119188651
kshitij

Homework 2

ENPM 809X
February 23, 2023

Problem 1

Sample Space $S = A[1...n]$ and the event is switch at i.
Probability of switch occurring is

$$P\{switch\} = \begin{cases} 1 & \text{if switch occurs} \\ 0 & \text{if switch does not occur} \end{cases} \tag{1}$$

$$E[X_s] = 1 \cdot P(switch) + 0 \cdot P(noswitch) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

for n such events

$$E[X_s] = E\left[\sum_{i=1}^{n} X_i\right] = \frac{n}{2}$$

Problem 2

We can prove that a random subset of size m is created by induction on m.
When $m == 0$ only one subset of size m is possible.
If S is a subset of size $m-1$ of $n-1$ (assumption): ∀ ∈(n-1),P(x $\in S$) = $\frac{m-1}{n-1}$. Let S‘ be the returned subset then

$$P(x \in S‘) = P(x \in S) + P(x \notin S \wedge i = x)$$

Where i is a random element from 1 to n.
Since

$$P(x \in S) = \frac{m-1}{n-1}$$

therefore

$$P(x \notin S) = (1 - \frac{m-1}{n-1})$$

and

$$P(i = x) = \frac{1}{n}$$

as i is taken from a set of size 1 to n at random.
Which leads to

$$\frac{m-1}{n-1} + (1 - \frac{m-1}{n-1})\frac{1}{n}$$

$$= \frac{m-1}{n-1} + \left(\frac{(n-1)-(m-1)}{n-1}\right)\frac{1}{n}$$

$$= \frac{m-1}{n-1} + \frac{n-m}{n(n-1)}$$

$$= \frac{n(m-1)+n-m}{n(n-1)}$$

$$= \frac{m}{n}$$

Since the subset contains all elements of $(n-1)$ with the correct probability $\frac{m-1}{n-1}$, it must also contain n with the probability $\frac{m}{n}$ as the probabilities sum to 1

Figure 1: Recursive calls of the algorithm

Problem 4

1. $PARENT(i) = [i/d]$, and $CHILD(j, i) = d * i - d + j + 1$, where $CHILD(k, i)$ gives the k^{th} child of the node i when representing an array as a *d-ary* tree.

2. The height of a binary tree is given as $log_2 n$ where n is the number of nodes. i.e. $(2^x = n)$ from this we can deduce that the height of a d-ary tree will be equal to $log_d n$.

3. Algorithm 1: Max Extract

```
DMAX-HEAP(A, i)
    l = i
    for j = 1 to d
        if CHILD(j,i) <= A.heap_size and A[CHILD(k,i)] > A[i]
            if A[CHILD(k,i)] > largest
                largest = A[CHILD(k,i)]
            end
        end
    end
    if l \neq i
        swap A[i] and A[l]
        DMAX-HEAP(A,l)
    end
swap A[1] and A[heap-size]
DMAX = A[heap-size]
```

Analysis: Getting the max has a constant time complexity plus the complexity of DMAX-HEAP. DMAX-HEAP is similar to the MAX-HEAP algorithm with minor changes. Since the complexity of MAX-HEAP was dependent on its height $O(log_2 n)$ the complexity of DMAX-HEAP will also be dependent on its height i.e. $O(log_d n)$

Total Complexity: $O(log_d n) + O(2) = O(log_d n)$

4. Algorithm 2: Insert

```
increase heap size by 1
A[A.heap_size] = key
i = A.heap_size
while A[PARENT] < A[i] and i > 1
    swap A[i] and A[PARENT]
    i = PARENT
end
```

The complexity of the d-ary heap will depend on its height hence the complexity will be $O(log_d n)$

