الجلسة 1

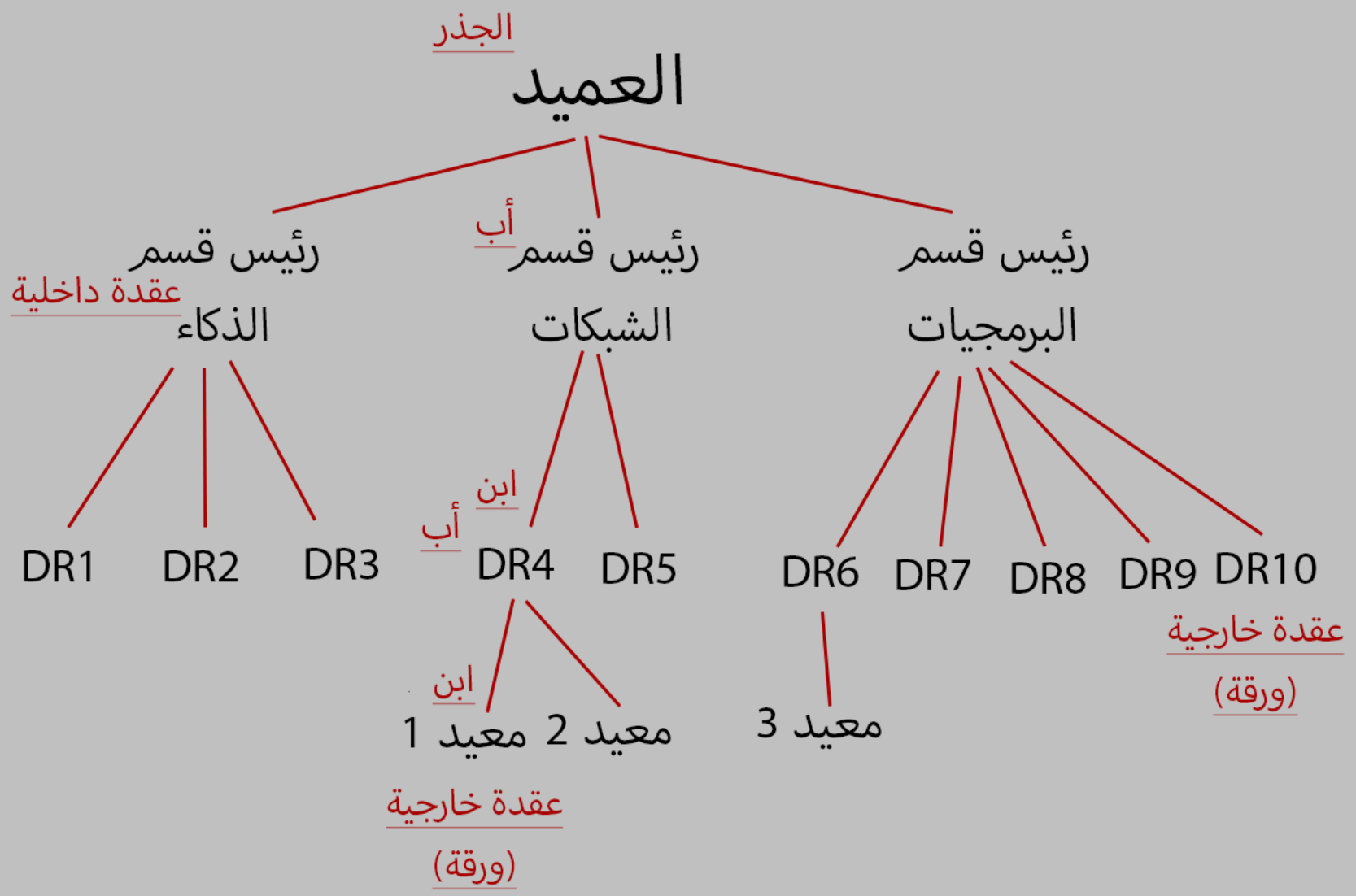
ورشة خوارزميات 2

Algorithms & Data Structures 2

- Trees
- Binary Trees
- Binary Search Trees (BST)
- Heap
- AVL Trees
- B-Trees
- Graphs

تعاريف:

عقدة
وصلة
ابن
أب
حفيد
سلف (جد)
عقدة خارجية
عقدة داخلية


الجذر
العميد
رئيس قسم
الذكاء
عقدة داخلية
أب
رئيس قسم
الشبكات
رئيس قسم
البرمجيات
DR1
DR2
DR3
ابن
أب
DR4
DR5
DR6
DR7
DR8
DR9
DR10
عقدة خارجية
(ورقة)
ابن
معيد 1
معيد 2
معيد 3
عقدة خارجية
(ورقة)

تعاريف:

