1. **Show that any comparison based algorithm to sort 4 elements requires atleast 5 comparisons in the worst case.**

 Number of possible permutations for 4 elements: 4! = 24

 Decision tree must have at least 24 leaves

 Let h = height of the decision tree

 Equation: 2^h ≥ 24

 Take log₂ of both sides: h ≥ log₂ (24)

 Calculate: log₂(24) ≈ 4.58

 Round up: h = 5

1. **Explain how RadixSort can be used to sort the following s ={125, 27, 729, 1,27, 8, 64,343, 216} using radix =9.**

**Radix Sort Solution: Sorting in Base 9**

**a. Radix Definition**

* Radix = 9 (base-9 number system)
* In base 9, we use digits 0-8 instead of the standard 0-9
* Conversion process involves repeatedly dividing by 9 and tracking remainders

**b. Base 9 Conversion Process and Digit Analysis**

| **Original Number** | **Conversion Process** | **Base 9 Representation** | **Number of Digits** |
| --- | --- | --- | --- |
| 125 | 125 ÷ 9 = 13 remainder 8  13 ÷ 9 = 1 remainder 4  1 ÷ 9 = 0 remainder 1 | 148 | 3 |
| 27 | 27 ÷ 9 = 3 remainder 0  3 ÷ 9 = 0 remainder 3 | 30 | 2 |
| 729 | 729 ÷ 9 = 81 remainder 0  81 ÷ 9 = 9 remainder 0  9 ÷ 9 = 1 remainder 0  1 ÷ 9 = 0 remainder 1 | 1000 | 4 |
| 1 | 1 (no division needed) | 1 | 1 |
| 8 | 8 (no division needed) | 8 | 1 |
| 64 | 64 ÷ 9 = 7 remainder 1  7 ÷ 9 = 0 remainder 7 | 71 | 2 |
| 343 | 343 ÷ 9 = 38 remainder 1  38 ÷ 9 = 4 remainder 2  4 ÷ 9 = 0 remainder 4 | 421 | 3 |
| 216 | 216 ÷ 9 = 24 remainder 0  24 ÷ 9 = 2 remainder 6  2 ÷ 9 = 0 remainder 2 | 260 | 3 |

**Key Observations**

* Maximum number of digits in base 9: 4 digits
* We'll pad shorter numbers with leading zeros during sorting

**c. Radix Sort Detailed Walkthrough**

**Initial Set**

* Original Numbers: {125, 27, 729, 1, 8, 64, 343, 216}
* Base 9 Representation: {148, 30, 1000, 1, 8, 71, 421, 260}

**Sorting Strategy**

* Use Least Significant Digit (LSD) Radix Sort
* Perform 4 passes, one for each digit position
* Use Counting Sort for each pass

**Pass 1: Sort by Least Significant Digit**

* Focuses on rightmost digit
* Sorted result: {30, 1000, 30, 260, 1, 71, 421, 148, 8}

**Pass 2: Sort by Second Digit from Right**

* Examines second digit
* Sorted result: {1000, 1, 8, 421, 30, 30, 148, 260, 71}

**Pass 3: Sort by Third Digit from Right**

* Focuses on third digit
* Sorted result: {1000, 1, 8, 30, 30, 71, 148, 260, 421}

**Pass 4: Sort by Fourth Digit from Right**

* Uses most significant digit
* Final sorted result: {1, 8, 30, 30, 71, 148, 260, 421, 1000}

**Final Result**

* Original Sorted Order: {1, 8, 27, 27, 64, 125, 216, 343, 729}

**Complexity Analysis**

* Time Complexity: O(d \* (n + k)), where:
  + d = number of digits
  + n = number of elements
  + k = radix (9 in this case)
* Space Complexity: O(n + k)

1. **Carry out the steps of RadixSort to sort the following {80, 27, 72, 1, 27, 8, 64, 34, 16}. Use 9 as your radix.**

**Initial Set**

Input Set: {80, 27, 72, 1, 27, 8, 64, 34, 16}

**Base 9 Conversion Process**

**Division and Remainder Tracking**

| **Number** | **1st Division** | **2nd Division** | **3rd Division** | **Remainder from Bottom** |
| --- | --- | --- | --- | --- |
| 80 | 8 remainder 8 | 0 remainder 8 | - | 88 |
| 27 | 3 remainder 0 | 0 remainder 3 | - | 30 |
| 72 | 8 remainder 0 | 0 remainder 8 | - | 80 |
| 1 | 0 remainder 1 | - | - | 1 |
| 27 | 3 remainder 0 | 0 remainder 3 | - | 30 |
| 8 | 0 remainder 8 | - | - | 8 |
| 64 | 7 remainder 1 | 0 remainder 7 | - | 71 |
| 34 | 3 remainder 7 | 0 remainder 3 | - | 37 |
| 16 | 1 remainder 7 | 0 remainder 1 | - | 17 |

**Initial Representation**

{88, 30, 80, 1, 30, 8, 71, 37, 17}

**Sorting Process**

Two passes due to maximum divisions

**First Pass**

Representation: {30, 80, 30, 1, 71, 37, 17, 88} Original: {27, 72, 27, 1, 64, 34, 16, 80}

**Second Pass**

Representation: {1, 17, 30, 30, 37, 71, 80, 88} Original: {1, 16, 27, 27, 34, 64, 72, 80}

**Final Sorted Order**

{1, 16, 27, 27, 34, 64, 72, 80}