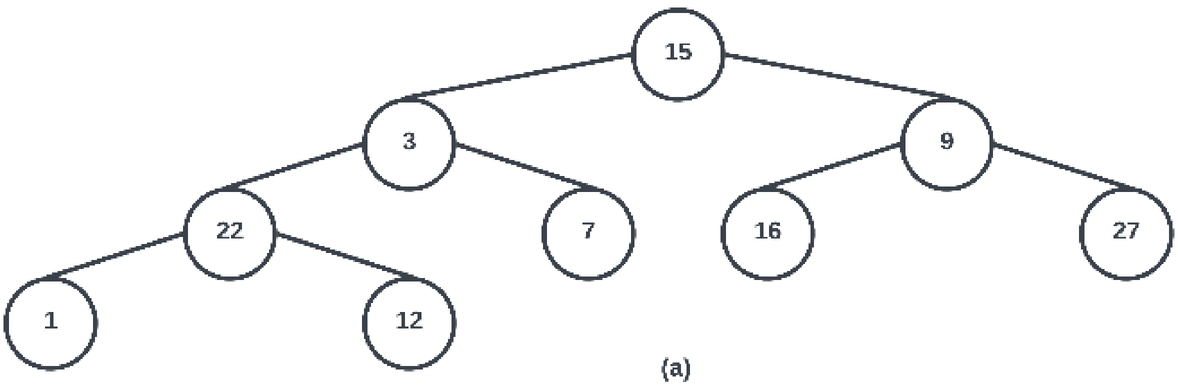
5. Algorithm 3: Increase key

```
if key < A[i]
    error
end
A[i] = key
while A[PARENT] < A[i] and i > 1
    swap A[i] and A[PARENT]
    i = PARENT
end
```

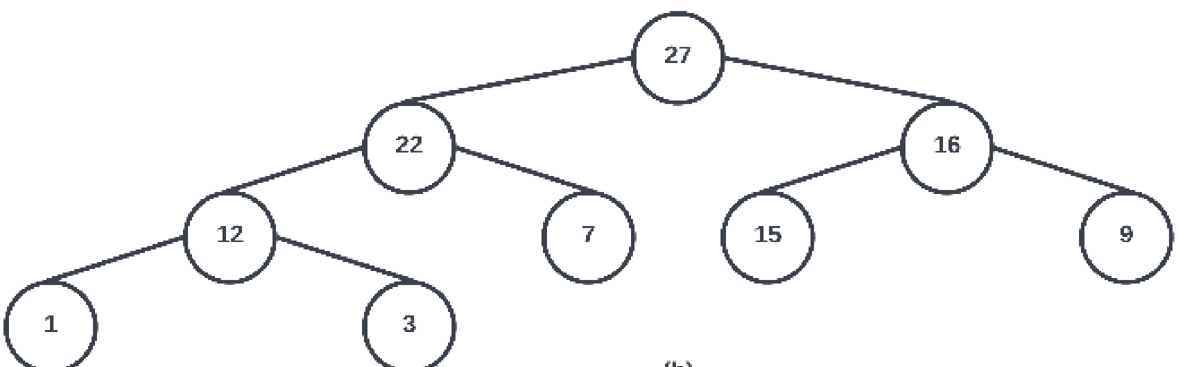
The complexity of the d-ary heap will depend on its height hence the complexity will be $O(log_d n)$

Problem 3

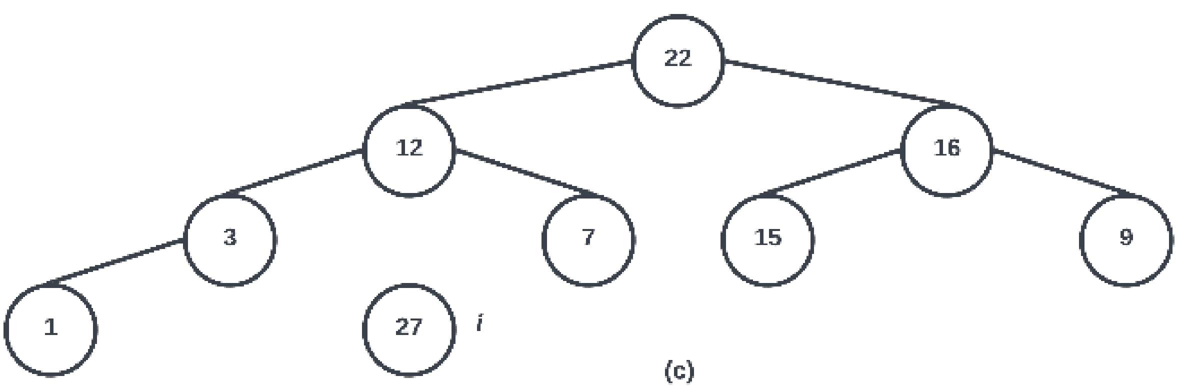
$$A = [15, 3, 9, 22, 7, 16, 27, 1, 12]$$



