

## Linear Search

60	1	88	10	11	100
[0]	[1]	[2]	[3]	[4]	[5]

array এর স্থির পদক্ষেপ করেনা অবস্থার প্রতি দ্বে ক্ষেত্র, মাত্র  
কর্তৃত index দ্বা প্রয়োজু।

```
#include <iostream>
using namespace std;
int linear_Search(int A[], int n, int x)
{
    int i; [i = index no.]
    for (i=0; i<n; i++)
    {
        if (A[i] == x)
            return i;
    }
    return -1;
}
```

```
int main()
{
    int A[] = { 60, 1, 88, 10, 11, 100 };
    int x = 11;
    int n = sizeof(A)/sizeof(A[0]);
    int res = linear_Search(a, n, x);
    cout << "The elements of the array are : ";
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
```

```
Cout << "In Element to be searched is " << x;
if (res == -1)
    Cout << "In Element is not present in the array";
else
    Cout << "InElement is present at " << res << "Position of array";
return 0;
}
```

## Binary Search

ଦେଖି ପରେ କହିବା କଥା ପରେ ଦେଖି ଆମ୍ବାର ରମଣାନ୍ତା  
-ଶ୍ରୀମଦ୍ ।

କାଳେ ଅର୍ଥାତ୍ ମାଧ୍ୟମ କିମ୍ବା ଏହି ବ୍ୟକ୍ତିରେ କାହାରେ

1	2	3	4	5	6
w		w	v	v	

কামিনি অংশ হের ক্রান্তি প্রযুক্তি সহজের টি-লেখে ক্ষেত্রে দৃঢ় হচ্ছে।  
 একটি বিশেষ ক্ষেত্রে Binary Search টি-লেখে  $\log_2 n$  রে হচ্ছে।  
 নিচের তালো।

$\log_2 4 = 2$  or Binary Search 7.3.6 201

$$\log_2^{16} = 4$$

1	4	6	2	10	19	22	23	30
---	---	---	---	----	----	----	----	----

```

#include <iostream>
using namespace std;

int binarySearch( int A[], int n, int x)
{
    int left, right, mid;
    left = 0; [index no]
    right = n - 1; [index no]
    while (left <= right)
    {
        mid = (left + right) / 2;
        if (A[mid] == x)
            return mid;
    }
    if (A[mid] < x)
        left = mid + 1;
    else
        right = mid - 1;
    return -1;
}

int main()
{
    int A[] = {1, 4, 6, 2, 10, 19, 22, 23, 30};
    int x = 7;
    int res = binarySearch(A, 0, n - 1, x);
    cout << "The elements of the array : ";
    for (int i = 0; i < n; i++)
        cout << A[i] << " ";
    cout << endl;
    cout << "Element to be searched is : " << x;
    if (res == -1)
        cout << "Element is not exist";
    else
        cout << "Element is present at ";
    cout << res << " position of array";
    return 0;
}

```

## Selection Sort

10	5	2	8	3
----	---	---	---	---

କୌଣସି ନାହାର ଏହାଦୁଇମି କବର୍ତ୍ତା କଲାପ ଅଛି (2/2)

ଏହା ଏହା କାମନାର ମଧ୍ୟ କୁଟି କାମନା କାମନା କାମନା ଏହା

କି କାମନା କାମନା ଏହା କାମନା କାମନା କାମନା ଏହା ।

ଏହାର କାମନା କାମନା କାମନା କାମନା କାମନା ଏହା ଏହା ଏହା ଏହା

ଏହାର ସ୍ଥଳ କାମନା ଏହା କାମନା କାମନା କାମନା ।

ଏହାର କାମନା ଏହାର କାମନା ।

ex:

```
#include <iostream>
using namespace std;
void Selection_Sort(int A[], int n)
{
    int i, j, index_min, temp;
    for (i=0; i<n-1; i++) {
        index_min = i;
        for (j=i+1; j<n; j++) {
            if (A[j] < A[index_min])
                index_min = j;
        }
    }
}
```

