DESIGN AND ANALYSIS OF ALGORITHMS

Tutorial - 3

Name - Deepti Pandey
Section - CST SPL1
Class Roll NO - 21
University Roll NO - 2016724

① Write linear search pseudocode to search an element in a sorted array with minimum comparisons.

```
for i = 1 ~~ton~~ n
    if a[i] == key
        print "Element found"
```

② Write pseudocode for iterative and recursive insertion sort. Insertion sort is called online sorting. Why? What about other sorting algorithms that have been discussed in lectures?

Iterative:

```
for i=1 to n-1
    t = A[i];
    j = i-1;
    while (j>=0 && A[j] < t)
    { A[j+1] = A[j];
      j--;
    }
    A[j+1] = t
```

Recursive:

```
    val = A[i];
    j = i;
    while (j> 0 && A[j-1]>val)
    { A[j] = A[j-1]; j--;
    }
    A[j] = val;
if (i+1 <= n)
{ insertionSort (A, i+1, n);
}
```

Insertion Sort is called online Sorting because it can sort an array as it receives it. An online Sort does not ~~[illegible]~~ know the whole input, insertion sort considers one element in each iteration and without considering the next elements.

~~③ Complexity~~

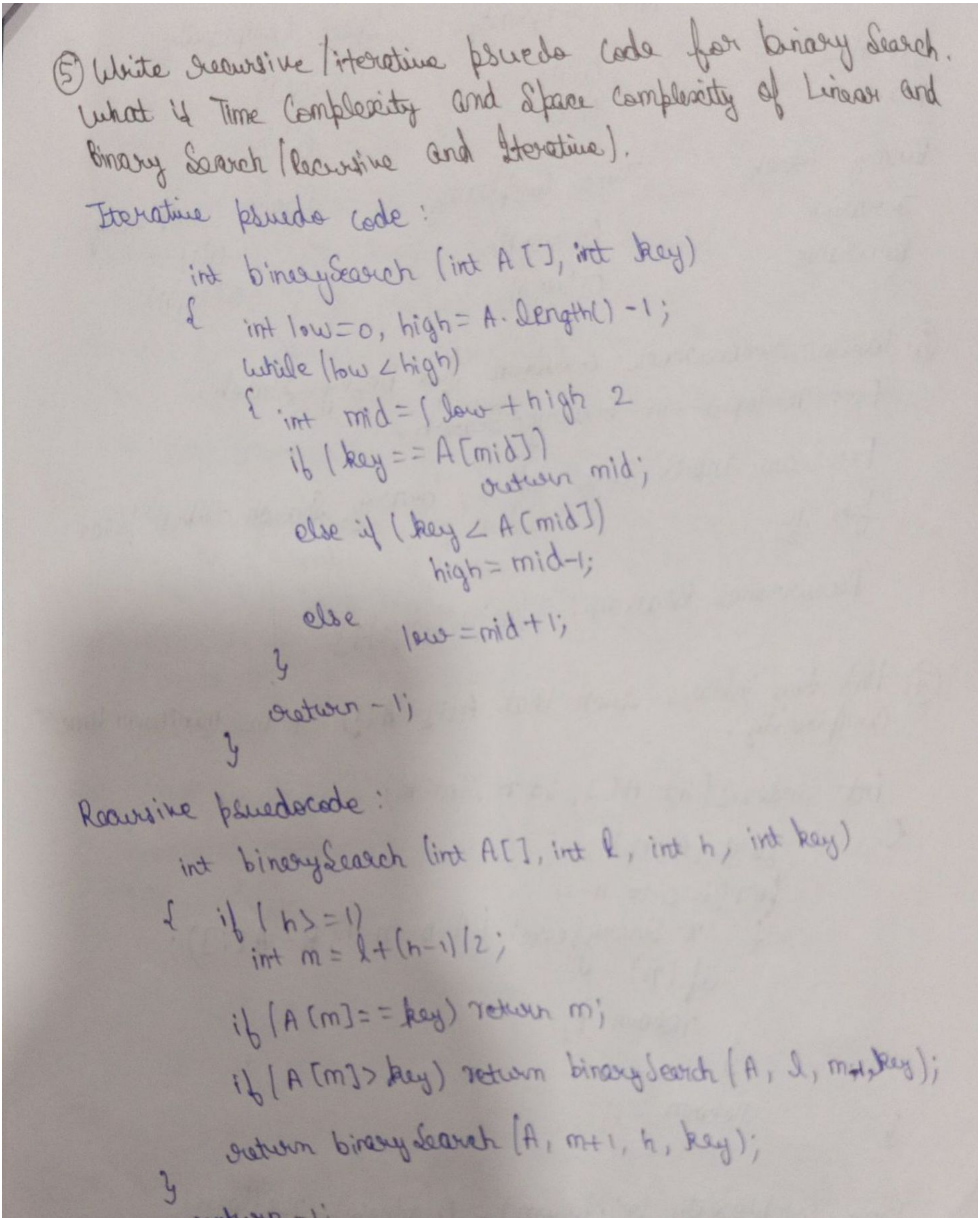
~~Selection~~

③ Complexity of all Sorting algorithms that have been discusse in lectures.