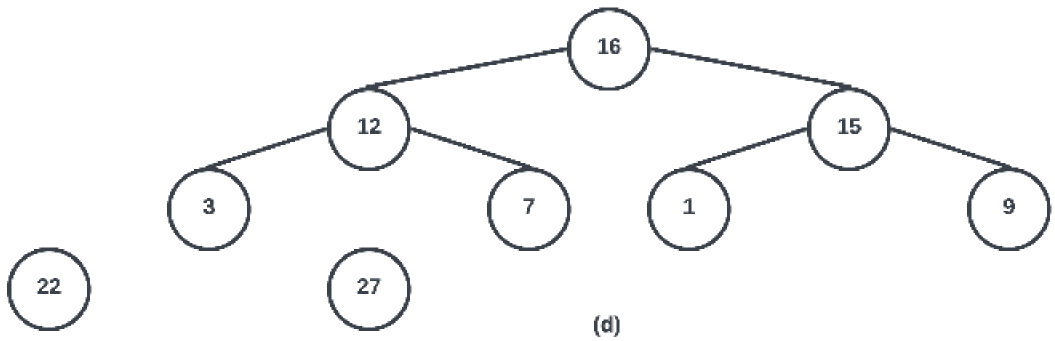
(a)



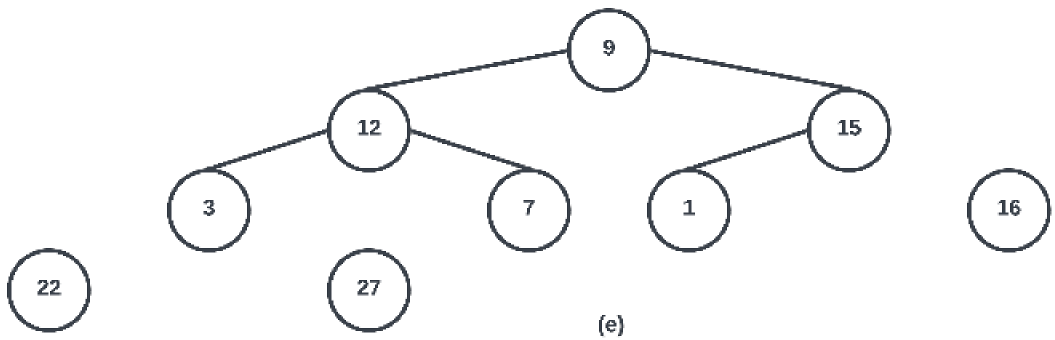
(b)



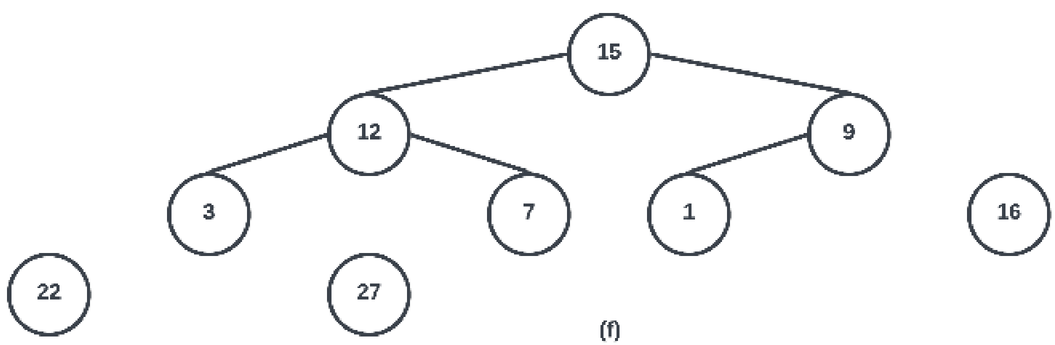
(c)


