

15/12/24

Lecture - 1

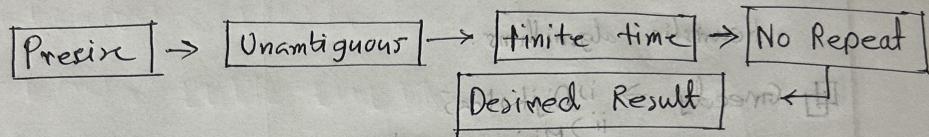
Introduction to Algorithms

- Thomas H. Cormen

(viva)

DEF Algorithm \Rightarrow

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



Ex Find the area of Rectangle \rightarrow

General Purpose

Algorithm Strategy

- Approach to solving a problem
- May combine several approaches

Algorithm structure

Top level ③

Tree ②

Bottom level ④

P21/21

I - Methods

. Divide and conquer algorithm.

• Dynamic programming algorithm.

• Greedy algorithm.

• Backtracking algorithm.

• Branch Bound.

• Heuristic algorithms.

II) Greedy \rightarrow

i) Dijkstras

ii) Minimum Spanning

iii)

iv) Huffman

Algorithm Design

III) Divide and conquer \rightarrow Examples

① Binary Search

② Quick sort

③ Merge sort

IV) Dynamic Programming

• Based on remembering past results

• Approaches

1. Divide into sub problems

• Must be of same type

• Must overlap.

2. Recomine

3. Combine solutions.

4. Optimize problem.

• Fibonacci Algorithm.

Example - 0-1 Knapsack

i) Warshall's All pairs shortest path

ii) Bellman Ford's Shortest path

iii) Matrix chain multiplication

Backtracking \Rightarrow

$0 \leq i, j < n = |A|$

$i + j > k, 0 = i + j$

$i + j \leq k < |A|$

$[i]_A = q_{i,j}$

$[i+k]_A = [i]_A$

$q_{i+k} = [i+k]_A$

X 8 4 2 7 5 1
X 8 2 2 7 5 1 ← } 1st
X 8 2 4 2 7 5 1 ← } 2nd

X 8 2 4 3 2 5 1 ← } 3rd
X 8 2 4 2 3 5 1 ← } 4th
X 8 2 4 2 3 5 1 ← } 5th

\Rightarrow 2 3 1 2 2 5 1

antidiagonal

(i, A) true/false

$i \neq i = i$ not

$i \neq i + i = i$ not

$[i]_A > [i]_A$ not

$[i]_A$ true $[i]_A$ false

Lecture - 2

18/12/24

Bubble Sort

Compare each element

- Swap if out of order.
- Except the last one.

Example

Pass 1 {
 1 2 5 6 3 4 8 7
 → 1 2 5 3 6 4 8 7
 → 1 2 5 3 4 6 8 7

Pass 2 {
 → 1 2 5 3 4 6 7 8
 → 1 2 3 5 4 6 7 8
 → 1 2 3 4 5 6 7 8

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

$\Rightarrow 23, 1, 10, 5, 2$

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:

Inserion sort (A, n) { Time: n^2 } Algorithm for insertion sort

for $i = 2$ to n { $i > i - 1 \neq i + 1$ } do

 key = $A[i]$; $i = 1$

$j = i - 1$; $i - i - 1 = j$

 while ($j > 0$) and ($A[j] > \text{key}$) { $j < i$

$A[j+1] = A[j]$

$j = j - 1$

}

$A[j+1] = \text{key}$

}

5, 7, 10, 1, 3, 2, 8

5, 7, 10, 1, 3, 2, 8, 6

5, 7, 10, 1, 3, 2, 8, 6, 4

5, 7, 10, 1, 3, 2, 8, 6, 4, 1

3, 5, 7, 10, 1, 3, 2, 8, 6, 4

3, 5, 7, 10, 1, 3, 2, 8, 6, 4

3, 5, 7, 10, 1, 3, 2, 8, 6, 4

Selection Sort

Code:

```
void selectionSort( int arr[ ], int n ) {
```

```
    for ( int i = 0, i < n - 1; i++ ) {
```

```
        int min_idx = i; (R.A) trial with sel
```

```
        for ( int j = i + 1; j < n; j++ ) {
```

```
            if ( arr[ j ] < arr[ min_idx ] )
```