5, 2, 6, 3, 4

↓      ↓

swap

if  $A[j]$  is less than  $A[index\_min]$ .  
 $\Rightarrow index\_min = j;$

```

if (index_min == i) {
    temp = A[i];
    A[i] = A[index_min];
    A[index_min] = temp;
}
}

void printArr(int A[], int n)
{
    int i;
    for (i=0; i<n; i++)
        cout << A[i] << " ";
}

int main()
{
    int A[] = {10, 5, 2, 8, 7};
    int n = sizeof(A)/sizeof(A[0]);
    cout << "Before sorting array elements are: " << endl;
    printArr(A, n);
    Selection(A, n);
    cout << "After sorting array elements are: " << endl;
    printArr(A, n);
    return 0;
}

```

## Bubble sort

10	5	2	8	2
----	---	---	---	---

ଛାତ୍ର ପରେ କଥା ମହିନା,

sort

ଏହାର ଦୁଇଟି ମାତ୍ର ହୁଏନା, କୌଣସି କିମ୍ବା ଆମର କାହାରେ, ଏହାର  
ଦୁଇଟି ଦୁଇଟି କାହାରେ ଏବଂ କିମ୍ବା ଆମର କାହାରେ ଏହାର କାହାରେ  
କାହାରେ କାହାରେ ।

ଏହାର କାହାରେ କାହାରେ ଆମର କାହାରେ କାହାରେ କାହାରେ କାହାରେ କାହାରେ Same  
Sorting 201 Continue, But 201

10	5	2	8	2
----	---	---	---	---

5 10 2 8 2

5 2 10 8 2

5 2 8 10 2

5 2 8 7 10 → fixed

2 5 8 2

2 5 8 2

2 5 7 2 → fixed

2 5 2

2 5

```

#include <iostream>
using namespace std;
void bubble_Sort (int A[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (A[j] > A[j + 1]) {
                temp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = temp;
            }
        }
    }
}

```

$(i, j)$  এর বিন্দু  
 নির্দেশ করে।  
 $i$  পরিশোধ  
 করা হচ্ছে।  
 $j$  পরিশোধ  
 করা হচ্ছে।  
 $(n - i - 1)$  কাজ করা হচ্ছে।  
 এটা মানে আরেকটি পরিশোধ করা হচ্ছে।  
 $A[j+1]$  এর দুর্বল  
 কাজ।

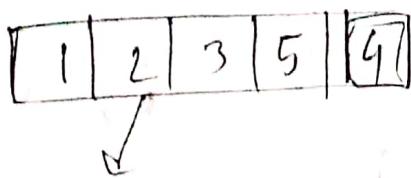
```

int main() {
    int i, j, +
    int A[] = {10, 5, 2, 3, 23};
    int n = sizeof(A) / sizeof(A[0]);
    cout << "Before Sorting array elements are : " << n;
    print(A, n);
    bubble(A, n);
    cout << "After sorting array
elements are : " << n;
    print(A, n);
    return 0;
}

```

## Insertion Sort

(Practical life & useable)



एक सॉर्टेड अर्रे (0 तक सॉर्ट होना चाहिए) का रूप  
यहाँ दिया गया है।

1 2 3 5 4

1 2 3 5 4

1 2 3 5 4

1 2 3 5 4

1 2 3 4 5

यहाँ, अलग रूप से सॉर्ट होना होता है, किन्तु इस सॉर्ट में,

5	4	2	3	1
(index)	0	1	2	3 4

i = 1 करने के बाद 2 का

j = i - 1 करने के बाद 1 का

दूसरा 2 का

```

#include <iostream>
using namespace std;

void insertion_sort(int A[], int n)
{
    int i, j, item;
    for (i = 1; i < n; i++) {
        item = A[i];
        j = i - 1;
        while (j >= 0 && A[j] > item) {
            A[j + 1] = A[j];
            j = j - 1;
        }
        A[j + 1] = item;
    }
}

void printArr(int A[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << A[i] << " ";
}

int main()
{
    int A[] = {1, 2, 3, 5, 4};
    int n = sizeof(A) / sizeof(A[0]);
    cout << "Before sorting array elements are : " << endl;
    printArr(A, n);
    insert(A, n);
    cout << "After sorting array elements are : " << endl;
    printArr(A, n);
    return 0;
}

