Use recurrences to show that binary search on a sorted array runs in $O(\log n)$ time.

**Pseudo code**

```
for a low and high:
    calculate mid
    if toFind > mid:
        R == recurse on 2nd half
    else:
        L == recurse on first half
    if toFind == mid:
        return true
```

$T(n) = O(1) + T(n/2)$

$T(n) = O(1) + T(n/2)$

$= O(1) + O(1) + T(n/2)$

$= O(1) + O(1) + O(1) + T(n/8)$

$= \log_2 n \cdot O(1)$

$T(n) = O(\log n)$
Provide two topological orderings of the following graph. Show steps.

```
A B C D E F G H I J K L
∅ ∅ ∅ ∅ ∅ ∅ ∅ ∅ ∅ ∅ ∅ ∅
6 6 6 6 6 6 6 6 6 6 6 6

ABDI, C, E, FGH, K, L

JL
```
Starting with an array of size 7, insert the following elements into a hash table using linked-list chaining.

Use $k = 7$ as the hash function.

51, 14, 91, 16, 3, 11, 28, 2, 23, 37

```
0 → 14 → 91 → 28
1
2 → 51 → 16 → 2 → 23 → 37
3
4
5
6
```

Discuss any interesting results as far as runtime.
Is a resize advisable at any point?

$5/9$ elements are in index 2.

$O(n)$ find we

Resize: recognize that 2 is overloaded
Use Dijkstra's algorithm to find the shortest path from A to D. Show intermediate steps.

<table>
<thead>
<tr>
<th>Known Path</th>
<th>Previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
</tr>
</tbody>
</table>

(A, B), (B, C), (C, D)

Cost: 3
Provide the MST using Kruskal's algorithm.

\begin{align*}
A, D & 1 & \checkmark \\
D, G & 1 & \checkmark \\
B, E & 1 & \checkmark \\
E, H & 1 & \checkmark \\
D, E & 2 & \checkmark \\
B, C & 2 & \checkmark \\
G, H & 3 & \times \quad D, G, E, H \\
H, I & 3 & \checkmark \\
A, B & 4 & \times \quad A, B, D, E \\
E, F & 4 & \checkmark \\
C, F & 5 & \\
F, I & 5 & \checkmark \\
\end{align*}
Provide the MST for this graph using Prim's algorithm. Show steps. Indicate if multiple MSTs may exist and why.

Select A

<table>
<thead>
<tr>
<th>K</th>
<th>edge</th>
<th>prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>A, E</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>3</td>
<td>H</td>
</tr>
</tbody>
</table>

Steps:
1. Select A
2. Add B (5, A, E)
3. Add C (2, B)
4. Add D (1, A)
5. Add E (2, D)
6. Add F (4, H)
7. Add G (3, D)
8. Add H (2, I)

Total cost: 16
Use the Quicksort algorithm of 3 approach to sort the following list.
Show steps.
Show the u/free after the following operations.
Use weight, union and path compression.
union(1, 5)
union(6, 2)
union(4, 5)
union(1, 2)
find(6)
find(3)
union(3, 2)
find(1)

In case of ties, let the first argument of union be the representative.
Design an algorithm which determines whether 2 integers share any common factors.

Recall that if two numbers are prime and not equal, they share no common factors.

Also recall that if a number n is composite (non-prime), it must have a factor k ≤ √n.

All numbers have a unique prime factorization.

j, k are the numbers

if j % p and k % p = 0, then
they have a common factor p

if j = k,
check all p from 2 to k

check all p from 2 to √k