Design and Analysis of Algorithm (DAA)
Lab File

Name : Abir Banerjee

Registration No. : 209303126

Section : CCE D

1. BINARY SEARCH – ITERATIVE

CODE :

```
#include <iostream>
using namespace std;

int binarySearch(int array[], int x, int l, int h){
  while (l <= h){
    int m = l+(h-l)/2;
    if (array[m] == x)
      return m;
    if (array[m] < x)
      l = m+1;
    else
      h = m-1;
  }
  return -1;
}

int main(void){
  cout<<”Registration Number : 209303126”<<endl;
  int n,x;
  cout<<"Enter size of array"<<endl;
  cin>>n;
  int array[n];
  for(int i=0;i<n;i++){
    cout<<"Enter element "<<(i+1)<<endl;
    cin>>array[i];
  }
  cout<<"Enter the element to look for "<<endl;
  cin>>x;
  int result = binarySearch(array, x, 0, n - 1);
  if (result == -1)
    cout<<"Not found";
  else
    cout<<"Element is found at index "<< result<<" and position "<<result+1;
}
```

OUTPUT :

```
Reg Number : 209303126
Enter size of array
5
Enter element 1
10
Enter element 2
22
Enter element 3
34
Enter element 4
46
Enter element 5
58
Enter the element to look for
46
Element is found at index 3 and position 4
```

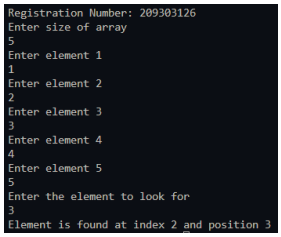
2.BINARY SEARCH – RECURSIVE

Code:

```
#include <iostream>
using namespace std;

int binarySearch(int array[], int x, int l, int h){
  if (l == h){
    if (array[l] == x)
      return array[l];
    else{
      return -1;
    }
  }
  else{
    int m = l+(h-l)/2;
    if(x==array[m]){
      return m;
    }
    else if(x<array[m]){
      return binarySearch(array,x,l,m-1);
    }
    else{
      return binarySearch(array,x,m+1,h);
    }
    return -1;
  }
}
int main(void){
  int n, x;
  cout << "Enter size of array" << endl;
  cin >> n;  int array[n];
  for (int i = 0; i < n; i++){
    cout << "Enter element " << (i + 1) << endl;
    cin >> array[i];
  }
  cout << "Enter the element to look for " << endl;
  cin >> x;
  int result = binarySearch(array, x, 0, n - 1);
  if (result == -1)    cout << "Not found";
  else    cout << "Element is found at index " << result << " and position " << result + 1;
}
```

OUTPUT:



```
Registration Number: 209303126
Enter size of array
5
Enter element 1
1
Enter element 2
2
Enter element 3
3
Enter element 4
4
Enter element 5
5
Enter the element to look for
3
Element is found at index 2 and position 3
```

3. SELECTION SORT

Code:

```
#include <iostream>
using namespace std;

void swap(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}
void printArray(int array[], int size) {
  for (int i = 0; i < size; i++) {
    cout << array[i] << " ";
  }
  cout << endl;
}
void selectionSort(int array[], int size) {
  for (int i = 0; i < size - 1; i++) {
    int x = i;
    for (int j = i+1; j<size; j++) {
      if (array[j] < array[x])
        x = j;
    }
    swap(&array[x], &array[i]);
  }
}
int main() {
  cout<<"Registration Number: 209303126"<<endl;
  int n,x;
  cout<<"Enter size of array"<<endl;
  cin>>n;
  int data[n];
  for(int i=0;i<n;i++){
    cout<<"Enter element "<<(i+1)<<endl;
    cin>>data[i];
  }
  selectionSort(data, n);
  cout << "Sorted array in Acsending Order:\n";
  printArray(data, n);
}
```

Output:

```
Registration Number: 209303126
Enter size of array
5
Enter element 1
20
Enter element 2
12
Enter element 3
56
Enter element 4
39
Enter element 5
100
Sorted array in Acsending Order:
12 20 39 56 100
```

4. BUBBLE SORT

Code:

```
#include <iostream>
using namespace std;

void bubbleSort(int array[], int size){
  for (int i = 0; i < size; ++i){
    for (int j = 0; j < size – i-1; ++j){
      if (array[j] > array[j + 1]){
        int temp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temp;
      }
    }
  }
}
void printArray(int array[], int size){
  for (int i = 0; i < size; ++i){
    cout << "  " << array[i];
  }
  cout << "\n";
}
int main(){
  cout<<"Registration Number: 209303126"<<endl;
  cout<<"Enter size of array"<<endl;
  int n; cin>>n;
  int data[n];
  for(int i=0;i<n;i++){
    cout<<"Enter element "<<(i+1)<<endl;
    cin>>data[i];
  }
  bubbleSort(data, n);
  cout << "Sorted Array in Ascending Order:\n";
  printArray(data, n);
}
```