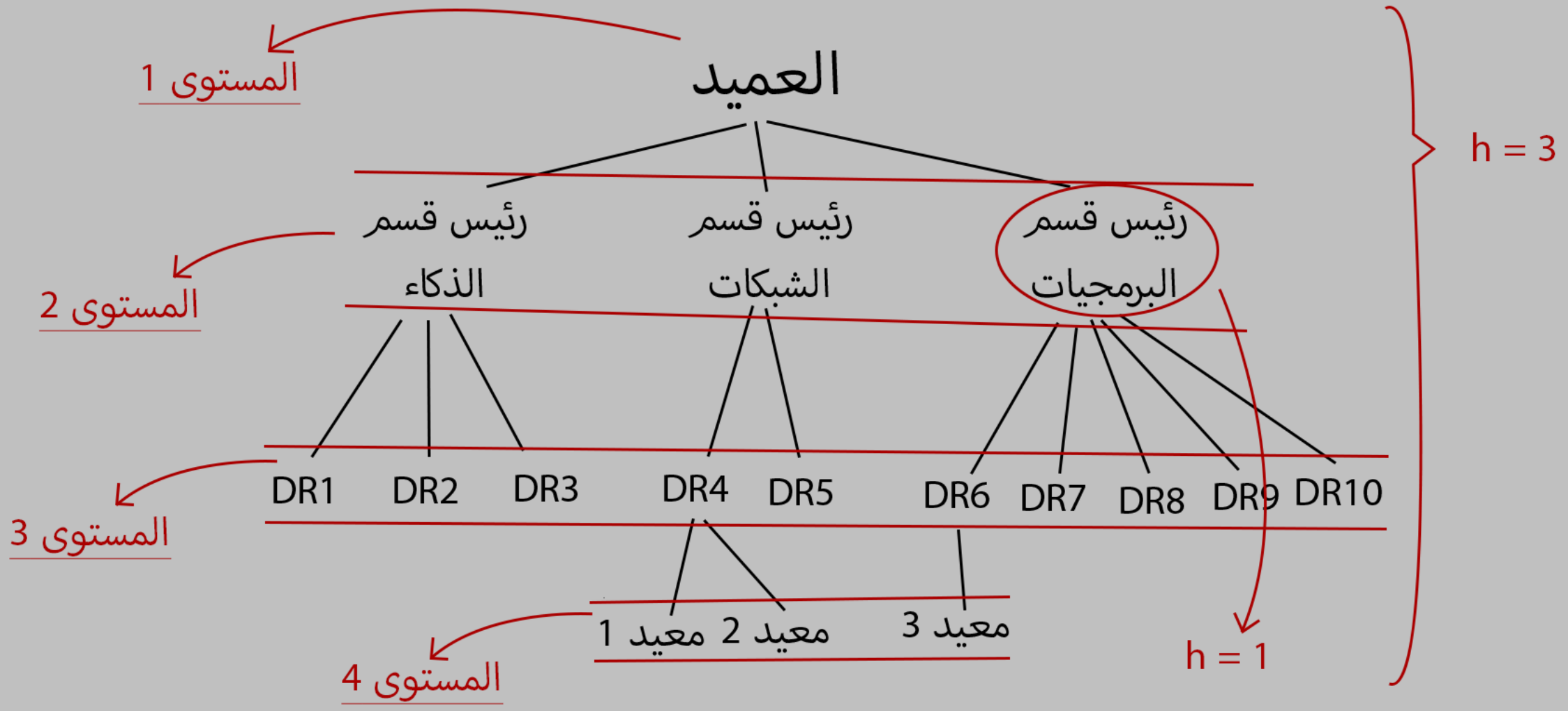
ارتفاع عقدة

ارتفاع الشجرة

المستوى

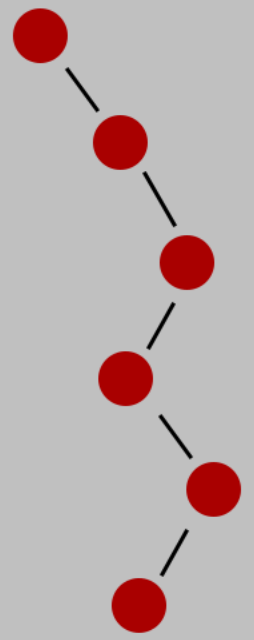
درجة عقدة

درجة الشجرة

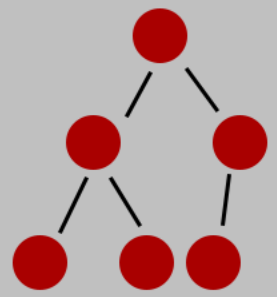

المستوى 1
العميد
h = 3
رئيس قسم الذكاء
رئيس قسم الشبكات
رئيس قسم البرمجيات
المستوى 2
DR1
DR2
DR3
DR4
DR5
DR6
DR7
DR8
DR9
DR10
المستوى 3
معيد 1
معيد 2
معيد 3
h = 1
المستوى 4