	Best	worst
Bubble Sort	$O(n)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$
Count Sort	$O(n+range)$	$O(n+range)$
Merge Sort	$O(n\log n)$	$O(n\log n)$
Quick Sort	$O(n\log n)$	$O(n^2)$
Randomized Quick Sort	$O(n\log n)$	$O(n^2)$
Heap Sort	$O(n\log n)$	$O(n\log n)$

④ Divide all the sorting algorithms ~~the~~ into inplace / Stable / online Sorting.

Sorting Algorithms	Inplace	Stable	Online
Bubble Sort	✓	✓	
Selection Sort	✓	X	X
Insertion Sort	✓	✓	✓
Count Sort	X	✓	
Quick Sort	✓	X	
Merge Sort	X	✓	X
Heap Sort	✓	X	



(5) Write recursive/iterative psuedo code for binary Search. What is Time Complexity and Space complexity of Linear and Binary Search (Recursive and Iterative).

Iterative psuedo code:

```
int binarySearch (int A[], int key)
{   int low=0, high= A.length() -1;
    while (low < high)
    {  int mid = (low + high 2
       if (key == A[mid])
                  return mid;
       else if (key < A[mid])
                  high = mid-1;
       else
                  low = mid + 1;
    }
    return -1;
}
```

Recursive psuedocode:

```
int binarySearch (int A[], int l, int h, int key)
{  if ( h>=1)
     int m = l+(h-1)/2;

     if (A[m]==key) return m;
     if (A[m]>key) return binarySearch (A, l, m-1, key);
     return binarySearch (A, m+1, h, key);
   }
   return -1;
}
```

Linear Search	Time Complexity	Space Complexity
Iterative	O(n)	O(1)
Recursive	O(n)	O(1)

Binary Search	Time Complexity	Space Complexity
Iterative	O(log n)	O(1)
Recursive	O(log n)	O(log n)

⑥ Write reccurrence relation for Binary Search.

~~For n input in Binary Search,~~

For an input size of n, binary Search takes place for $\frac{n}{2}$,

Reccurrence Relation: $T(n) = T(n/2)+1$

⑦ Find two indexes such that $A[i]+A[j] = k$ in minimum time complexity.

```
int indexes(int A[], int n, int K)
{       Sort (A, n);
        for(i=0 to n-1)
        {   x=binarySearch(A, 0, n-1, K-A[i]);
            if(n)
              return 1;
        }
        return -1;
}
```

Time complexity (minimum) $= O(n\log(n)) + nO(\log n)$

$= O(n\log(n))$.

(8) Which Sorting is best for practical uses. Explain ?

Each Sorting technique has its own perks ~~for~~ which makes them suitable for practical uses. But if the focus is on specific requirements, ~~like dealing with large~~ then accordingly a Sorting technique is of best use. Like, for dealing with large amount of data, Merge Sort is used. Also considering Nowadays data has only got bigger, Merge Sorting is the best to use ~~in~~ practically in current times.

(9) What do you mean by number of inversions in an array? Count the number of inversions in Array arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5} using merge Sort.

a pair is said to be inversion, let (a[i], a[j]) if a[i] > a[j]

~~let~~ given, arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5}

The total number of inversions are 31 using Merge Sort.

(10) In which cases, Quick Sort will give the best and worst case time complexity?

The best ~~ca~~ case time complexity Quick Sort gives is when pivot is selected as a mean element. Selecting mean element as pivot will divide the array in branches of equal size so that height of the ~~recursive~~ recursion tree will be minimum

The worst case time complexity i.e, $O(N^2)$, Quick sort gives is when our array is already sorted and the smallest element is selected as pivot or the largest element is. This will mean that the height of recursion tree will be n and we will be doing N number of operations.

(11) Write Reccurrence Relation of Merge and Quick Sort in best and worst case time Complexity?

Merge Sort	Best Case	Worst Case
Recurrenle Relation	$T(n) = 2T(n/2)+n$	
Quick Sort		
Reccurrence Relation	$T(n)= 2T(n/2)+n$	$T(n-1)+n$

(12) Selection Sort is not stable by default but ~~yout~~ you can write a version of Stable selection.

A version of Stable Selection can be written in which instead of swapping values, these values can be inserted.

(13) Bubble sort scans whole array even when array is sorted. Can you modify the bubble sort so that it doesn't scan the whole array once it is sorted.

```
void bubbleSort (int A[ ], int n)
{ for (int i=0 ; i<n; i++)
  {  int swap = 0;
   . for (int j=0 ; j<n-1-i ; j++)
```

7

~~(14) 4o~~

```
{ if (A[j] > A[j+1])
  { int t = A[i];
    A[j] = A[j+1];
    A[j+1]=t;
    swap++;
  }
}
if (swap==0)
      break;
}
}
```

(14) Your computer has a RAM of 2 GB and you are given an array of 4 GB for sorting. Which algorithm you are going to use for this purpose and why? Also explain the concept of External and Internal Sorting.

The algorithm that is going to be used ~~for this~~ to implement this task is external sorting. The 4GB array is read in main memory and sorted by Quick Sort. As quick sort is one of the external sorting methods which deals with data to be stored in RAM and is based on divide and conquer method.

External Sorting - It is a type of sorting algorithm that deals with large amount of data. It is used when the data to be stored does not fit in main memory and is stored in external memory. ex- Quick Sort, Merge Sort.

Internal Sorting - It is a type of sorting algorithm that deals with data which can be adjusted in the main memory. ex- Bubble Sort, Selection Sort.