```

## Merge Sort

6, 5, 37, 46, 44, 43, 50, 48, 19, 15, 4, 3

Divide into two halves (2x2),  
then 2x2 into 4x1, then next 2x2 into 2x1 then  
each smaller 2x1, until no more elements remain  
8x1 number 2x1 sort 2x1 Continue 2x1

```
#include<iostream>
using namespace std;
```

```
void merge(int A[], int left, int mid, int right)
```

```
{ int i, j, k;
int index_a, index_l, index_r;
int size_left, size_right;
```

```
size_left = mid-left + 1;
```

```
size_right = right - mid;
```

```
int L[size_left], R[size_right];
```

```
for ( i=0; i<size_left ; i++) {
```

```
    L[i] = A[left+i];
```

```
}
```

```

for (j=0; j<size_right; j++)
{
    R[j] = A[min+1+j];
}

i = 0;
j = 0;
k = left;

while (i < size_left && j < size_right)
{
    if (L[i] <= R[j])
    {
        A[k] = L[i];
        i++;
    }
    else
    {
        A[k] = R[j];
        j++;
    }
    k++;
}

while (i < size_left)
{
    A[k] = L[i];
    i++;
    k++;
}

```

```
while (i < size_right)
{
    A[k] = R[i];
    i++;
    k++;
}
```

```
void merge_sort(int A[], int left, int right)
{
    if (left >= right)
        return;
    }
```

```
int mid = left + (right - left) / 2;
```

[यदि यह int नंबर अपने int highest limit से ज्यादा हो तो इसका अवैध बन जाएगा]

```
merge_sort(A, left, mid);
```

```
merge_sort(A, mid+1, right);
```

```
merge(A, left, mid, right);
```

```
}
```

```

void PrintArr(int A[], int n)
{
    int i;
    for (i=0; i<n; i++)
        cout << A[i] << " ";
}

int main()
{
    int A[] = {11, 30, 34, 7, 24, 31, 16, 39, 41, 3};
    int n = sizeof(A)/sizeof(A[0]);
    cout << "Before sorting array elements are: " << n;
    PrintArr(A, n);
    merge_Sort(A, 0, n-1);
    cout << "After sorting array elements are: " << n;
    PrintArr(A, n);
    return 0;
}

```

## Quick Sort.

low

ମୋଟାରେ କୌଣସି ଦେଖନ୍ତୁ

high

9	4	3	11	15	20	2	24	30	1	35
---	---	---	----	----	----	---	----	----	---	----

Pivot

Quick Sort - ଏହା ପିଲାଗା ଏହାରେ Pivot ନିର୍ଦ୍ଦେଖନ କରିବା ହେଉଥିବା  
ଅନ୍ୟାନ୍ୟ ପିଲାଗା ଏହାରେ Pivot ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ  
ଅନ୍ୟାନ୍ୟ ପିଲାଗା ଏହାରେ, ଆବଶ୍ୟକ ଏହାରେ same procedure.

array'ର ମୂଳ ଏହାରେ ଏହାରେ ଏହାରେ Pivot ନିର୍ଦ୍ଦେଖନ କରି  
ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ  
ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ  
ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ ଏହାରେ

```
#include<iostream>
```

```
using namespace std;
```

```
int partition (int A[], int l, int h)
```

```
    int pivot = A[l];
```

```
    int i=l, j=h;
```

```
    while (i < j)
```

```
        { while (A[i] <= pivot)
```

```
            { i++;
```

```
}
```

[l = low, h = high]

```

while (A[j] > pivot)
{
    j--;
}
if (i < j)
{
    swap(A[i], A[j]);
}
swap(A[l], A[j]);
return j;
}

void quick(int A[], int l, int h)
{
    if (l < h)
    {
        int j = partition(A, l, h);
        quick(A, l, j - 1);
        quick(A, j + 1, h);
    }
}

```

$\boxed{\text{swap} = \text{खेल अपेक्षा } \Rightarrow \text{चुनौती इसी}}$   
Another form:  
 $\text{int } t = a[i];$   
 $a[i] = a[j];$   
 $a[j] = t;$

```
void printArr(int A[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << A[i] << " ";
}

int main()
{
    int A[] = {9, 4, 3, 11, 15, 20, 2, 24, 30, 1, 35};
    int n = sizeof(A)/sizeof(A[0]);
    cout << "Before sorting array elements are: \n";
    printArr(A, n);
    quick(A, 0, n - 1);
    cout << "\n After sorting array elements are: \n";
    printArr(A, n);
    return 0;
}
```