Binary Trees

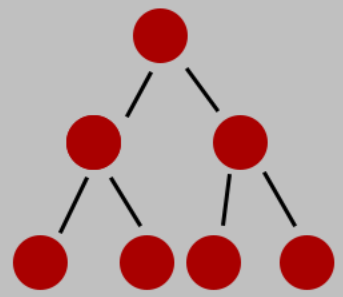
خطية
linear



تامة
complete



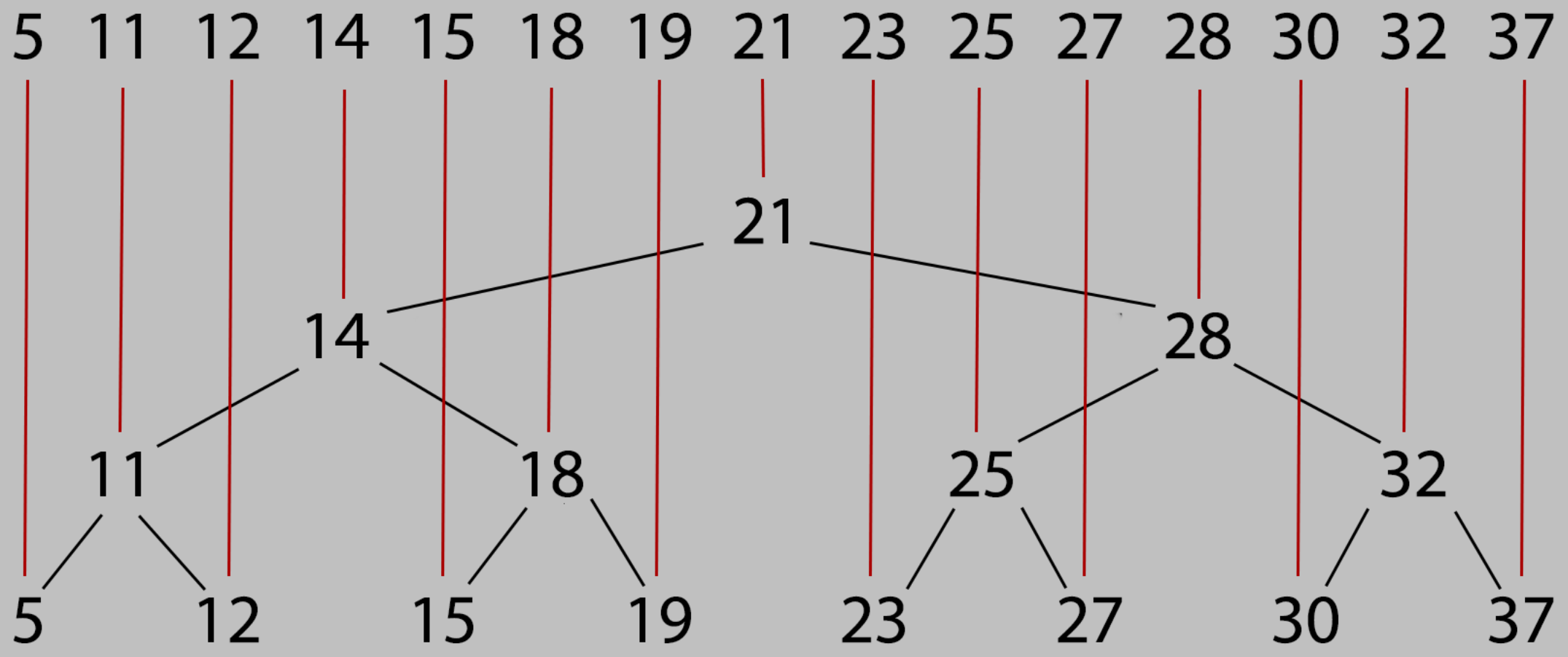
كاملة
full



Binary Search Trees (BST)

5 11 12 14 15 18 19 21 23 25 27 28 30 32 37

Binary Search Trees (BST)



BST Operations

(insertion)

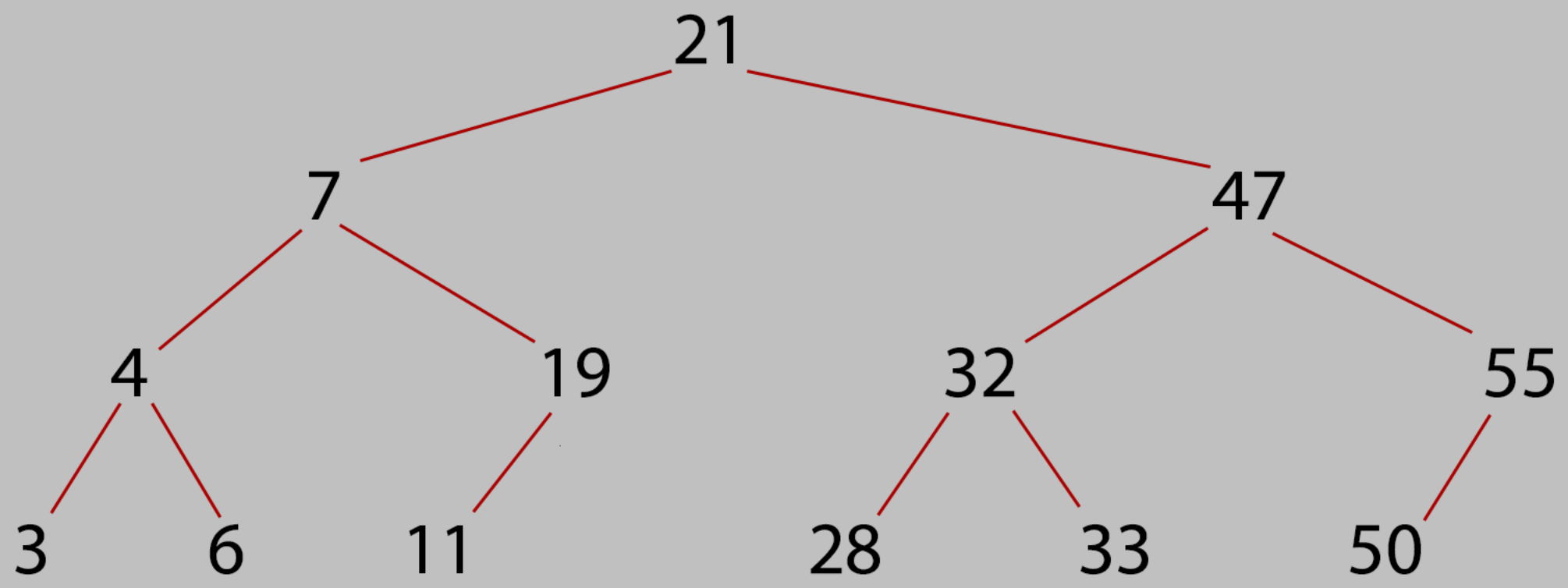
عقدة داخلية
(غير مطلوبة)

