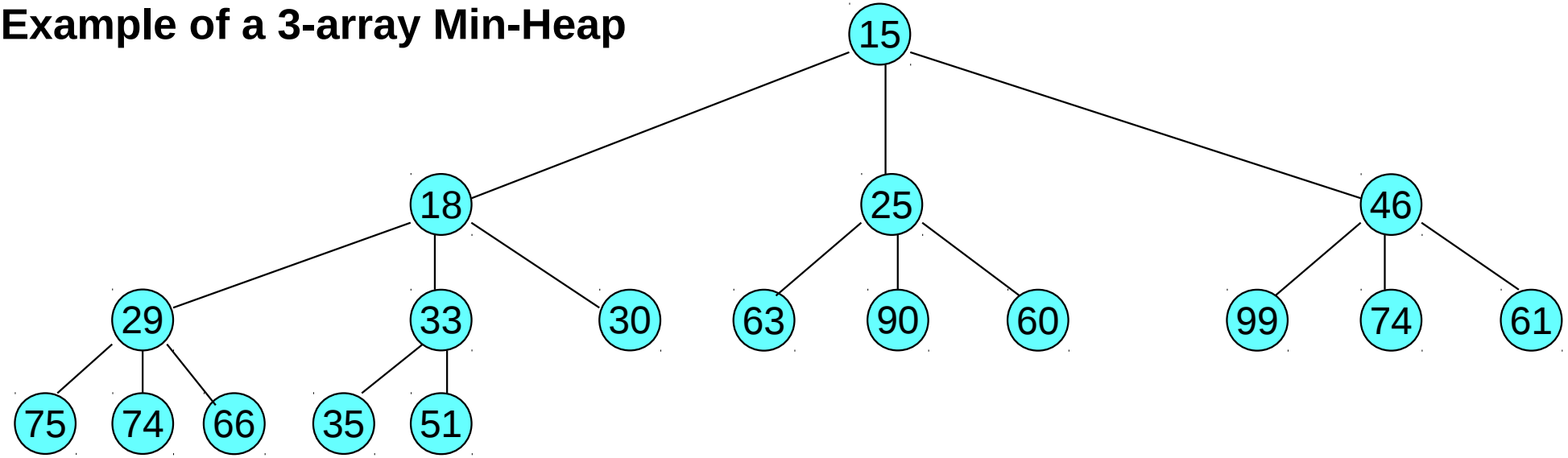
Example of a 3-array Min-Heap



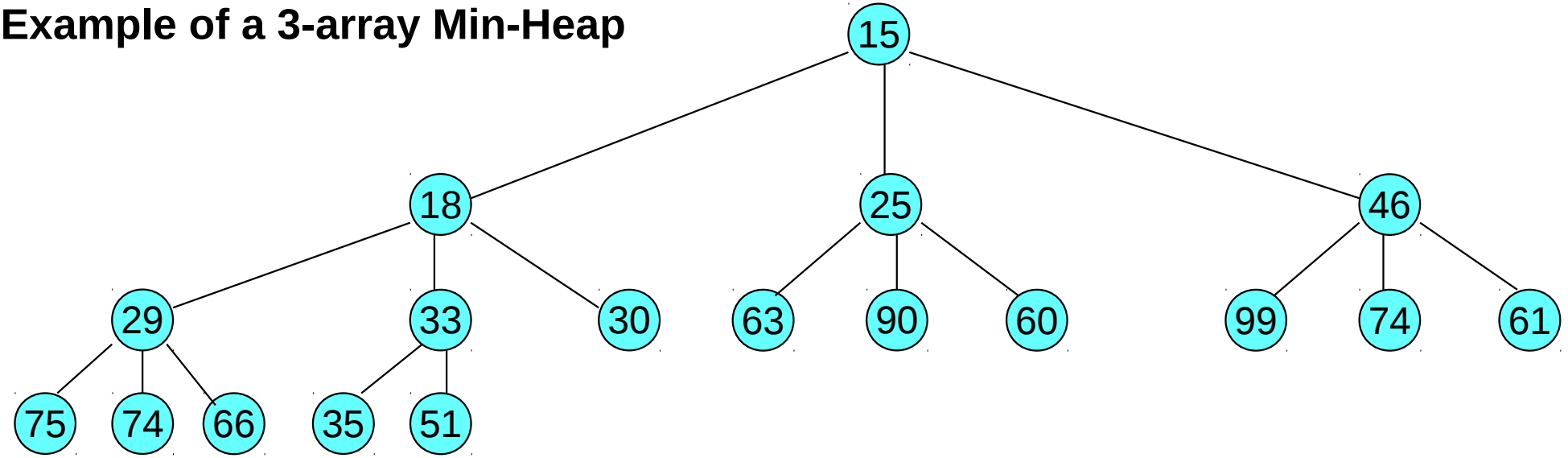
Min Heap Property: parent is less than or equal to its children

For a d-array heap:

Each node has d (or less) children.

There are d^t nodes in level t (except for the last level which may have fewer nodes).