## Stack

It's work as a method LIFO (Last in, first out)

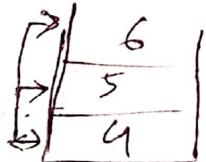


जैसे, C अंदरूनी इन 22मध्ये नोंद नाही तर 20 आणि 21मध्ये ।

operation:

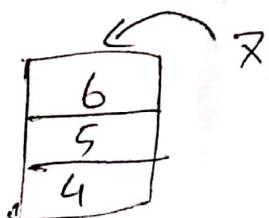
push()  
pop()  
top()

push - इनमध्ये top ठारावू,  
pop - " " top ठारावू,



→ यासे, data insert करताऱ्याचे झालेच आवडे top रुढा आवडा, तर  
प्रकारहूनी वापरा ।

overflow: Stack full राहील new element push करावू तर,



underflow: Stack 6 elements नाही, but pop करावू  
करावू underflow झाला ।

[Top = -1 means, Stack is empty.]

```

void push()
{
    if (top == max_stack - 1)
    {
        printf("overflow");
    }
    else
    {
        scanf("%d", &x);
        top++;
        stack[top] = x;
    }
}

```

max\_stack = 5  
[stack empty so, top = -1]

```

void pop()
{
    if (top == -1)
    {
        printf("underflow");
    }
    else
    {
        printf("%d", stack[top]);
        top--;
    }
}

```

[pop 2nd element  
print max]

```

void display()
{
    if (top == -1)
    {
        printf("Stack is empty");
    }
    else
    {
        for (i=0; i<=top; i++) → for (i=top, i>=0, i--)
        printf("%d", stack[i]);
    }
}

void push(int data)
{
    if (top == MAX)
    {
        printf("Stack overflow");
    }
    else
    {
        stack[++top] = data;
    }
}

int pop()
{
    if (top == -1)
    {
        printf("Stack underflow");
    }
    else
    {
        int data = stack[top];
        stack[top] = -1;
        return data;
    }
}

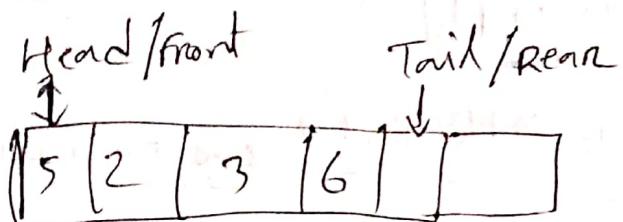
```

## Queue

Queue ହଳ୍ଟ ମାତ୍ରର କ୍ଷେତ୍ରରେ ଅନ୍ଧିତ ହଜାରୀ ।

ଲାଗୁ ହୋଇ ଥାଏଇ ନିତେ ଚାଲିବା ପାଇଁ କିମ୍ବା କିମ୍ବା କିମ୍ବା

ଦୁଇ ହଳ୍ଟ କ୍ଷେତ୍ରରେ ହେବାରେ ।

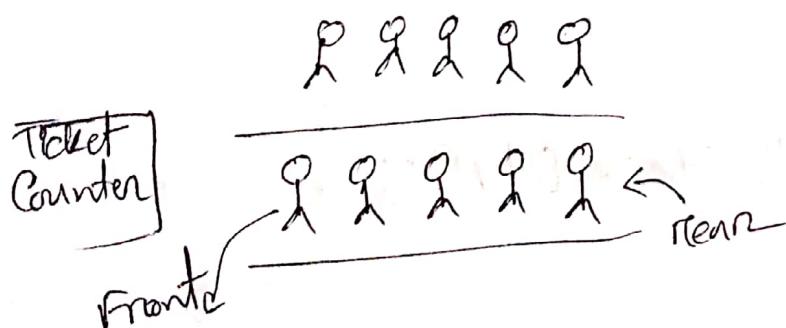


Queue କେ କିମ୍ବା କିମ୍ବା ENQUEUE ହାଲ୍ ,

queue କେ କିମ୍ବା କିମ୍ବା DEQUEUE ହାଲ୍ ,

### Queue:

A queue is another special kind of list, where items are inserted at one end called rear. and deleted at the other end called front. Another name for a queue is FIFO (first in first out)



