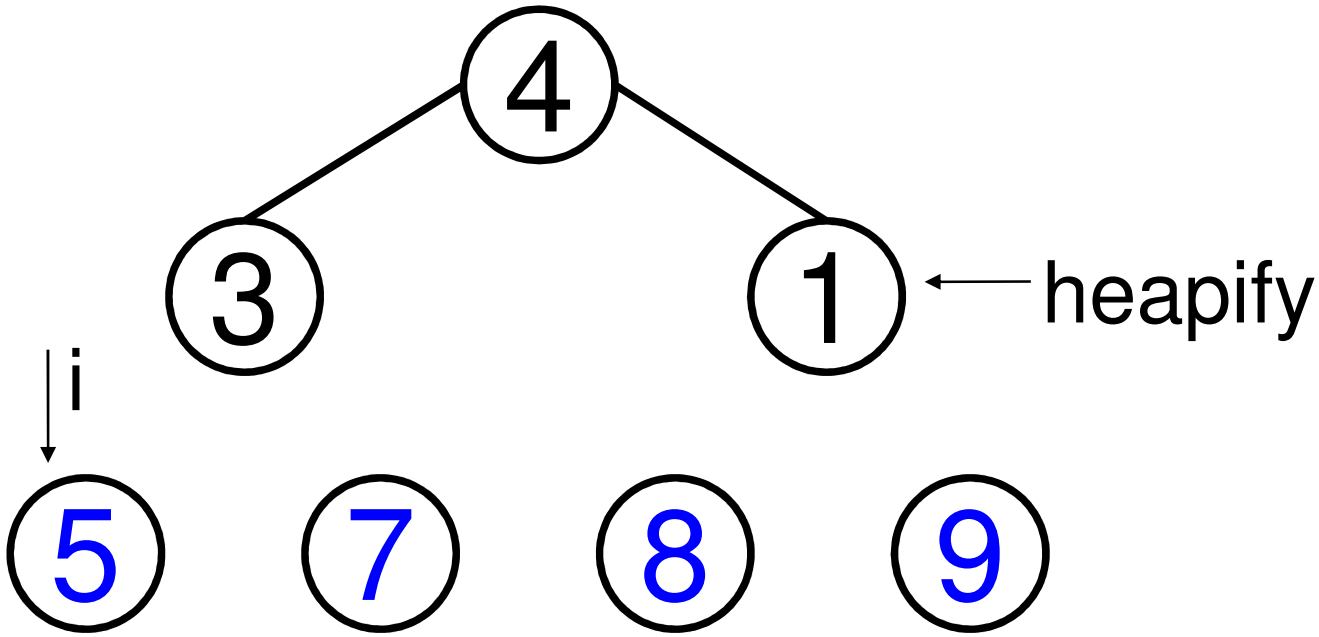
Heapsort

1	3	4	5	7	8	9



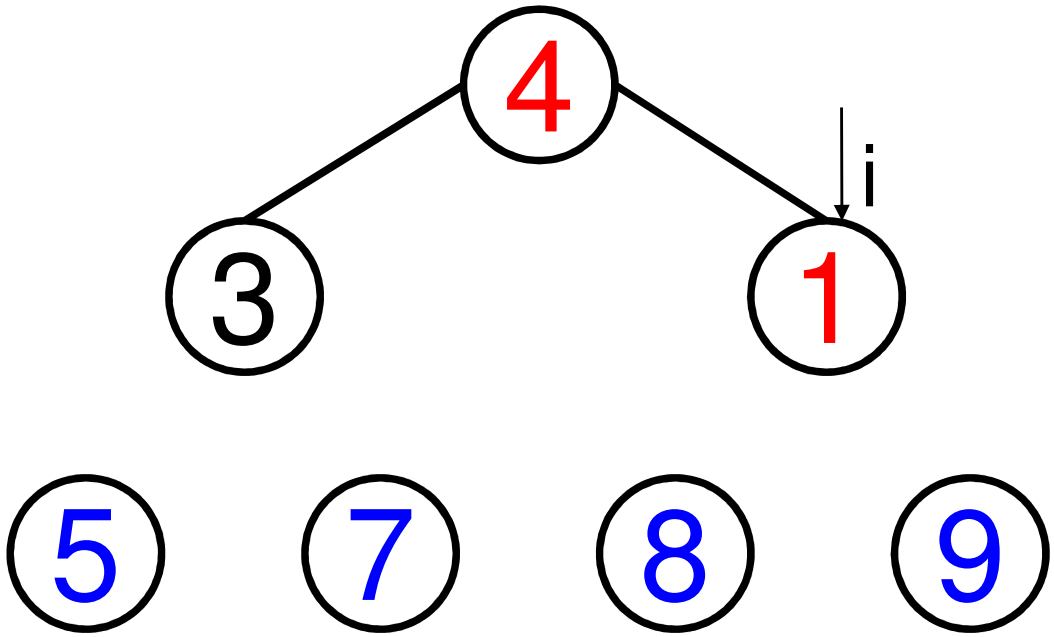
Heapsort

4	3	1	5	7	8	9