ورقة

جذر

BST Operations
(insertion)

مثال:



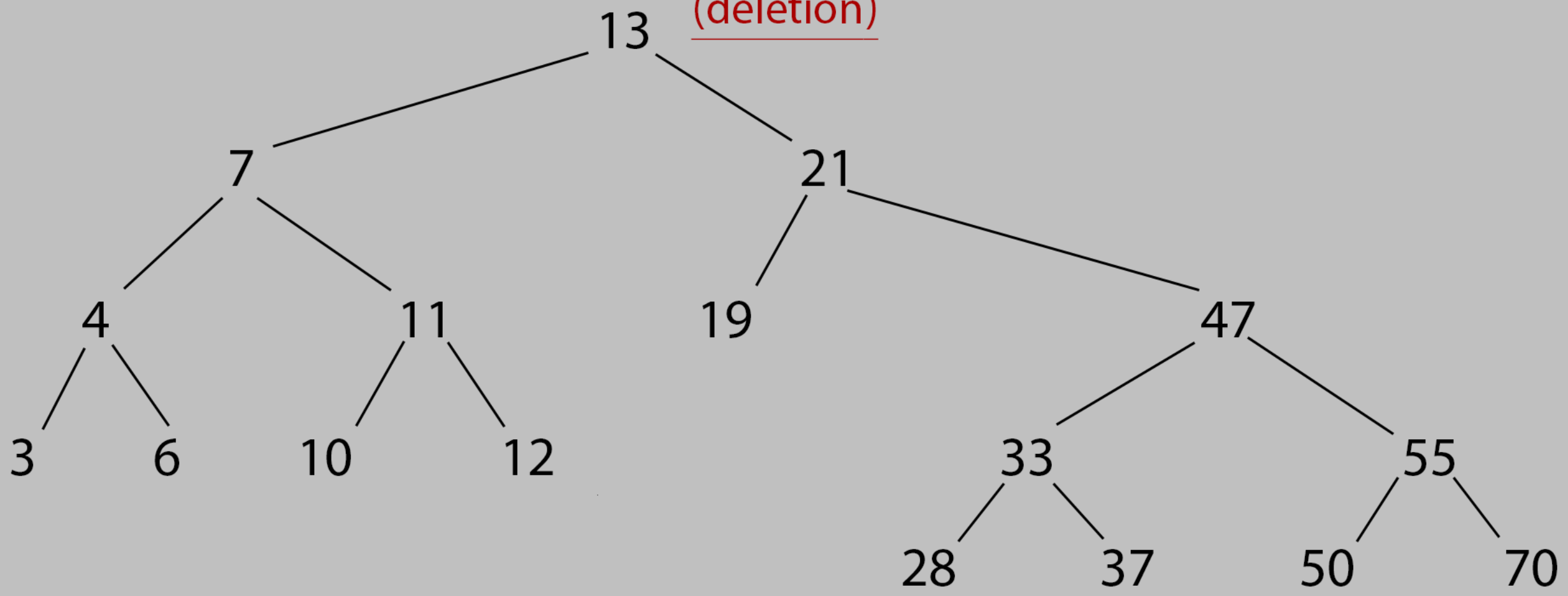
BST Operations

(deletion)

- حذف عقدة خارجية (ورقة)
  - قص
- حذف عقدة داخلية
  - لها ابن وحيد
    - تخطي
  - لها ابنين اثنين
    - تبديل بالتالي