Example of a 3-array Min-Heap

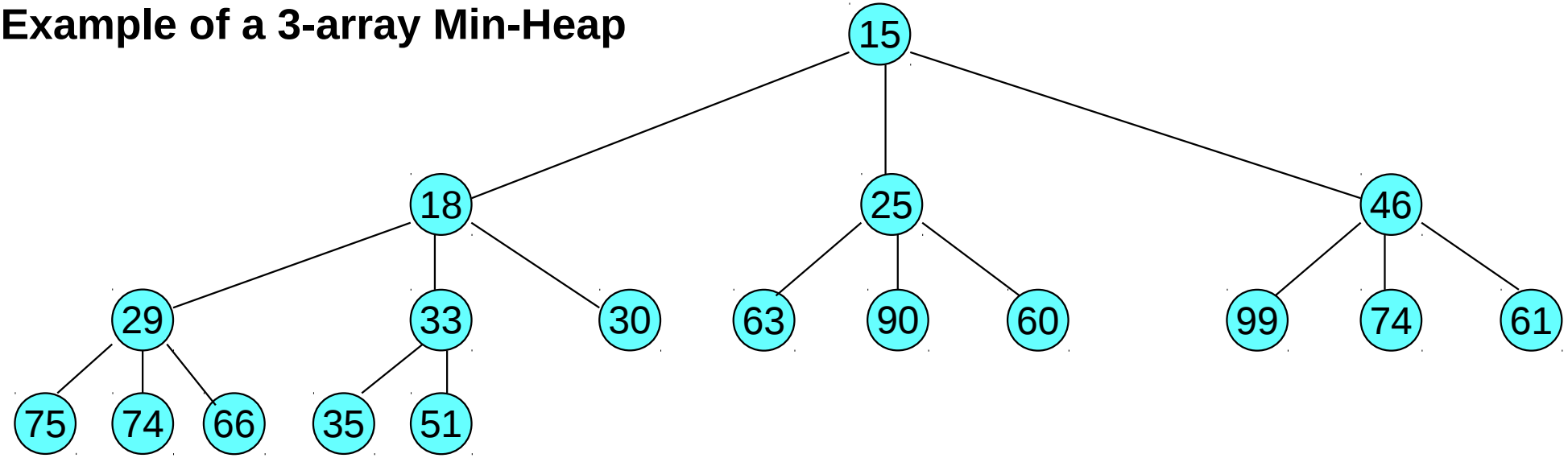


Min Heap Property: parent is less than or equal to its children

The underlying array which stores the heap elements looks like this...

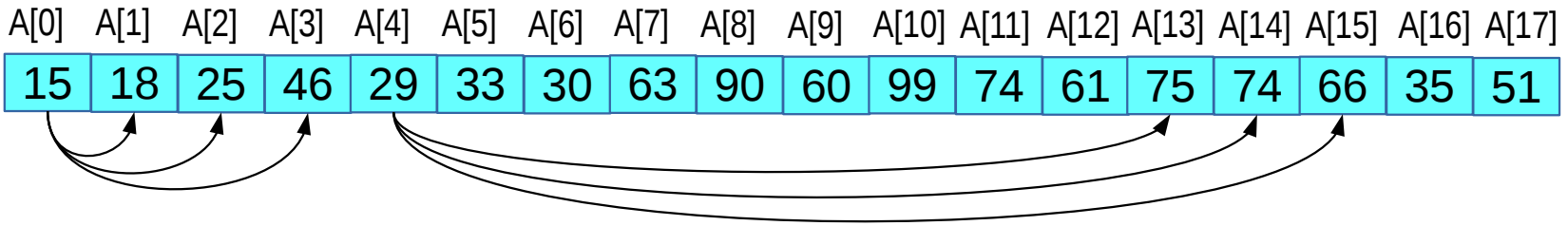
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]	A[13]	A[14]	A[15]	A[16]	A[17]
15	18	25	46	29	33	30	63	90	60	99	74	61	75	74	66	35	51

Example of a 3-array Min-Heap



Min Heap Property: parent is less than or equal to its children

The underlying array which stores the heap elements looks like this...



1st, 2nd and 3rd Children of A[0] are A[1], A[2] and A[3]
1st, 2nd and 3rd Children of A[4] are A[12], a[13] and A[14]

- Implement the following two functions. (these are NOT member funtions)
  `int Parent(int)`
  `int Child(int,int)`

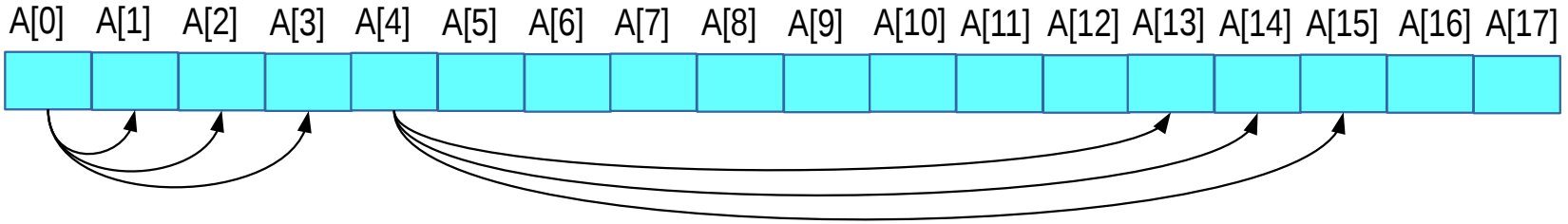
- `Parent(i)` should return the index of the parent of thr i-th element.

