15/12/24

Lecture-1

Introduction to Algorithms (Viva)
— Thomas H. Corman

Algorithm ⇒
A set of strictly defined instructions which is used to accomplish a specific task.

Precise → Unambiguous → finite time → No Repeat
Desired Result

Find the area of Rectangle →

General Purpose

Algorithm Strategy
- Approach to solving a problem
- May combine several approaches

Algorith structure

- Divide and conquer algorithm.
- Dynamic programming algorithm.
- Greedy algorithm.
- Backtracking algorithm.
- Branch Bound.
- Heuristic algorithms.

Greedy → i) Dijkstra's
ii) Minimum spaning
iii)
iv) Huffman

Deride and conquer → Examples

① Binary Search

② Quick sort

③ Merge sort

Dynamic Programming

- Based on remembering past results
- Approach
  1. Deride into sub problems
     - Must be of same type - Must overlap.

2. Recursive
3. Combine solutions.
4. optimize problem.

- Fibonacci Algorithm.

Example – i) 0-1 Knapsack
ii) Wanshall's All pairs shortest path
iii) Bellman Ford's Shortest path
iv) Matrix chain multiplication

Backtracking ⇒

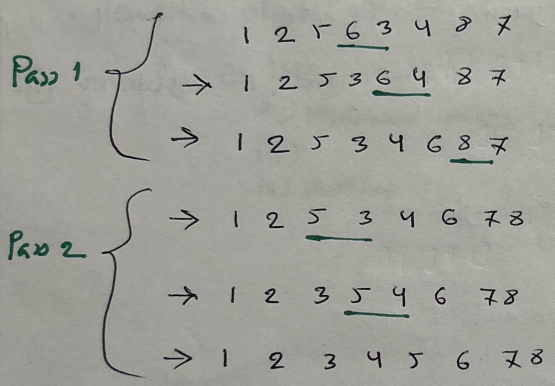
Lecture-2

18/12/24

Bubble Sort

- Compare each element
- Swap if out of order.
- Except the last one.

- Example




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

Algorithm

```
Bubblesort (A,n){
    for i = 1 to n {
        for j = i+1 to n {
            if A[j] < A[i] {
                swap A[i] and A[j]
            }
        }
    }
}
```

05/01/25

Insertion sort

Code:

```
void insertionSort(int arr[], int n){
    for(int i=1; i<n; ++i){
        int key = arr[i];
        int j = i-1;
        while(j>=0 && arr[j] > key){
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = key;
    }
}
```

=> 23, 1, 10, 5, 2

Pass 1: 1, 23, 10, 5, 2

Pass 2: 1, 10, 23, 5, 2

Pass 3: 1, 10, 5, 23, 2

1, 5, 10, 23, 2

Pass 4: 1, 5, 10, 2, 23

1, 5, 2, 10, 23

1, 2, 5, 10, 23

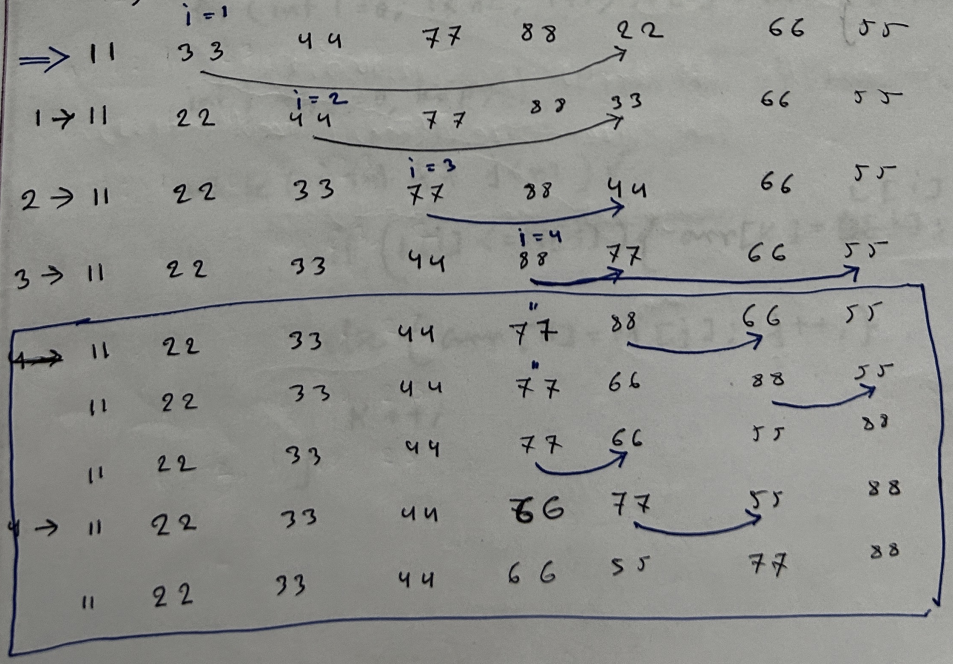
Algorithm:

```
Insertion Sort (A, n) {
    for i = 2 to n {
        Key = A[i];
        j = i - 1;
        while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
        }
        A[j+1] = key
    }
```