Output:

```
Registration Number: 209303126
Enter size of array
5
Enter element 1
-1
Enter element 2
100
Enter element 3
-20
Enter element 4
200
Enter element 5
89
Swapping elements 100 and -20
Swapping elements 200 and 89
Swapping elements -1 and -20
Swapping elements 100 and 89
Sorted Array in Ascending Order:
  -20  -1  89  100  200
```

5. QUICK SORT

Code:

```
#include <iostream>
using namespace std;

void printArray(int array[], int size){
  int i;
  for (i = 0; i < size; i++)
    cout << array[i] << " ";
  cout << endl;
}
int partition(int array[], int low, int high){
  int pivot = array[low];
  int i = low+1;
  int j=high;
  do{
  while(array[i] < pivot){
    i++;
  }
  while(array[j] > pivot){
    j--;
  }
  if(i<j){
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
  }
  }while(i<j);

    int temp = array[j];
    array[j] = array[low];
    array[low] = temp;
    return j;
}
void quickSort(int array[], int low, int high){
  if (low < high){
    int pi = partition(array, low, high);
    quickSort(array, low, pi - 1);
    quickSort(array, pi + 1, high);
  }
}
int main(){
  cout<<"Registration Number: 209303126"<<endl;
  cout<<"Enter size of array"<<endl;
  int n; cin>>n;
  int data[n];
  for(int i=0;i<n;i++){
    cout<<"Enter element "<<(i+1)<<endl;
```

```
    cin>>data[i];
  }
  cout << "Unsorted Array: \n";
  printArray(data, n);
  quickSort(data, 0, n - 1);
  cout << "Sorted array in ascending order: \n";
  printArray(data, n);
}
```

Output:

```
Registration Number: 209303126
Enter size of array
6
Enter element 1
-10
Enter element 2
9
Enter element 3
-8
Enter element 4
7
Enter element 5
-6
Enter element 6
5
Unsorted Array:
-10 9 -8 7 -6 5
Sorted array in ascending order:
-10 -8 -6 5 7 9
```

6. Merge Sort

Code:

```
#include<iostream>
using namespace std;
void display(int *array, int size) {
   for(int i = 0; i<size; i++)
      cout << array[i] << " ";
   cout << endl;
}
void merge(int *array, int l, int m, int r) {
   int i, j, k, nl, nr;
   nl = m-l+1; nr = r-m;
   int larr[nl], rarr[nr];
   for(i = 0; i<nl; i++)
      larr[i] = array[l+i];
   for(j = 0; j<nr; j++)
      rarr[j] = array[m+1+j];
   i = 0; j = 0; k = l;
   while(i < nl && j<nr) {
      if(larr[i] <= rarr[j]) {
         array[k] = larr[i];
         i++;
      }else{
         array[k] = rarr[j];
         j++;
      }
      k++;
   }
   while(i<nl) {
      array[k] = larr[i];
      i++; k++;
   }
   while(j<nr) {
      array[k] = rarr[j];
      j++; k++;
   }
}
void mergeSort(int *array, int l, int r) {
   int m;
   if(l < r) {
      int m = l+(r-l)/2;
      mergeSort(array, l, m);
      mergeSort(array, m+1, r);
      merge(array, l, m, r);
   }
}
int main() {
   cout<<"Registration Number : 209303126"<<endl;
```

```
  int n;
  cout << "Enter the number of elements: "<<endl;
  cin >> n;
  int arr[n];
  for(int i = 0; i<n; i++) {
  cout << "Enter element "<<(i+1)<<endl;
    cin >> arr[i];
  }
  cout << "Array before Sorting: ";
  display(arr, n);
  mergeSort(arr, 0, n-1);
  cout << "Array after Sorting: ";
  display(arr, n);
}
```

Output:

```
Registration Number : 209303126
Enter the number of elements:
5
Enter element 1
-100
Enter element 2
10
Enter element 3
56
Enter element 4
62
Enter element 5
-87
Array before Sorting: -100 10 56 62 -87
Array after Sorting: -100 -87 10 56 62
```

