2. Input String: EASY QUESTION

QuickSort:

E A I E E N U Y S T S O &
E A E E
A E E
O O S T S U Y
O
S T S U Y
S T S O
S S T
S
T

The sorting result is: A E E I N O & S S T U Y
1. Version 1:

   Input story:

   EASY QUESTION

2. E A E I

3. E A E I

4. O E

5. E

6. U S S T Y O G

The final OTP is:

A E E I N O Q S S T U Y
7.6: Example of 6 files in the case where Quicksort gives worst-case performance.

Ex-1:
9 8 7 6 5 4 3 2 1 10
    (Am)

Ex-2:
8 7 6 5 4 3 2 1 9 10

Ex-3:
7 6 5 4 3 2 1 8 9 10

Ex-4:
6 5 4 3 2 1 7 8 9 10

Ex-5:
5 4 3 2 1 6 7 8 9 10

Ex-6:
4 3 2 1 5 6 7 8 9 10

(Quicksort shows worst case behavior when the partitioning element resides on the extreme sides of the list. (Am)}
LSD Radix Sort

Keywords

1st leading place

now
is
the
time
for
all
good
people
to
come
the
aid
of
their
party

2nd leading place

party
people
of
the
the
the
aid
time.

of
people
party
time
to
the
their.


(Rev).
Returns no of nodes in a circular linked list.

Method:

```c
int count_node (Node p) {
    int count = 0;
    Node start = p;
    while (p->next != start)
        count ++;
    p = p->next;
    return count;
}
```
3.26. Code fragment that determines the number of nodes that are between two given references \( x \) and \( t \).

```c
int count_nodes(node x, node t)
{
    int count = 0;
    while (x.next != t)
    {
        count++;
        x = x.next;
    }
    return count;
}
```
void list-insert (node t, node x)
{
    Node q = x. next;
    Node p = t. next;
    x. next = p;
    t. next = q;
}

This entire insertion
takes \( \Theta(1) \) time.
class IntStack
{
private int [] S;
private int N;
inStack(int maxN)
{ S = new int [maxN]; N = 0; }
int Count()
{ return N; }
void push(int item)
{ S[N++] = item; }
int pop()
{ return S[--N]; }
}

Linked List
class IntStack
{ private node head;
private class node
{ int item;
node next;
}
int Count()
{ private Count;
while (head.next != null)
Linked List representation:

```java
class intStack
{
    private Node head;
    private class Node
    {
        // Node definition
    }
    intStack(int maxN)
    {
        int count = 0;
        while (head.next != null)
        {
            head = head.next;
            count ++;
        }
        return count;
    }
    void push(int item)
    {
        // Push logic
    }
    int pop()
    {
        // Pop logic
    }
}
```
5.9. Convert to postfix expression:

\[(5 \ast ((9 \ast 8) + ((7 \ast (4 + 6))))\]

<table>
<thead>
<tr>
<th>Input</th>
<th>Output stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>8</td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>7</td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>6</td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>\ast</td>
<td></td>
</tr>
</tbody>
</table>

\[5 \ast (9 \ast 8) + (7 \ast (4 + 6))\]
Prefix Evaluation (showing stack content):

<table>
<thead>
<tr>
<th>Taken</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5 9</td>
</tr>
<tr>
<td>*</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>45 8</td>
</tr>
<tr>
<td>7</td>
<td>45 8 7</td>
</tr>
<tr>
<td>4</td>
<td>45 8 7 4</td>
</tr>
<tr>
<td>+</td>
<td>45 8 7 10</td>
</tr>
<tr>
<td>*</td>
<td>45 8 70</td>
</tr>
<tr>
<td>2</td>
<td>45 8 70 2</td>
</tr>
<tr>
<td>1</td>
<td>45 8 70 2 1</td>
</tr>
<tr>
<td>3</td>
<td>45 8 70 2 3</td>
</tr>
<tr>
<td>*</td>
<td>45 8 70 5</td>
</tr>
<tr>
<td>+</td>
<td>45 358</td>
</tr>
<tr>
<td>*</td>
<td>16110 (Answer)</td>
</tr>
</tbody>
</table>
Initially empty queue.

