30/8

1st assumption

→ Array is sorted

1	7	8	13	25	27	29	351

l   int element = 1   h.

scanf("%d", &element);

a[(l+h)/2]

while (l < h)
{
mid = (l+h)/2

If (element == a[mid])
{
printf("%d", mid)
break;
}
If (element < a[mid])
h = mid-1;
else
l = mid+1;
}

→ h = (l+h)/2

OR

→ l = (l+h)/2

→ printf("%d" is in %d position", element, mid);

```
for(i=0; i<n; i++)
  if (element==a[i]).
    printf("Found");
```

$T_{1000} = T_{500} + 1$

$T_k = T_{k/2} + 1$ | $T_k = c_1 \log_2 k + c_2$

Suppose $k = 2^p$

$T_{k/2} = T_{k/4} + 1$

$T_{k/4} = T_{k/8} + 1$

.

$T_{k/2^p} = T(1) = 1$

$T_k = p$

$= \log_2 k.$

8	2	17	6	11	13

	8	17	6	11	13

→ Find the lowest

2	8	17	6	11	13

2	6	8	17	11	13

2	6	8	11	17	13

2	6	8	11	13	17

TASK 1 → Find the smallest, and its location

TASK 2 : Push till the smallest

Task 3 : put smallest in the beginning.

```
main().
{ scanf("%d",&n)
  for(i=0;i<n;i++).
    scanf("%d",&a[i]);
for(j=0; j<n-1; j++
{  min=a[j]; minl=j;
   for(i=j+1; i<n; i++)
   {
     if(a[i]<min)
     { min=a[i]; minl=i;
     }
   }
   for(i=minl-1; i>=j; i--)
      a[i+1]=a[i];
   a[j]=min;
}
for(i=0;i<n;i++).
  printf("%d",a[i]);
```

$C_1 n$

$C_2 n$

1

$T_n = C_1 n + C_2 + T_{n-1}$

$C_1 n + C_2$ $\quad \cancel{C_x n + C_y}$

$T_n = T_{n-1} + C_1 n + C_{21}$

$T_{n-1} = T_{n-2} + C_1(n-1) + C_{22}$

$\vdots$

$T_n = C_1(n + (n-1) \cdots + 1) + C$

$= C_1 \left(\frac{n(n+1)}{2}\right) + C.$

$$\boxed{T_n = O(n^2).}$$

$f(x) = a_1 x^n + a_2 x^{n-1} \cdots + a_{n+1}$

$f(x) = O(x^n).$

k

Sort' Search $= O(n^2) + {}_{k}\, O(\log n) \approx O(\log n)$

Linear Search $= O(n)$