(A)

front = rear = -1

[Queue is empty]

Front & rear same ~~same~~ index 0 zero ~~then~~ 20 queue to  
1 for element 20 1

void enqueue (int n)

{ if (rear == n-1)

{ printf ("overflow");

}

else if (front == -1 && rear == -1)

{ front = rear = 0;

queue [rear] = n;

else

{ rear++;

queue [rear] = n;

}

}

```
void dequeue()
{
    if (front == -1 && rear == -1)
        printf("Queue is empty");
    else if (front == rear)
    {
        printf("%d", queue[front]);
        front = rear = -1;
    }
    else
    {
        printf("%d", queue[front]);
        front++;
    }
}
```

```
void display()
{
    int i;
    if (front == -1 && rear == -1)
        printf("Queue is empty");
    else
    {
        for (i = front; i < rear; i++)
            printf("%d ", queue[i]);
    }
}
```

else

{ for(i=front; i < rear; i++)

{ printf("%d", queue[i]);

}

}

}

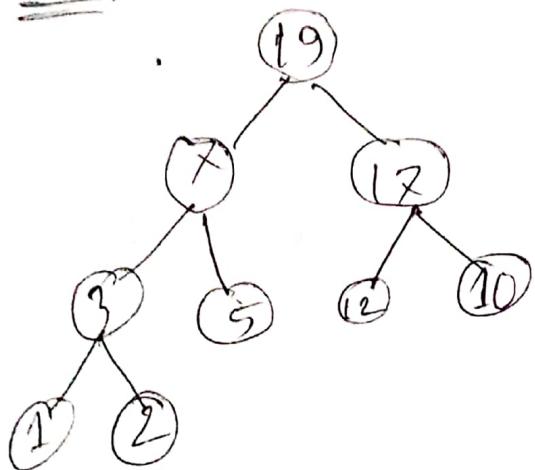
## Heap Sort

Heap sort is called complete binary tree.

(1) Max heap: means Root has child 19 & 20,

(2) Min heap: means Root has child 1 & 2,

Ex:



int heap\_size = 9;

int heap[] = { 0, 19, 7, 12, 3, 5, 10, 1, 2 };

heap[1] = 19;

heap[2] = 7;

heap[3] = 12;

heap[2\*2] = 3;

heap[2\*2 + 1] = 5;

heap[3\*2] = 12;

heap[3\*2 + 1] = 10;

heap[4\*2] = 1;

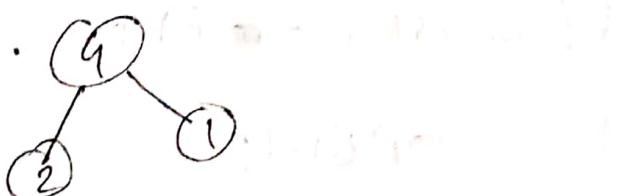
heap[4\*2 + 1] = 2;

left child  $\Rightarrow$   $index \times 2$

index \* 2 = left

index \* 2 + 1 = right

index / 2 = Parent



```
#include <iostream>
```

```
using namespace std;
```

```
void max_heapsify(int
```

```
heap[], int heap_size, int i)
```

```
{ int l, r, largest, t;
```

```
largest = i;
```

```
l = left(i); 2*i + 1;
```

```
r = right(i); 2*i + 1;
```

```
if (l < heap_size &&
```

```
largest = l;
```

```
}
```

```
else {
```

```
largest = i; if (r <
```

```
if (r < heap_size &&
```

```
largest = r;
```

```
}
```

```
heap[l] > heap[i]) {
```

```
temp = heap[i];
```

```
heap[i] = heap[l];
```

```
heap[l] = temp;
```

```
heap[r] > heap[largest]) {
```

```
temp = heap[i];
```

```
heap[i] = heap[r];
```

```
heap[r] = temp;
```

if (largest != i) {

    t = heap[i];

    heap[i] = heap[largest];

    heap[largest] = t;

    max\_heapify(heap, heap\_size, largest);