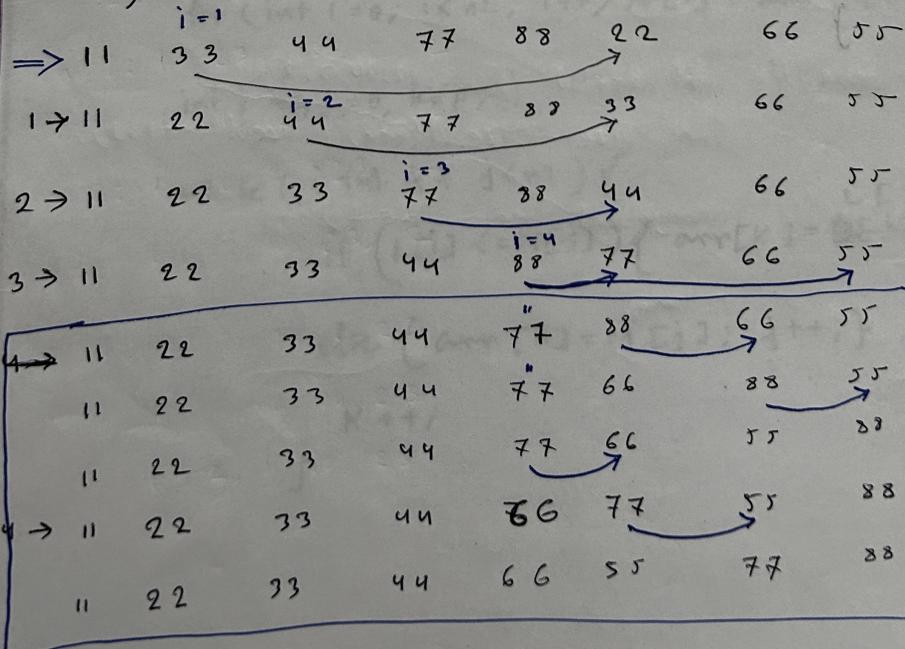
```
                min_idx = j;
```

```
    } } A > [ ] A 4i
```

```
    swap( &arr[ min_idx ], &arr[ i ] );
```

```
} for (A loop [ ] A scenarios
```

}



$4 \rightarrow 11 \quad 22 \quad 33 \quad 44 \quad 55 \quad 77 \quad 66 \quad 88$

$5 \rightarrow 11 \quad 22 \quad 33 \quad 44 \quad 55 \quad 66 \quad 77 \quad 88$

(After 1st Iteration) from middle b/w

Algorithm:

$\{ (++i : i < n & i < itni) \text{ not}$

SelectionSort (A, n) {

for ($i = 1$ to n) {

($min = i$ & $i > min$) $\forall i$

for ($j = i+1$ to n) {

if $A[j] < A[min]$

((i)_{ans} & $min = j$) gone

exchange $A[i]$ and $A[min]$

}

} do

$55 \quad 88 \quad FF \quad NP \quad EC \quad II \leftarrow i$

} do

$22 \quad 88 \quad FF \quad NP \quad EC \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

} do

$NP \quad 88 \quad EC \quad EC \quad SS \quad II \leftarrow i$

12/01/25

Merge Sort } (m > i) subproblems

- Divide into sub-problems

- Conquer

- Combine

Code:

```
void merge (int arr[], int p, int q, int r) {  
    int n1 = q - p + 1; n1 = m + n1;  
    int n2 = r - q; n2 = m + n2;  
  
    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];  
  
    int i = 0, j = 0, k = p; k = p, j = 0;  
    while (i < n1 && j < n2) {  
        if (L[i] <= M[j]) { arr[k] = L[i]; i++; }  
        else { arr[k] = M[j]; j++; }  
        k++;  
    }  
}
```

28/10/21

while ($i < n_1$) { $arr[K] = L[i]$; $i++$; $K++$; }

while ($j < n_2$) { $arr[K] = M[j]$; $j++$; $K++$; }

}

void mergesort (int arr[], int l, int r) {

} (n + m) $\approx \frac{n+m}{2}$ tri. $\approx \frac{1}{2} \times n \times m$ bior

if ($l < r$) {

int m = $l + (r-l)/2$; tri

mergesort (arr, l, m);

mergesort (arr, m+1, r);

merge (arr, l, m, r);

$([i+q] \text{ min} = [i]) \wedge (i > i \text{ min} \wedge o = i \text{ min})$ not

$([i+q] \text{ min} = [i] \text{ min} \wedge i > i \text{ min} \wedge o = i \text{ min})$ not

int main() { int arr[] = {6, 5, 12, 10, 9, 1};

int size = sizeof(arr) / sizeof(arr[0]);

mergesort (arr, 0, size-1);

return;

{ if ($i > m = \text{min}$) odo

}

Algorithm:

```
void merge(int a[], int beg, int mid, int end){  
    int i, j, k;  
    int n1 = mid - beg + 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 ($i < n_1$) {

} ($i = i + 1$, $k = k + 1$, $a[k] = LeftArray[i]$, $n_1 = n_1 - 1$)

$i++$;

$k++$;

}

while ($j < n_2$) {

$a[k] = RightArray[j]$;

$i = i + 1$, $j = j + 1$, $k = k + 1$, $n_2 = n_2 - 1$)

$j++$;

$k++$;

}

}

mergeBy($i > l \wedge r > i$)

} ($[l, r]_{\text{parent}} = [l, r]_{\text{parent}}$) if

$a[k] = LeftArray[i]$

int min() { int small = $i + 1$; }

int size = sizeOfTree(); }

else

$a[k] = RightArray[j]$

$i++$

}

{

$k++$

{