16
12
15
3
7
1
9
22
27

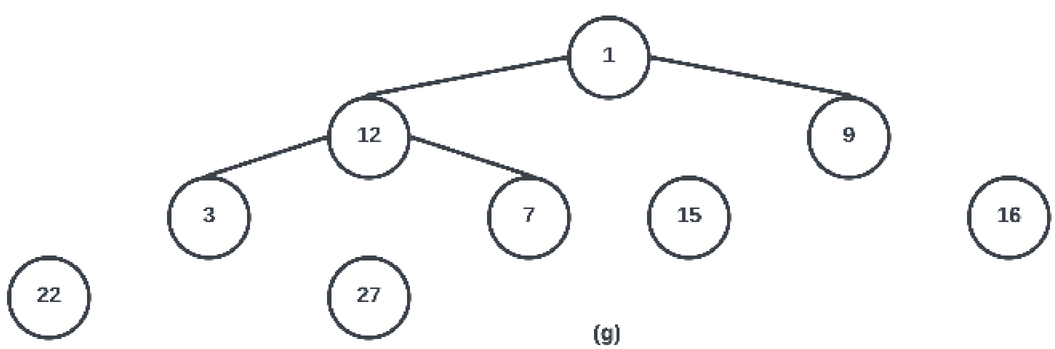
(d)


9
12
15
3
7
1
16
22
27

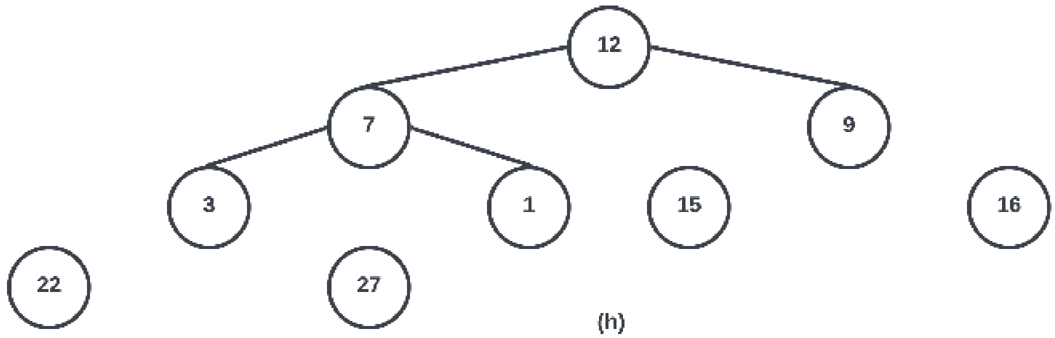
(e)


15
12
9
3
7
1
16
22
27

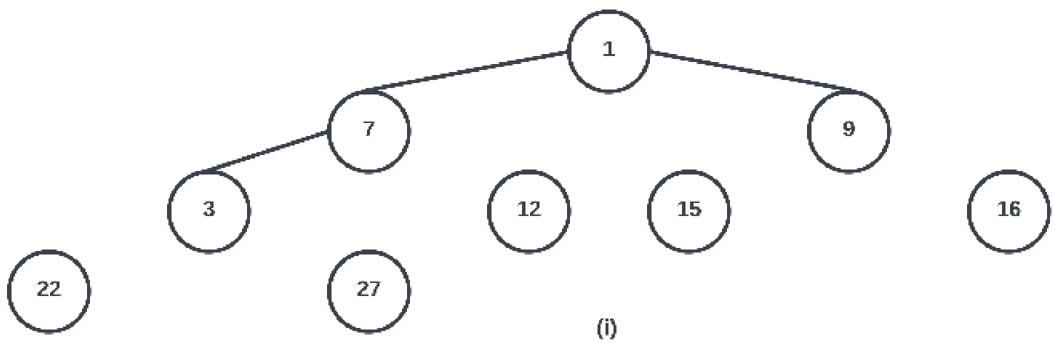
(f)


1
12
9
3
7
15
16
22
27

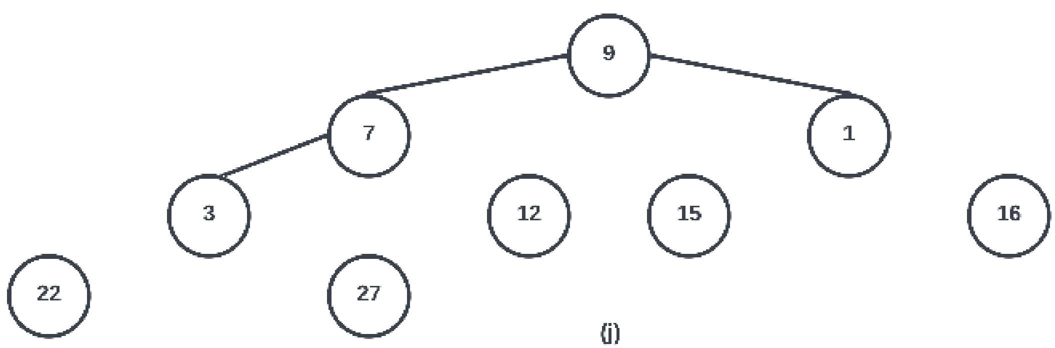
(g)