}

}

void heapSort (int heap[], int heapSize)

{ int i, temp;

    for (i = heap\_size; i > 1; i--) {

        t = heap[1];

        heap[1] = heap[i];

        heap[i] = t;

        heap\_size -= 1;

        max\_heapify(heap, heap\_size, 1);

}

    for (int i = heap\_size / 2 - 1; i >= 0; i--) {

        max\_heapify(heap, heap\_size, i);

    for (int i = heap\_size - 1; i >= 0; i--) {

        int temp = heap[0];

        heap[0] = heap[i];

        heap[i] = temp;

        max\_heapify(heap, i, 0);

}

}

```
void PrintArr(int arr[], heap[], int heap_size)
```

```
{ for (int i=0; i<heap_size; i++)
```

```
{ cout << heap[i] << " ";
```

```
}
```

```
}
```

```
int main()
```

```
{ int heap[] = { 48, 9, 22, 42, 27, 25, 0 };
```

```
int heap_size = sizeof(heap) / sizeof(heap[0]);
```

```
cout << "Before sorting array elements are : \n";
```

```
PrintArr(heap, heap_size);
```

```
heap_sort(heap, heap_size);
```

```
cout << "\n After sorting array elements are : \n";
```

```
PrintArr(heap, heap_size);
```

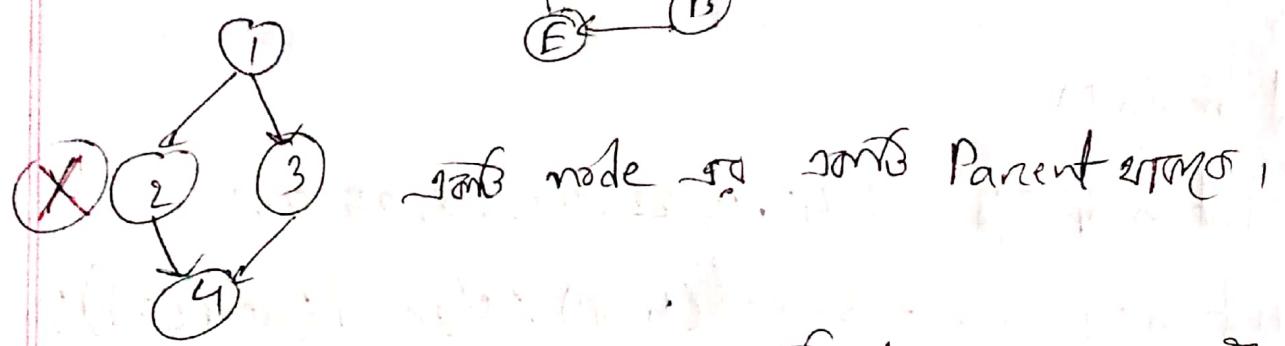
```
return 0;
```

```
}
```

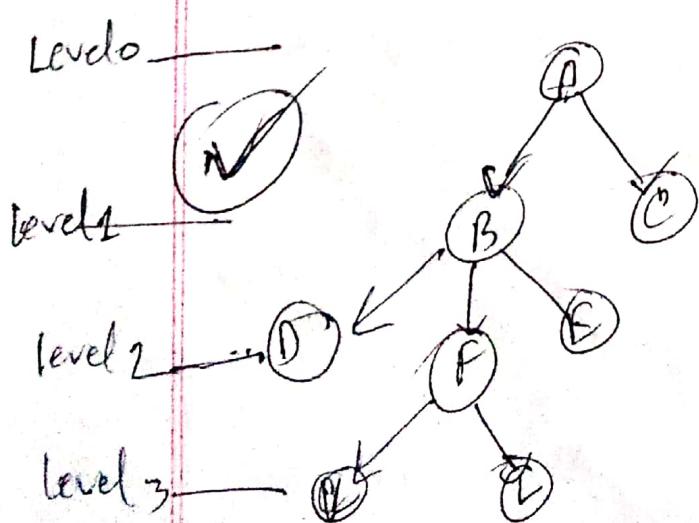
# Tree

~~leaf node~~: पर नोड से कम चाहे भी बच्चे नहीं हो।

② Cycle एवं tree विभीत



