perplexity

explain Linear and Binary Search algorithm and bubble sorting algorithm for array and linked list in C programming

Linear search, binary search, and bubble sort are fundamental algorithms with distinct implementations for arrays and linked lists in C. Here's a structured breakdown:

Linear Search

For Arrays
Sequentially checks each element until the target is found:

```
int linearSearch(int arr[], int n, int key) {
    for(int i=0; i<n; i++) {
        if(arr[i] == key)
            return i;
    }
    return -1;
}
```

- **Time Complexity**: O(n)[1]

For Linked Lists
Traverses nodes using pointers:

```
struct Node* LSearch(struct Node *p, int key) {
    while(p) {
        if(key == p->data)
            return p;
        p = p->next;
    }
    return NULL;
}
```

- Recursive version also exists[2]

Binary Search

Exclusive to Sorted Arrays
Divides search space by comparing with mid-point:

```
int binarySearch(int arr[], int left, int right, int key) {
    while(left <= right) {
        int mid = left + (right-left)/2;
        if(arr[mid] == key) return mid;
        if(arr[mid] < key) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
```

- **Time Complexity**: O(log n)[3]
- *Not viable for linked lists* due to lack of random access.

Bubble Sort

For Arrays

Repeatedly swaps adjacent elements:

```
void bubbleSort(int arr[], int n) {
    for(int i=0; i<n-1; i++) {
        for(int j=0; j<n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

- **Time Complexity**: O(n^2)[4]

For Linked Lists

Swaps nodes instead of data elements:

```
void swapNodes(struct Node **head, struct Node *a, struct Node *b) {
    if(a == b) return;

    // Find previous nodes
    struct Node *prevA = NULL, *prevB = NULL, *temp = *head;
    while(temp && temp != a) {
        prevA = temp;
        temp = temp->next;
    }
    temp = *head;
    while(temp && temp != b) {
        prevB = temp;
        temp = temp->next;
    }

    // Update pointers
    if(prevA) prevA->next = b;
    else *head = b;

    if(prevB) prevB->next = a;
    else *head = a;

    struct Node *tmp = b->next;
    b->next = a->next;
    a->next = tmp;
}

void bubbleSort(struct Node **head) {
    int swapped;
    struct Node *ptr1;
    struct Node *lptr = NULL;

    do {
        swapped = 0;
        ptr1 = *head;

        while(ptr1->next != lptr) {
            if(ptr1->data > ptr1->next->data) {
                swapNodes(head, ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while(swapped);
}
```

- Requires pointer manipulation for node swaps[5] [6]

Key Considerations

- **Linear Search**: Simple but inefficient for large datasets
- **Binary Search**: 60x faster than linear for 1M elements, but requires pre-sorted array
- **Bubble Sort**:
  - For arrays: Easy to implement but impractical for large data
  - For linked lists: Node swapping avoids data movement overhead

Use linear search for unsorted data, binary search for sorted arrays, and bubble sort only for small datasets or educational purposes. Linked list sorting introduces complexity due to pointer management.

⁂

1. https://www.programiz.com/dsa/linear-search
2. https://dotnettutorials.net/lesson/linear-search-in-linked-list/
3. https://www.w3schools.com/dsa/dsa_algo_binarysearch.php
4. https://www.scaler.com/topics/c-bubble-sort/

5. https://www.prepbytes.com/blog/linked-list/bubble-sort-for-linked-list-by-swapping-nodes/
6. https://www.prepbytes.com/blog/linked-list/c-program-for-performing-bubble-sort-on-linked-list/