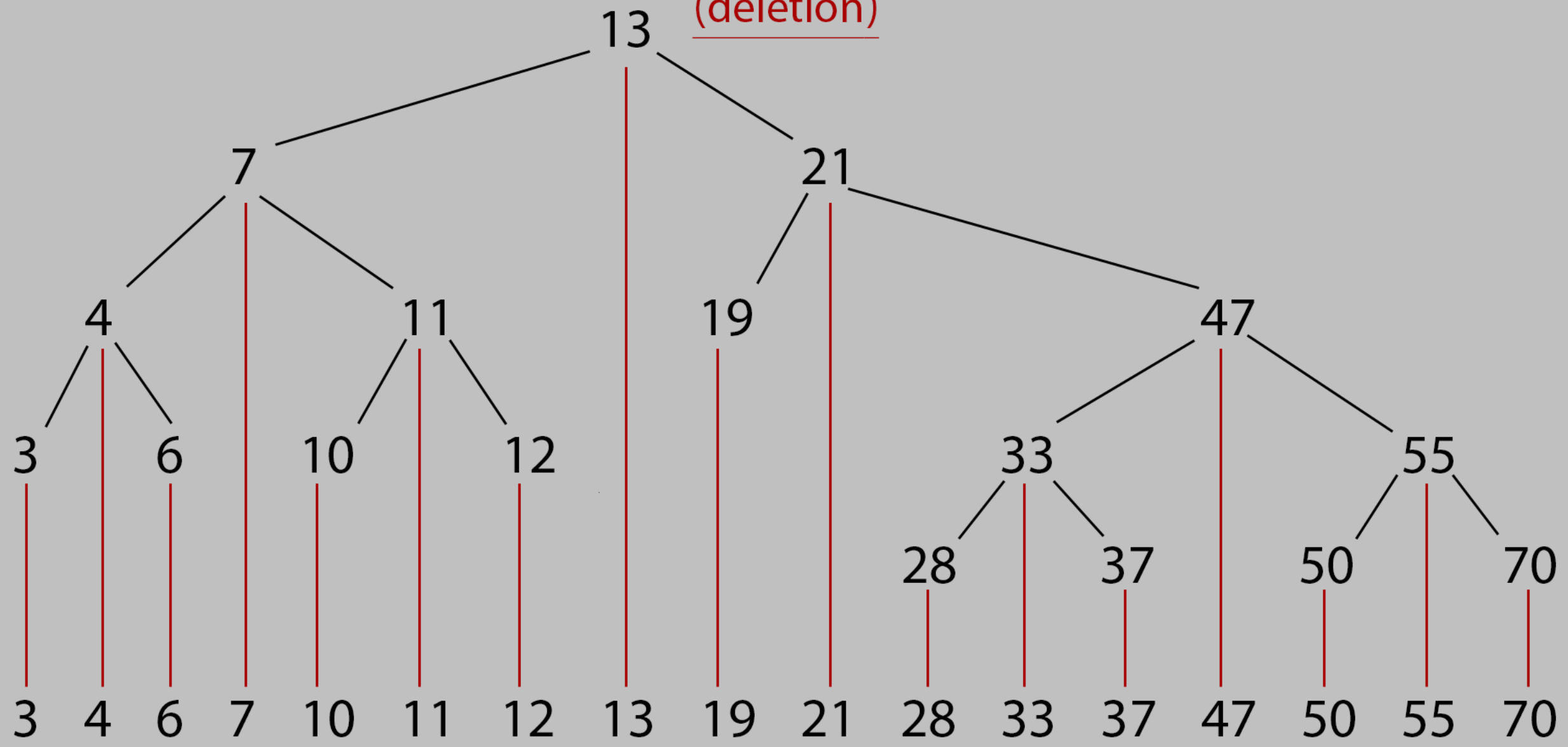
BST Operations

(deletion)



BST Operations

(deletion)



BST Operations

(Code)

```
struct Node {
    Data x;
    Node left;
    Node right;
    Node parent;
    int depth;
}
```

pointers

BST Operations

(Code)

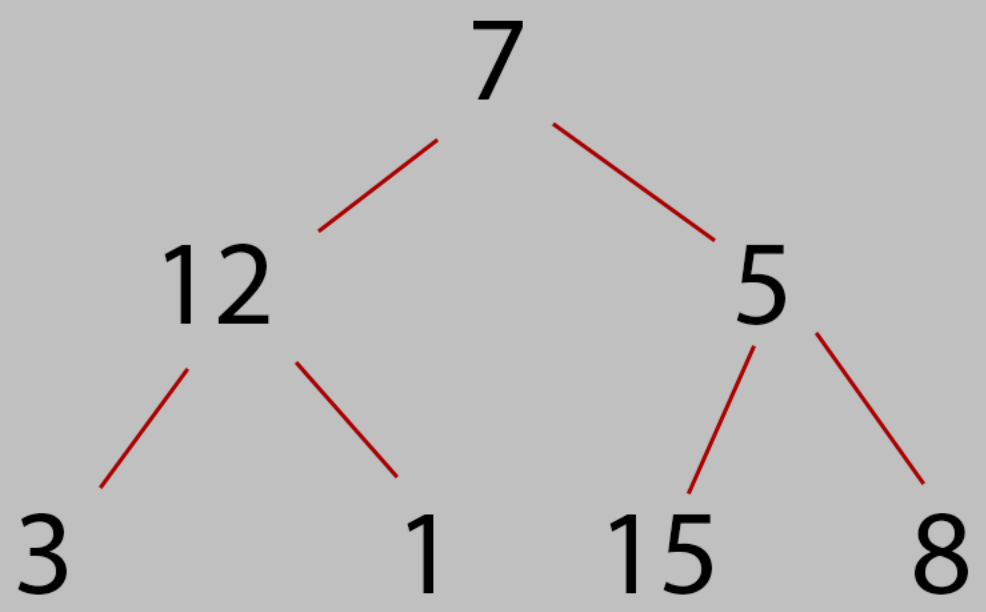
```
Node search(Node root, int x){
    if (root == NULL)
        return NULL;
    if (root.data == x)
        return root;
    if (root.data < x)
        return search(root.left, x);
    if (root.data > x)
        return search(root.eight, x);
}
```

```
void add(int x){
    Node p = new Node();
    p.data = x;
    if(root == NULL){
        root = p;
        return ;
    }
    Node v = root;
    while(v != NULL){
        if(x < v.data){
            if(v.left = NULL){
                v.left = p;
                return;
            }
            v = v.left;
        }
        else{
            if(v.right= NULL){
                v.right= p;
                return;
            }
            v = v.right;
        }
    }
}
```