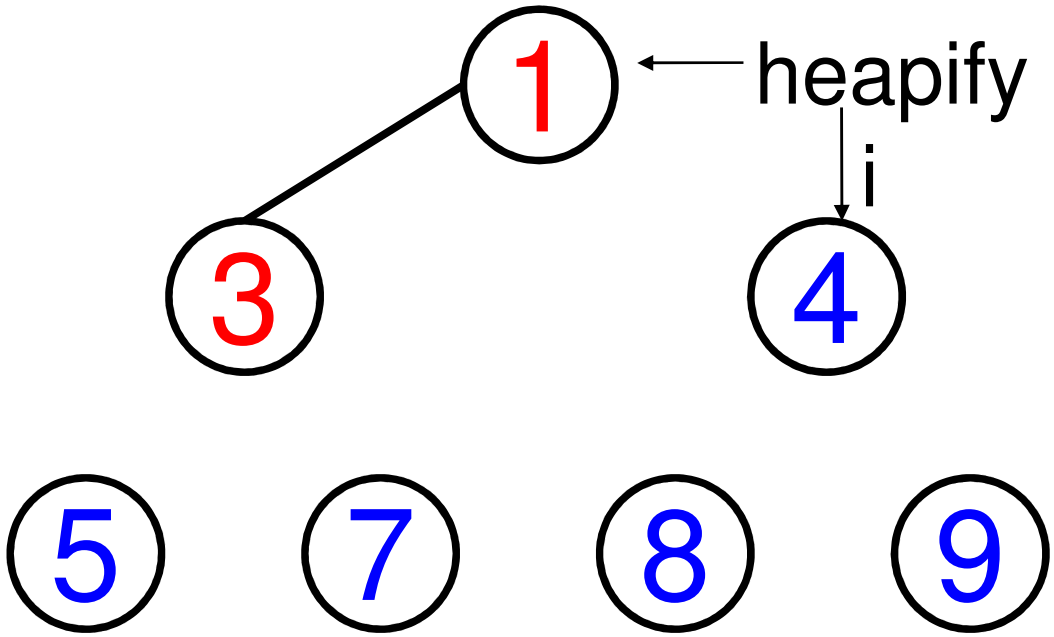
Heapsort

4	3	1	5	7	8	9



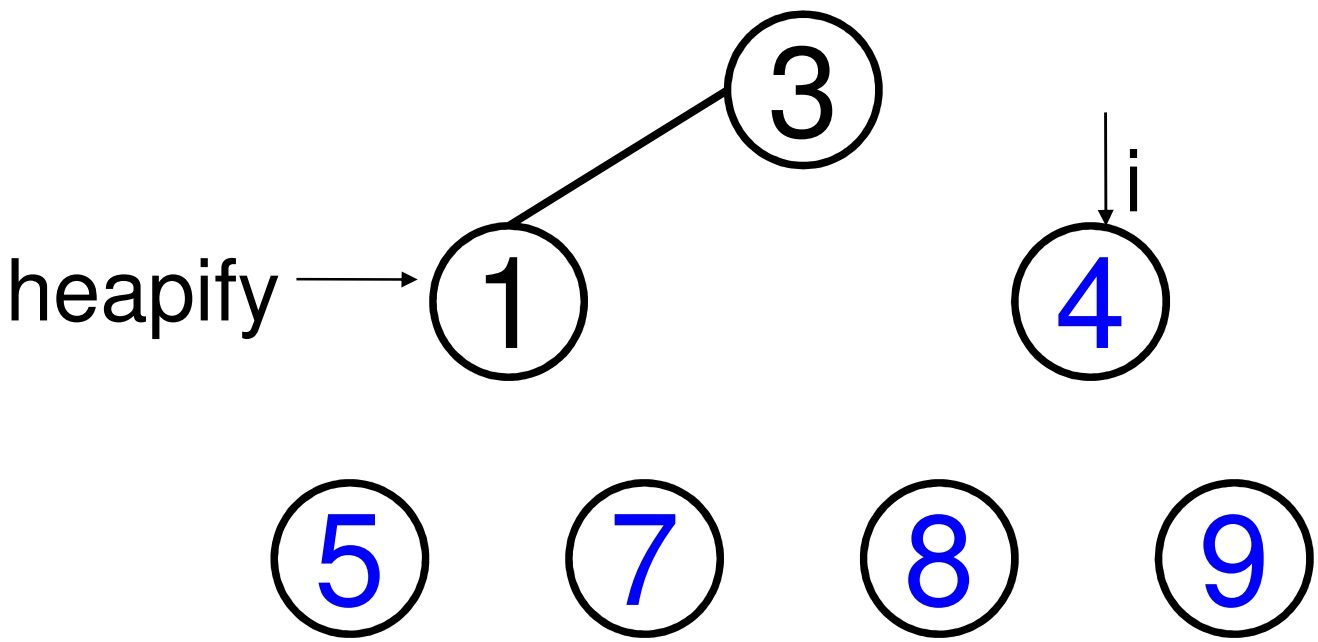
Heapsort