Selection Sort

Code:

```
void selectionSort(int arr[], int n){
    for(int i=0, i<n-1; i++){
        int min_idx = i;
        for(int j = i+1; j<n; j++){
            if(arr[j] < arr[min_idx])
                min_idx = j;
        }
        swap(&arr[min], &arr[i]);
    }
}
```

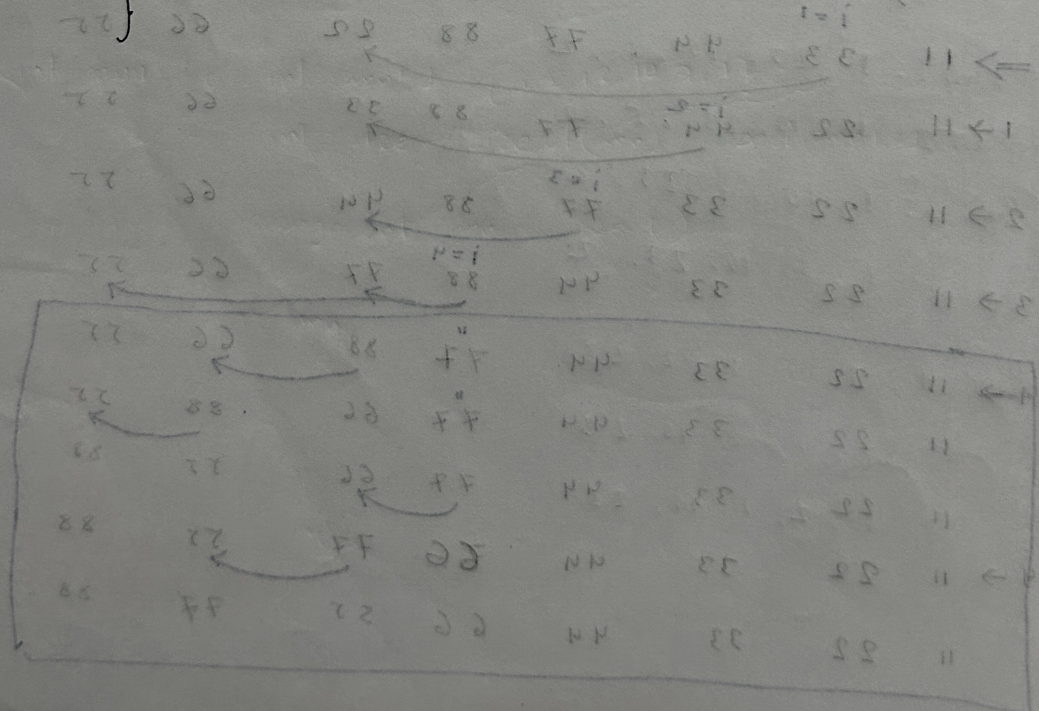


4→ 11 22 33 4 4 55 77 66 85

5→ 11 22 33 44 55 66 77 85

Algorithm :

```
SelectionSort (A, n) {
    for i = 1 to n {
        min = i
        for j = i+1 to n {
            if A[j] < A[min]
                min = j
        }
        exchange A[i] and A[min]
    }
}
```



12/01/25

Merge Sort

- Divide into sub-problems
- Conquer "
- Combine "

Code:

```
void merge (int arr[], int p, int q, int n){

    int n1 = q-p+1;
    int n2 = n-q;

    vector<int> L(n1);
    vector<int> M(n2);

    for (int i=0; i<n1; i++) L[i] = arr[p+i];

    for (int i=0; i<n2; i++) M[i] = arr[q+i+1];

    int i=0, j=0, k=p;
    while (i<n1 && j<n2){
        if (L[i] <= M[j]){ arr[k] = L[i]; i++; }
        else { arr[k] = M[j]; j++; }
        k++;
    }
```

```
    while (i< n1) { arr[k] = L[i]; i++; k++; }
    while (j <n2) {arr[k] = M[j]; j++; k++;}
}

void mergeSort (int arr[]; int l , int r) {
        if (l <r) {
            int m = l+(r-l)/2;
            mergeSort(arr, l,m);
            merge_sort (arr, m+1, r);
            merge (arr, l, m, r);
        }
}

int main() { int arr[] = {6,5,12,10,9,1};
             int size = sizeof(arr) / sizeof(arr[0]);
            mergeSort (arr ,0, size-1);
            return;
}
```

Algorithm:

```
void merge(int a[], int beg, int mid, int end){
    int i,j,k;
    int n1 = mid-big+1;
    int n2 = end-mid;
    int LeftArray[n1], RightArray[n2];
    for (int i=0; i<n1; i++) LeftArray[i] = a[beg+i];
    for (int i=0; i<n2; i++) RightArray[i] = a[mid+1+i];
    int i=0, j=0, K=beg;
    while (i<n1 && j<n2){
        if (LeftArray[i]<=RightArray[j]){
            a[k]=LeftArray[i];
            i++;
        }
        else {
            a[k] = RightArray[j];
            j++;
        }
        k++;
    }
```

```
while (i<n1){
        ~~Leta~~[k] = LeftArray[i];
        i++;
        k++;
    }
while (j<n2){
        a[k] = RightArray[j];
        j++;
        k++;
    }
}
```