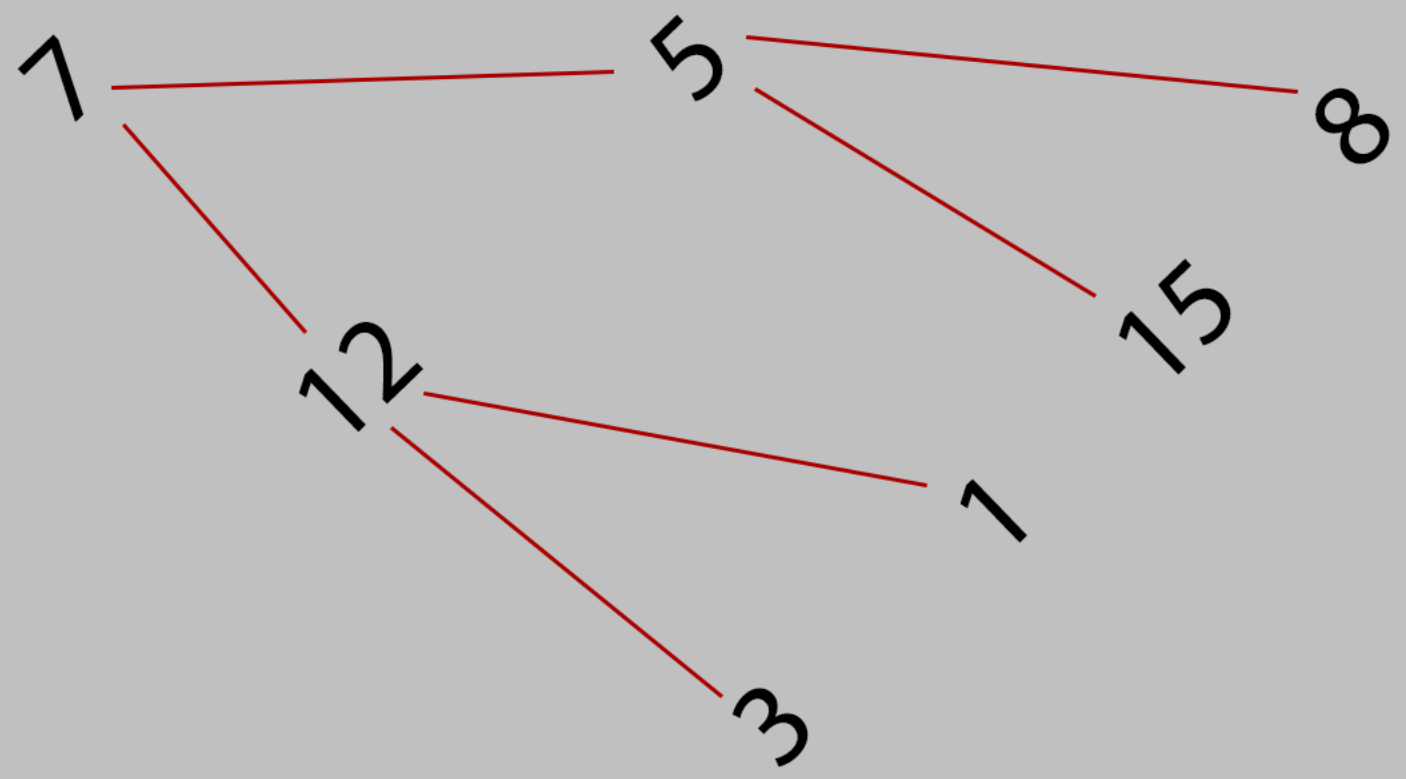
```
void searchAndDelete(Node &p, int x){
        if(p == NULL)
            return;
        if(p.data == x)
            deleteNode(p);
        else if(x < p.data)
            searchAndDelete(p.left, x);
        else if(x > p.data)
            searchAndDelete(p.right, x);
}
```

```
void deleteNode(Node &p){
    if(p.left == NULL)
        p = p.right;
    else if(p.right == NULL)
        p = p.left;
    else{
        Node v = getNext(p);
        p.data = v.data;
        delete(v);
    }
}
```