12
7
9
3
1
15
16
22
27

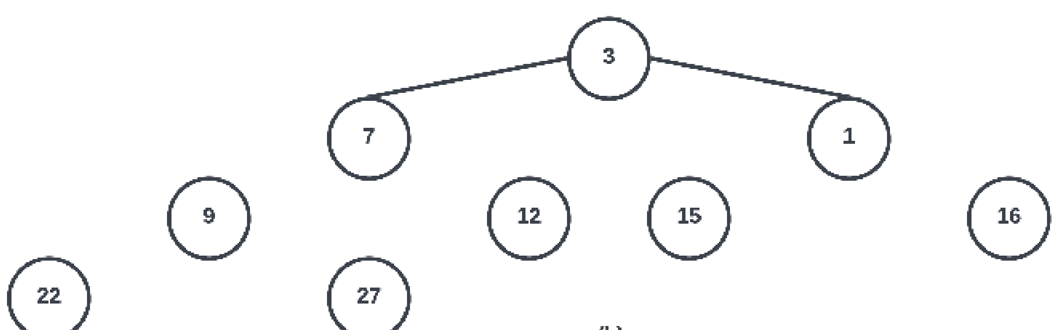
(h)


1
7
9
3
12
15
16
22
27

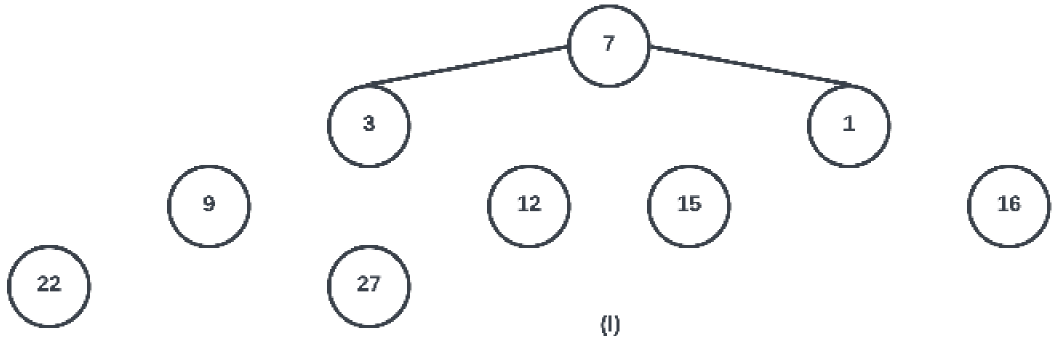
(i)


9
7
1
3
12
15
16
22
27

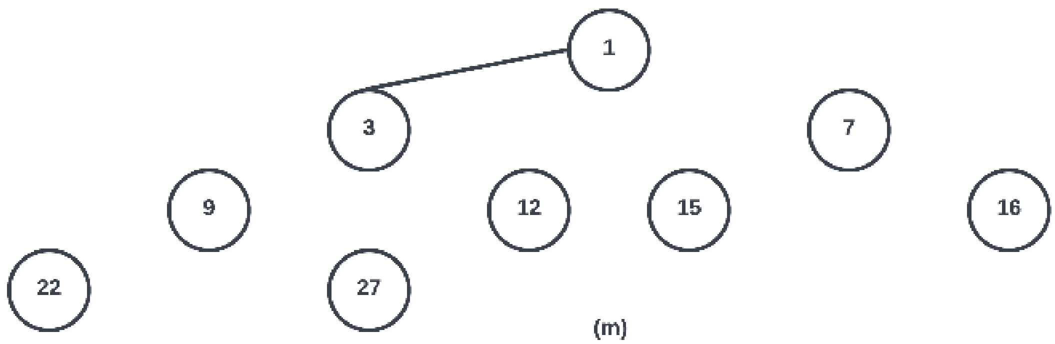
(j)