1	3	4	5	7	8	9



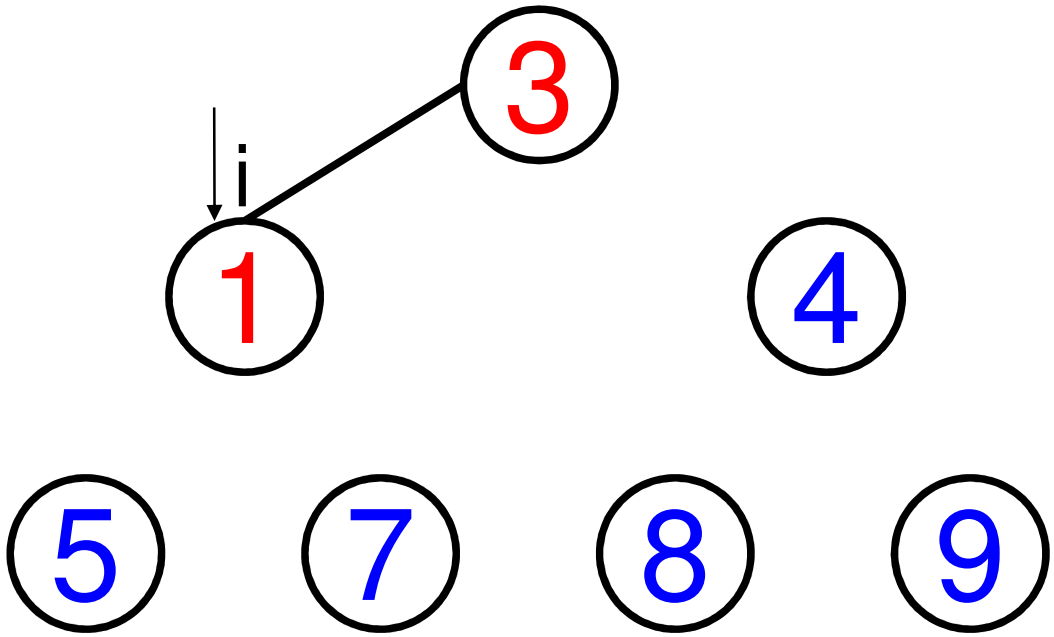
Heapsort

3	1	4	5	7	8	9



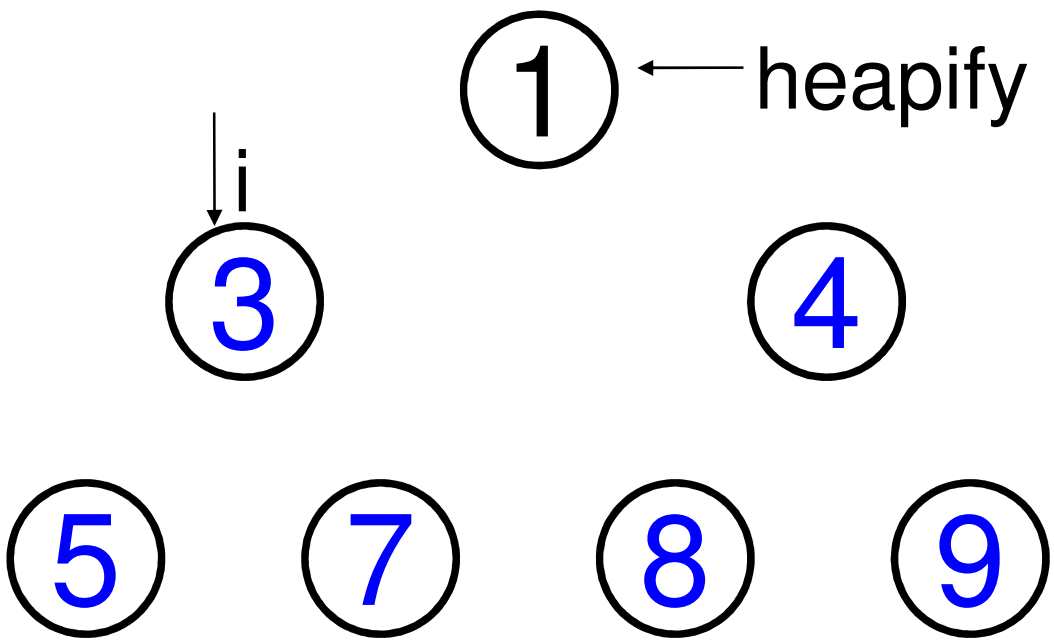
Heapsort

