SELECTION SORT

```
Selection sort (n)

{
      For i= 1 to n-1
         min=i;
         For  j=i+1 to n
            If (a[j] < a[min]) then set min =j;
         If (i!= min) swap a[i] and a[min]
}
```

Bubble SORT

```
Bubble sort (n)

{
        For i= 1 to n-1

            For  j=1 to n-i
                If (a[j+1] < a[j]) then  swap a[j] and a[j+1]
}
```

INSERTION SORT

```
Insertion sort (n)

{
        For i= 2 to n
            v=a[i]; j=i-1;
            while (j>0 and v<a[j])
                    a[j+1] = a[j]) ;
                    j=j-1;
             a[j+1]=v;
}
```

CREATING HEAP – ALGORITHM 1- SLOW

```
Slow Heap (a,n)
{
        for (j=2 to n)
        {
        item = a[j];i=j;
        while ((i>1) and (a[i/2] < item))
        {
                a[i]=a[i/2]; i=i/2;
        }
        a[i]= item;
        }
}
```

CREATING HEAP – ALGORITHM 2- FAST

```
Fast Heap (A,n)
{
    for i=n/2 to 1
        Modify (a,i,n);
}
```

```
Modify (A,i,n)
{
    J=2i; item =A[i];
    While (j<=n)
    {
                if((j<n) and (A[j]<A[j+1])) j++;

                if (item>= A[j])) break;

                A[j/2]=A[j]; j=2*j;
    }
    A[j/2]=item;
    }
```

HEAP SORT

```
• Heapsort (A,n)
• {
    FastHeap (A,n)
    for i=n to 2
      {
              swap (A[i], A[1]);
              Modify (A,1,i-1);
      }
   }
```

LOWER BOUND ON SORTING

Sorting is made of comparisons

Total n! Permutations

Each comparison halves the number of possible permutations.

Log (n!) comparisons needed

$(n/2)^{(n/2)} < n! < n^n$

$(n/2)\ \mathrm{Log}\ (n/2) < \log n! < n \log n$