7. KNAPSACK PROBLEM

Code:

```
#include<bits/stdc++.h>
using namespace std;
int max(int a, int b){
  if (a > b){ return a;}
  else{ return b; }
}z
int knapsack(int W, int wt[], int prof[], int n){
  int i, w;
  int knap[n + 1][W + 1];
  for (i = 0; i <= n; i++){
    for (w = 0; w <= W; w++){
      if (i == 0 || w == 0)
        knap[i][w] = 0;
      else if (wt[i - 1] <= w)
        knap[i][w] = max(prof[i - 1] + knap[i - 1][w - wt[i - 1]], knap[i - 1][w]);
      else
        knap[i][w] = knap[i - 1][w];
    }
  }
  return knap[n][W];
}
int main(){
  int n;
  cout<<"For registration number : 209303126 \n";
  cout<<"Enter number of values \n";
  cin>>n;
  int prof[n], wt[n];
  for (int i = 0; i < n; i++){
    cout<<"Enter the profit and weight of object"<<(i+1)<<endl;
    cin>>prof[i];
    cin>>wt[i];
  }
  cout<<"Enter the capacity of the knapsack \n";
  int weight;
  cin>>weight;
  cout<<"Maximum Profit is "<< knapsack(weight, wt, prof, n);
  return 0;
}
```

Output:

```
For registration number : 209303126
Enter number of values
5
Enter the profit and weight of object1
15 2
Enter the profit and weight of object2
4 15
Enter the profit and weight of object3
10 30
Enter the profit and weight of object4
20 80
Enter the profit and weight of object5
1 2
Enter the capacity of the knapsack
100
Maximum Profit is 40
```

8. INSERTION SORT

Code:

```
#include <iostream>
using namespace std;

void printArray(int array[], int size) {
  for (int i = 0; i < size; i++) {
    cout << array[i] << " ";
  }
  cout << endl;
}
void insertionSort(int array[], int size) {
  for (int step = 1; step < size; step++) {
    int key = array[step];
    int j = step - 1;
    while (key < array[j] && j >= 0) {
      array[j + 1] = array[j];
      --j;
    }
    array[j + 1] = key;
  }
}
int main() {
  cout<<"Enter number of elements"<<endl;
  int n;
  cin>>n;
  int a[n];
  for(int i=0;i<n;i++){
    cout<<"Enter element "<<(i+1)<<endl;
    cin>>a[i];
  }
  insertionSort(a,n);
  cout << "Sorted array in ascending order:\n";
  printArray(a,n);
}
```

Output:

```
For registration number : 209303126
Enter number of elements
10
Enter element 1
1
Enter element 2
9
Enter element 3
2
Enter element 4
8
Enter element 5
3
Enter element 6
7
Enter element 7
4
Enter element 8
8
Enter element 9
5
Enter element 10
10
Sorted array in ascending order:
1 2 3 4 5 7 8 8 9 10
```

9. BREADTH-FIRST SEARCH

Code:

```
#include <bits/stdc++.h>
using namespace std;
class Graph{
  int V;
  vector<list<int>> adj;
public:
  Graph(int V);
  void addEdge(int v, int w);
  void BFS(int s);
};
Graph::Graph(int V){
  this->V = V;
  adj.resize(V);
}
void Graph::addEdge(int v, int w){
  adj[v].push_back(w);
}
void Graph::BFS(int s){
  vector<bool> visited;
  visited.resize(V, false);
  list<int> queue;
  visited[s] = true;
  queue.push_back(s);
  while (!queue.empty()){
    s = queue.front();
    cout << s << " ";
    queue.pop_front();
    for (auto adjecent : adj[s]){
      if (!visited[adjecent]){
        visited[adjecent] = true;
        queue.push_back(adjecent);
      }}}}
int main(){
  cout<<"Registration Number : 209303126"<<endl;
  Graph g(4);
  g.addEdge(0, 1);  g.addEdge(0, 2);  g.addEdge(1, 2);  g.addEdge(2, 0);  g.addEdge(2, 3);
  g.addEdge(3, 3);
  cout << "Following is Breadth First Traversal "
       << "(starting from vertex 2) \n";
  g.BFS(2);
  return 0;
}
```

Output:

```
Registration Number : 209303126
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
```

10. DEPTH-FIRST SEARCH

Code:

```
#include <bits/stdc++.h>
using namespace std;
class Graph
{
public:
  map<int, bool> visited;
  map<int, list<int>> adj;
  void addEdge(int v, int w);
  void DFS(int v);
};

void Graph::addEdge(int v, int w){
  adj[v].push_back(w);
}

void Graph::DFS(int v){
  visited[v] = true;
  cout << v << " ";
  list<int>::iterator i;
  for (i = adj[v].begin(); i != adj[v].end(); ++i)
    if (!visited[*i])
      DFS(*i);
}

int main(){
  Graph g;
  cout<<"Registration Number : 209303126"<<endl;
  g.addEdge(0, 1);  g.addEdge(0, 2);
  g.addEdge(1, 2);  g.addEdge(2, 0);
  g.addEdge(2, 3);  g.addEdge(3, 3);
  cout << "Following is Depth First Traversal"
         " (starting from vertex 2) \n";
  g.DFS(2);
  return 0;
}
```

Output:

```
Registration Number : 209303126
Following is Depth First Traversal (starting from vertex 2)
2 0 1 3
```