3	1	4	5	7	8	9



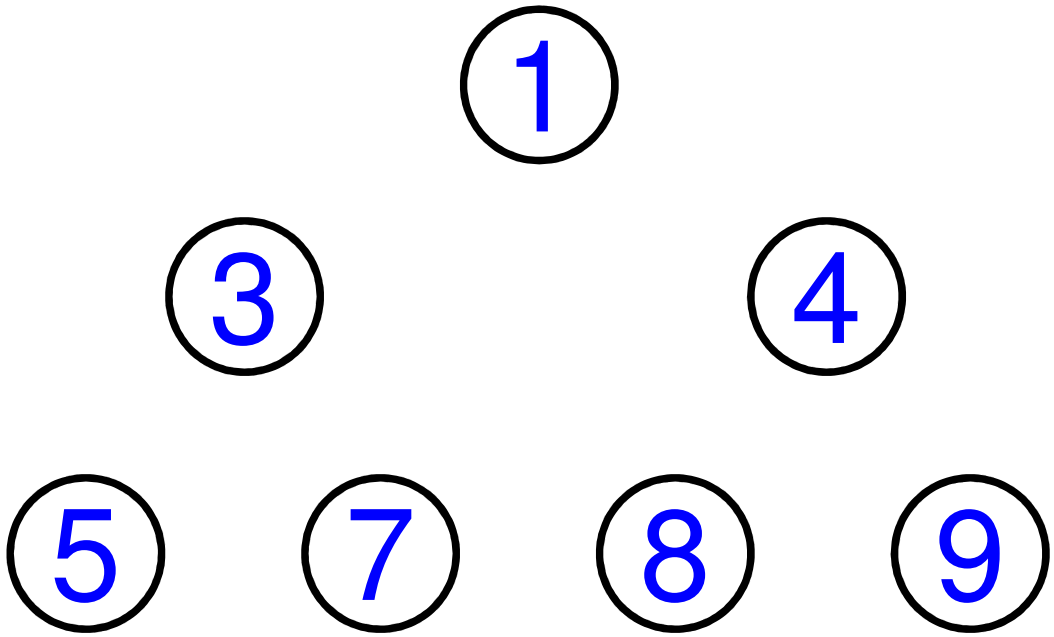
Heapsort

1	3	4	5	7	8	9

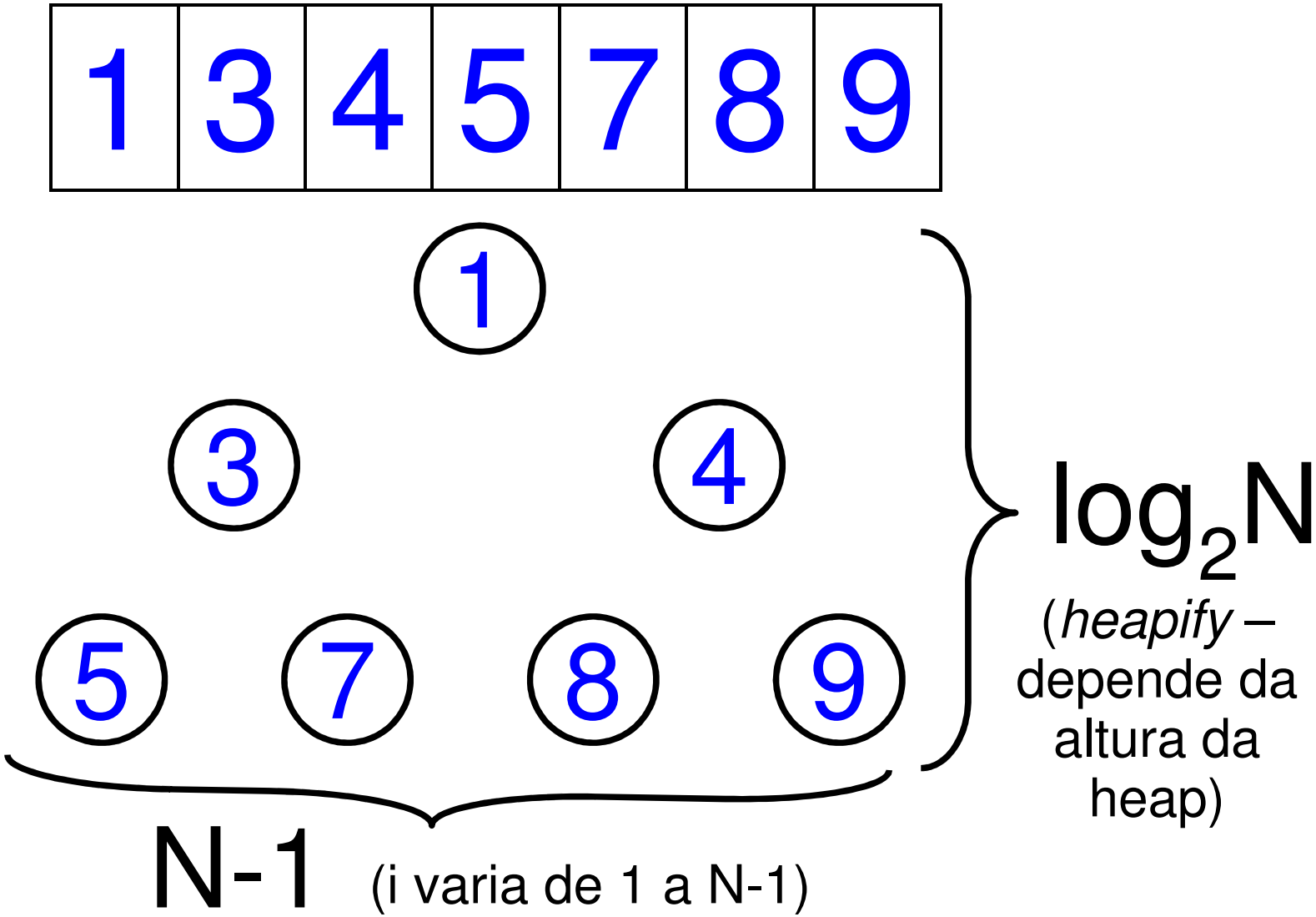


Heapsort

1	3	4	5	7	8	9



Operações do Heapsort



Operações do Heapsort

1	3	4	5	7	8	9

Total de operações:
$(N-1) * \log_2 N$