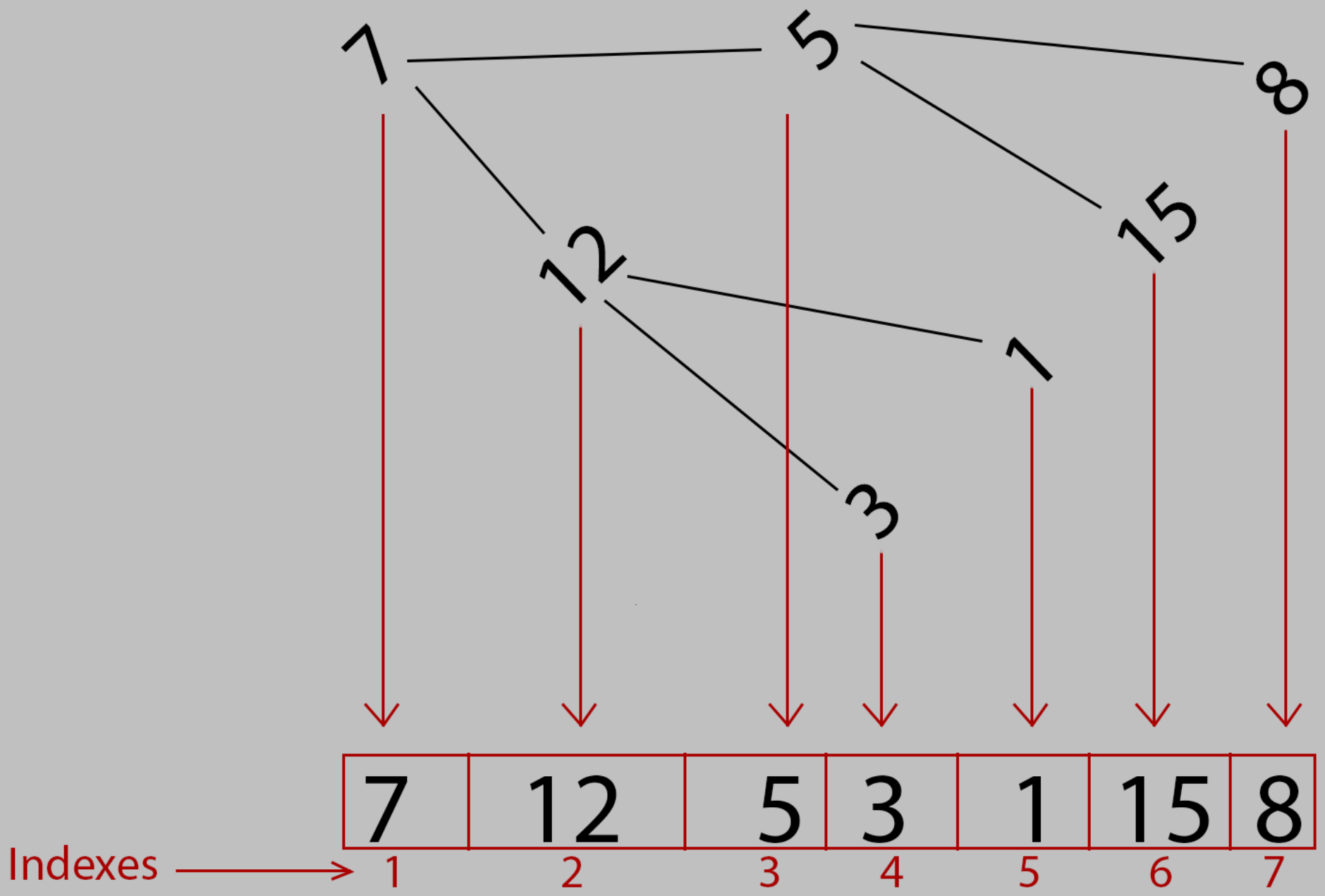
```
Node getNext(Node p){
        p = p.right;
        while(p.left != NULL)
                p = p.left;
        return p;
}
```