3
7
1
9
12
15
16
22
27

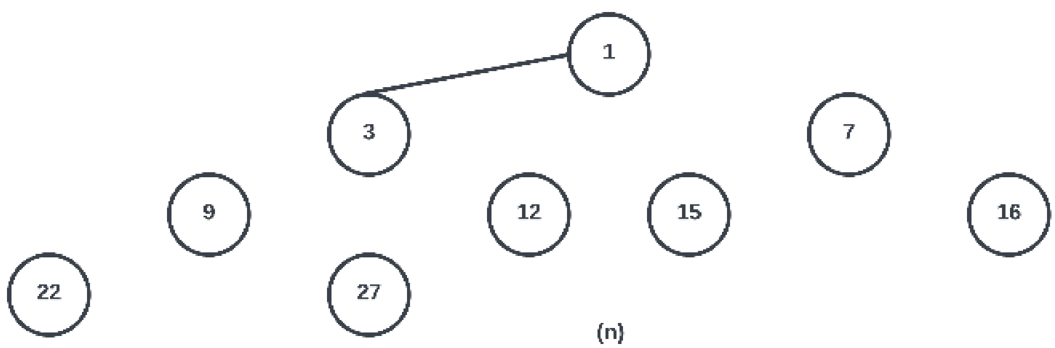
(k)


7
3
1
9
12
15
16
22
27

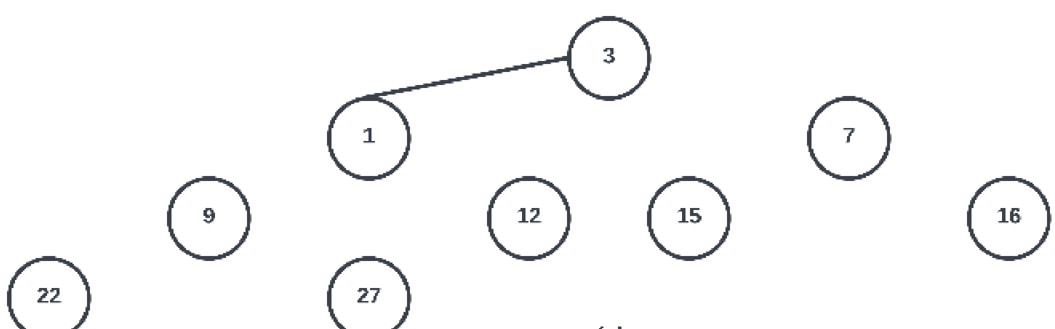
(l)


1
3
7
9
12
15
16
22
27

(m)


1
3
7
9
12
15
16
22
27

(n)


3
1
7
9
12
15
16
22
27

(o)

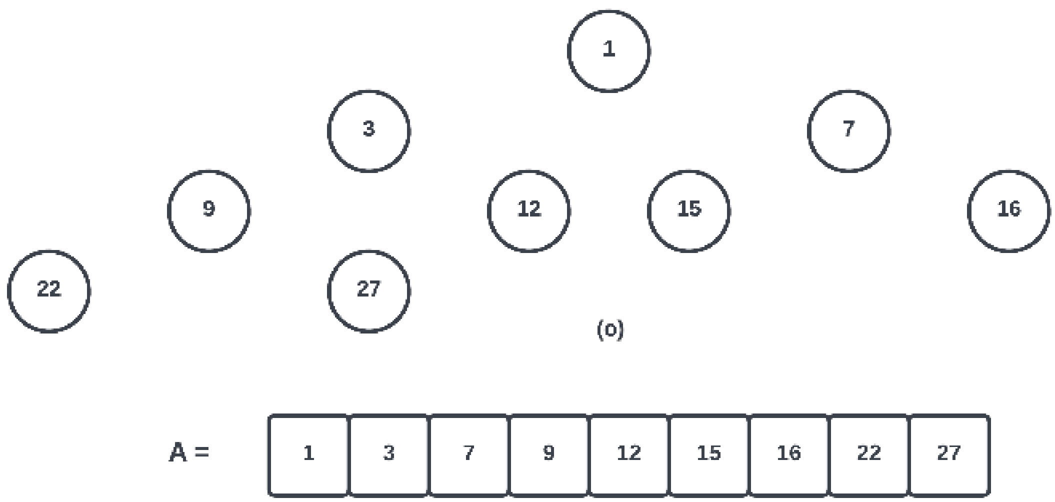


Figure 2: Heapsort