1st get (E) → Return E
2nd n (A) → n E
3rd x (S) → n A
4th n (Y) → n S
5th n (Q) → n Y
6th x (U) → n Q
7th x (E) → n U
8th x (T) → n E
9th n (O) → n T
10th n (I) → n O
11th n (O) → n I
12th n (N) → n O
13th n (?) → n N, (Ann).
14.31. EASY SOLUTION

\[ M = 16 \]

Hash function 1 = \( 11K \mod M \)
\[ n_2 = (K \mod 3) + 1. \]

Hash Table

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Y</td>
<td>T</td>
<td>Q</td>
<td>E</td>
<td>U</td>
<td>N</td>
<td>A</td>
<td>J</td>
<td>U</td>
<td>O</td>
<td>I</td>
<td>V</td>
<td>F</td>
<td>T</td>
<td>I</td>
</tr>
</tbody>
</table>

Input symbol

\[ E = H_f(E) \]
\[ = 11K \mod 16 \]
\[ = 11 \times 5 \mod 16 \]
\[ = 55 \mod 16 = 7 \]

\[ A = H_f(A) = 11 \mod 16 = 11 \]
\[ S = H_f(S) = 11 \times 19 \mod 16 = 1 \]
\[ Y = H_f(Y) = 25 \times 11 \mod 16 \]
\[ = 11 \mod 16 = 11 \] (collision)
\[ Q = H_f(Q) = (17 \mod 3) + 1 \]
\[ = 3 \] (probe needed)
\[ \] (instead of \( n = 14 \))

\[ U = H_f(U) = 18 \mod 7 (collision) \]
\[ V = H_f(V) = (21 \mod 3) + 1 = 1 \]
\[ T = H_f(T) = 12 \]

I = H_f(I) = 3 (collision)
\[ H_f(2(I) = 1 \]
\[ \] (I goes to \( 3 + 1 = 4 \) locates \( T = 12 \))

\[ \theta = H_f(\theta) = 5 \]
\[ N = H_f(\theta N) = 10 \]
\[ H_f^1 = 11k \mod M \quad \text{(initial probe)} \]
\[ H_f^2 = (k \mod 3) + 1 \quad \text{(increment)} \]
\[ M = 10 \]

**E A S Y S U T I O N**

\[ H_f^1(E) = 11 \times 5 \mod 10 = 5 \]

\[ H_f^1(A) = 11 \mod 10 = 1 \]

\[ H_f^1(S) = 19 \times 11 \mod 10 = 9 \quad \text{(collision)} \]

\[ H_f^1(Y) = 25 \times 11 \mod 10 = 5 \]

\[ H_f^2(Y) = (25 \mod 3) + 1 = 2 \]

\[ Y \text{ goes to } 5 + 2 = 7 \text{ th location} \]

\[ H_f^1(G) = 7 \quad \text{(collision)} \]

\[ H_f^2(G) = 3 \]

\[ G \text{ goes to } 5 \rightarrow 0 \text{ th location} \]

\[ H_f^1(U) = 1 \quad \text{(collision)} \]

\[ H_f^2(U) = 1 \]

\[ U \text{ goes to } 1 + 1 = 2 \]

\[ H_f^1(I) = 99 \mod 10 = 9 \quad \text{(collision)} \]

\[ H_f^2(I) = 1 \]

\[ I \text{ goes finally to location 4} \]

\[ H_f^1(T) = 0 \quad \text{(collision)} \]

\[ H_f^2(T) = (20 \mod 3) + 1 = 3 \]

\[ H_f^1(O) = 15 \times 11 \mod 10 = 5 \quad \text{(collision)} \]

\[ H_f^2(O) = 1 \]

\[ O \text{ goes to } 5 \text{ th location} \]

\[ H_f^1(N) = 4 \quad \text{(collision)} \]

\[ H_f^2(N) = 3 \]

\[ N \text{ finally gets inserted at location} \]

8 after trying 7, 0, 3, 6, 9, 2, 5 positions
Node remove(Node h, Item v)
{
    if (h == null) return null;
    if (equals(h.item, v))
    {
        h.l = null;
        h.r = null;
    }

    if (h.l != null) remove(h.l, v);
    if (h.r != null) remove(h.r, v);
}

5.68 Total no. of nodes = NM + 1

\[ \text{No. of external nodes} = MN + 1 - N. \]

5.79

preorder: - DBACFEGI, CBADE, EBCDAHFGI
inorder: - ABCDEFGI, ABCDE, ABCDEFGHI
postorder: - ACBEGFD, ABEDC, ABEDCFIHE
level order: - DBFACEGFI, CBDAE, ECHBDFIAG
static int countLeaves (TreeNode node) {
    if (node == null) return 0;
    else if ((node.left == null) & (node.right == null))
        return 1;
    else
        return countLeaves (node.left) + countLeaves (node.right);
}

static int countChild (TreeNode node) {
    if (node == null) return 0;
    else if (((node.left == null) & (node.right != null))
         || (node.left != null) & (node.right == null))
        return 1;
    else
        return countChild (node.left) + countChild (node.right);
}
12.70

static int height (Treenode node)
{
    if (node == null) return 0;
    else if ((node.left == null) && (node.right == null))
        return 0;
    else
        return max (height (node.left), height (node.right)) + 1;
}

12.84

12.90
double edges are the 'red' edges.

Because every 3-node can give 2 possible orientations, we can have a total of $2^t$ combinations.

Answer does not depend on the order of inserting items.