non-binary tree विभीत एवं binary tree विभीत विभीत



Degree of A is 2  
n = 2, B = 3

Height of A is 3

n = 2, B = 2

n = 2, F = 1

n = 2, K = 0

Depth of A is 0

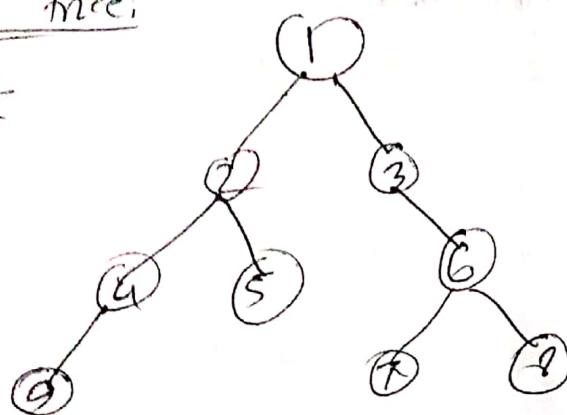
n = 2, B = 1  
n = 2, F = 2

## Binary tree

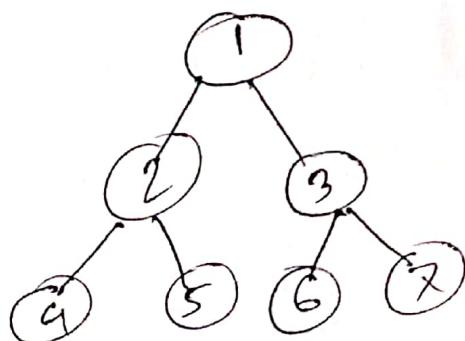
Parent  $\rightarrow$  child refers to  $\leftarrow$   $\text{P} \rightarrow \text{C}$

### Binary tree:

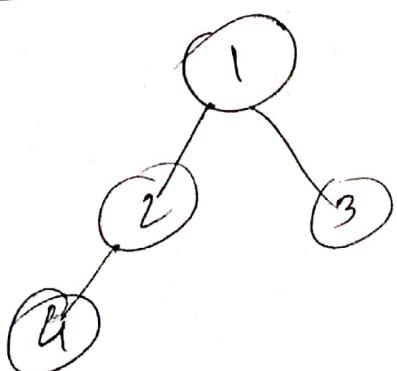
Full



full Binary tree:  $\forall$  node full ~~leaf~~  $\Rightarrow$  all node ~~leaf~~  
 $\rightarrow$  child node  $\forall$ .



Complete binary tree:  $\forall$  non full leaf must.



Binary Search Tree: Root  $\rightarrow$  8th element (B) 20.

$\rightarrow$  20 21 23 270 Root  $\rightarrow$  2nd

## Radix Sort

અંગેને અંગે એ વિધિઓ હોય, બાકીનો જુદ્ધ હિંદુ-તરફાની પૂર્વાની રૂપીત હોય।

જોણે, એકું કાર્ય કરે જાય એનોટ એવે વિધિ નિયે કાઢ કરીએ  
હોય। તૈયાની:

97, 58, 208, 699, 125, 234

∴ 097, 058, 208, 699, 125, 234

A = 

097	058	208	699	125	234
0	1	2	3	4	5

અંગે દાખે રૂપીઠા, કાર્ય, 3 4 5 6 7 8 9

Count = 

0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

A[], એ રોકડ સ્થાનાનું ગમ હોય આજ રીતે,

Count = 

0	2	3	4	5	6	7	8	9	
0	0	0	0	1	1	0	12	1	1

 Left to Right

After modifying (by adding)