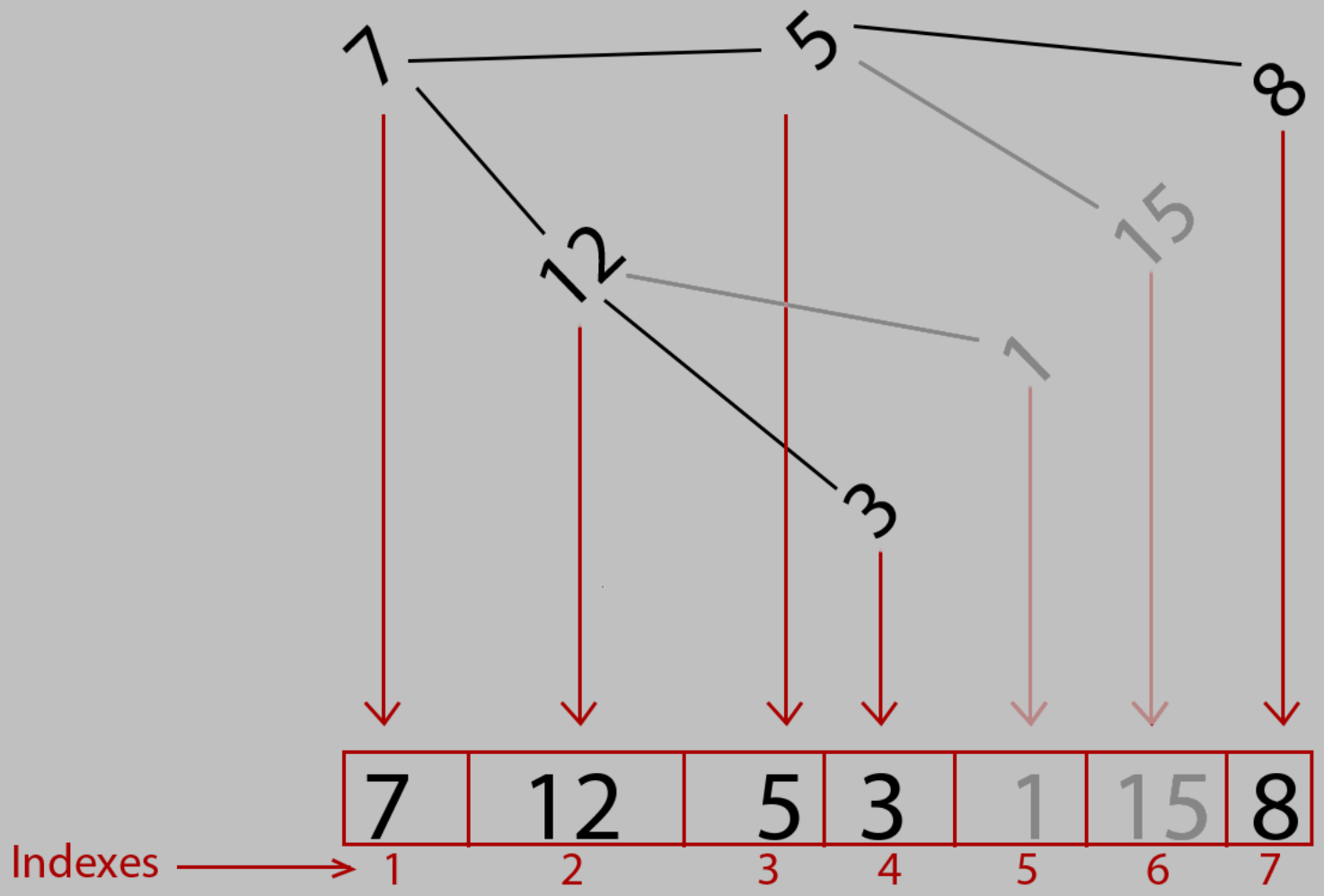
معلش نميلها شوي ؟



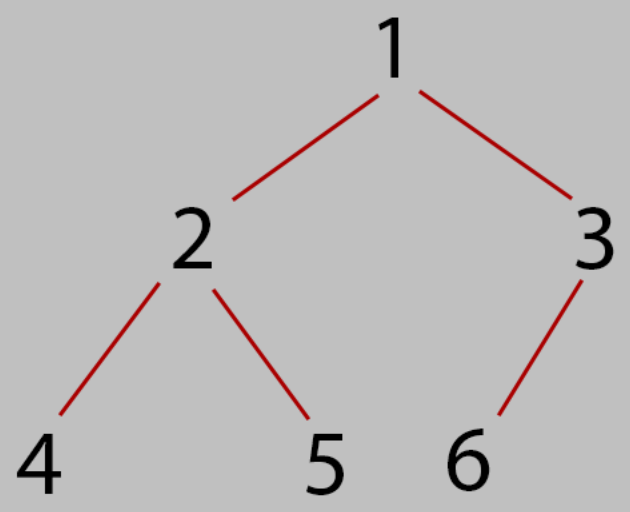
عادي هيك ؟


7
5
8
12
15
1
3
7
12
5
3
1
15
8
Indexes
1
2
3
4
5
6
7

ماذا لو لم تكن الشجرة تامة أو كاملة ؟!

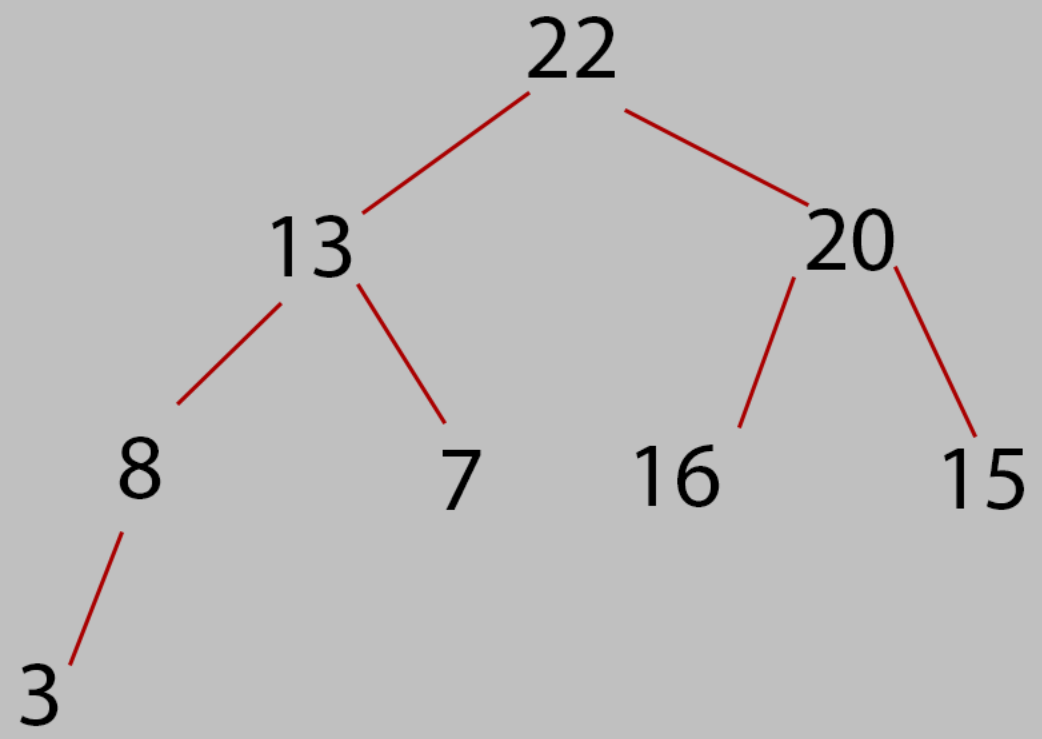


min-heap



1	2	3	4	5	6

max-heap



22	13	20	8	7	16	15	3

```
void maxThree(int i){
    int left = 2*i;
    int right = 2*i+1;
    int maxi = i;
    if(right <= heapsize && A[right] > A[maxi])
        maxi = right;
    if(left <= heapsize && A[left] > A[maxi])
        maxi = left;

    if(maxi != i){
        swap(A[i], A[maxi]);
        maxThree(maxi);
    }
}
```

```
void buildMaxHeap(){
    for(int i = n/2;i >= 1;i --)
        maxThree(i);
}
```