- `Child(i,j)` assumes that j is 1,2 or 3.
  `Child(i,1)`,`Child(i,2)` and `Child(i,3)` should return the indices of the first, second and third child of the i-th element.

Example

`Child(4,1)` is 13
`Child(4,2)` is 14
`Child(4,3)` is 15

`Parent(13)` is 4
`Parent(14)` is 4
`Parent(15)` is 4



1st, 2nd and 3rd Children of A[0] are A[1], A[2] and A[3]
1st, 2nd and 3rd Children of A[4] are A[12], a[13] and A[14]

- Implement the following two functions. (these are NOT member funtions)
  `int Parent(int)`
  `int Child(int,int)`

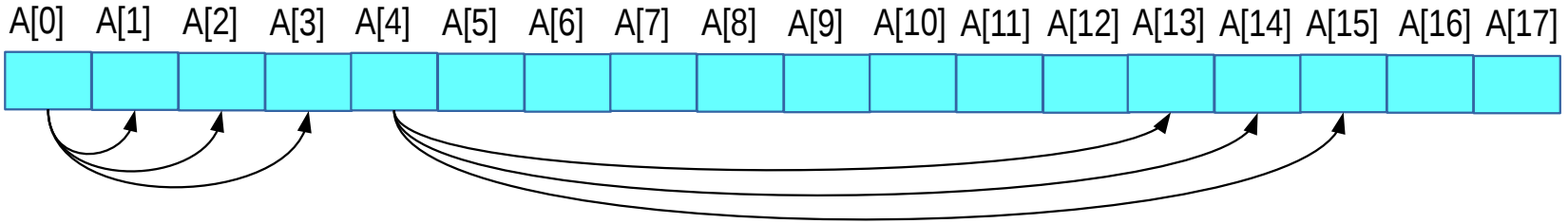
- `Parent(i)` should return the index of the parent of thr i-th element.

- `Child(i,j)` assumes that j is 1,2 or 3.
  `Child(i,1)`,`Child(i,2)` and `Child(i,3)` should return the indices of the first, second and third child of the i-th element.

Example

`Child(4,1)` is 13
`Child(4,2)` is 14
`Child(4,3)` is 15

`Parent(13)` is 4
`Parent(14)` is 4
`Parent(15)` is 4



1st, 2nd and 3rd Children of A[0] are A[1], A[2] and A[3]
1st, 2nd and 3rd Children of A[4] are A[12], a[13] and A[14]

Min-Heap-Property: `A[Parent(i)] ≤ A[i]`

Class Definition

```
class heap
{
    public:
    int *A, heap_size, length;

    heap(int);
    void MinHeapify(int);
    void BuildMinHeap(void);
    void Heapsort(void);
    int HeapMinimum(void);
    int HeapExtractMin(void);
    void HeapDecreaseKey(int,int);
    void MinHeapInsert(int);
};
```

`A[0], A[1], ...` stores the heap elements.

Size of heap (which could be less than array length)

Length of array A

In the constructor we must,

- **Initialise** `length` and `heap_size`
- Dynamically allocate space for `A[0], ... ,A[length-1]`

Class Definition

```
class heap
{
    public:
    int *A, heap_size, length;

    heap(int);
    void MinHeapify(int);
    void BuildMinHeap(void);
    void Heapsort(void);
    int HeapMinimum(void);
    int HeapExtractMin(void);
    void HeapDecreaseKey(int,int);
    void MinHeapInsert(int);
};
```

`A[0], A[1], ...` stores the heap elements.

Size of heap (which could be less than array length)

Length of array A

In the constructor we must,

- **Initialise `length`** and **`heap_size`**
- Dynamically allocate space for `A[0], ... ,A[length-1]`

Constructor

```
heap::heap(int s)
{
  A = new int[s];
  heap_size=0;
  length=s;
}
```

Notice that the constructor takes one parameter (which is supposed to be the array length)

Class Definition

```
class heap
{
    public:
    int *A, heap_size, length;

    heap(int);
    void MinHeapify(int);
    void BuildMinHeap(void);
    void Heapsort(void);
    int HeapMinimum(void);
    int HeapExtractMin(void);
    void HeapDecreaseKey(int,int);
    void MinHeapInsert(int);
};
```

Analogous to Max-Heapify in Cormen but enforces **Min-Heap-Property**

Analogous to Build-Max-Heap in Cormen but builds a **Min-Heap**

Analogous to Heapsort in Cormen but sorts in **descending order**

Analogous to Heap-Maximum in Cormen but returns the **Minimum element**

Analogous to Heap-Extract-Min in Cormen but Extracts the **Minimum element**

Analogous to Heap-Increase-Key in Cormen but **decreases** the key of an element in **Min-Heap**

Analogous to Max-Heap-Insert in Cormen but inserts element in a **Min-heap**