Count = 

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

output = 

0	1	2	3	4	5
734	125	097	058	208	699

 Right to left

દાખે સુધીનો કાર્ય

Count = 

0	1	2	3	4	5	6	7	8	9
1	0	1	1	0	1	0	0	0	12

Count =

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

output =

0	1	2	3	4	5
208	125	734	052	092	699

Digit Counting Method

Count =

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

Count =

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

output =

0	1	2	3	4	5
052	092	125	208	699	734

Finally sorted

time complexity:  $O(d * (n+k))$

[ $d$  = digit  $\Rightarrow$  10  
বিশেষ করে 02]

```

#include <iostream>
using namespace std;

int getMax (int a[], int n)
{
    int max = a[0];
    for (int i=1; i<n; i++) {
        if (a[i] > max)
            max = a[i];
    }
    return max;
}

void CountSort (int a[], int n, int pos)
{
    int output[n];
    int count[10] = {0};

    for (int i=0; i<n; i++)
        count[(a[i]/pos)%10]++;
    for (int i=1; i<10; i++)
        count[i] = count[i] + count[i-1];
    // count[i] += count[i-1];
    for (int i=n-1; i>=0; i--)
    {
        output[count[(a[i]/pos)%10]-1] = a[i];
        count[(a[i]/pos)%10]--;
    }
}

```

```

for (int i = 0; i < n; i++) {
    a[i] = output[i];
}

void radixsort (int a[], int n) {
    int max = getMax(a, n);
    for (int pos = 1; max / pos > 0; pos *= 10)
        CountSort(a, n, pos);
}

void printArray (int a[], int n) {
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
}

int main () {
    int a[] = { 171, 289, 380, 111, 155, 504, 112 };
    int n = sizeof(a) / sizeof(a[0]);
    cout << "Before Sorting: " << endl;
    printArray(a, n);
    radixsort(a, n);
    cout << "After Sorting: " << endl;
    printArray(a, n);
    return 0;
}

```

## Huffman Coding

A B B C D B C C D A A B B E E E B E A B = 20  
and calculate -

- characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E  
character 6 is normally ascii value store 20 P.C. 20

A → 65 = 01000001 = 8 bits

B → 66

C → -

D → -

20 fb character → 0, 1 (20 × 3) = 160 bits

But, individually character need 5 fb, so short form

→ represent 0, 1, 2<sup>n</sup>, 2<sup>n</sup> and 5 fb use 20, 3 fb unused 20 - 2<sup>2</sup> = 4 bits left over

Q 3 fb and unused, so 20 bits remaining, (2<sup>n</sup>) 80 bits,

0 0 0 - A  $\frac{2^3 = 8}{}$

0 0 1 - B

0 1 0 - C

0 1 1 - D

1 0 0 - E

1 0 1 - F

1 1 0 - G

1 1 1 - H

→ 3 bits for 3-character represent over/ encode over,

unused -

comprise 275,

$20 \times 3 = 60$  bits

21) A receiver receives 23 message bits, want to encode  
decade wise at 2000 bits/sec  $n = 8$ ,  $m = 3$  etc. OR  
msg  $\Rightarrow$  input to table output 270.

\* msg  $\Rightarrow$  bits Compress/encode bits;  $28 \times 3$   
 $= 60$  bits

table  $\Rightarrow$  Q,

A, B, C ... Character  $\Rightarrow$  Q  $= 5 \times 3 = 15$

[Each character = 8 bits]

A, B, C ... character  $\Rightarrow$  Compress/encode each (3 bits)  $\Rightarrow$  Q  
customized

$$5 \times 3 = 15$$

$$\therefore 40 + 15 = 55 \text{ bits}$$

$\therefore 60 + 55 = 115$  bits [with table etc  
all thing complete  
msg]

variable

A B B C D B C C D A A B B F F F E B E A B

variable length coding ; means. how many same character is exist.

বার্য ক্রিমেন্স এবং অর্থ সম্পর্ক

Char	Frequency Count	Million order
A	4	01
B	7	11
C	3	101
D	2	100
E	4	00

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

২০১

(222) Fixed bits (2), ~~for length of msg~~

$$4 \times 2 + 2 \times 2 + 3 \times 3 + 2 \times 3 + 4 \times 2$$

$$\therefore r = 45 \text{ bits}$$

For character and huffman code:

$$5 \times 8 \xrightarrow{\text{Ascci code}} = 40$$

$$40 + 12$$

$$= 52 \text{ bits}$$

$$\therefore 45 + 52 = 97 \text{ bits}$$

Prefix code :- no code is Prefix of another code.

Because it's give different value.

~~Ex:-~~  $A \leftarrow 0$   
 $B \leftarrow 1$   
 $C \leftarrow 01$

