21BDS0340

Abhinav Dinesh Srivatsa

Data Structures and Algorithms

Assignment – II

1. Sorting Algorithms

Aim

To implement linear and binary search.

a. Linear Search

Algorithm

```
search(array, size, n):
    Loop from 0 to size as x
        If array[x] = n
            Return x
    Return -1
```

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int search(int *arr, int size, int n)
{
    int pos = -1;
    for (int x = 0; x < size; x++)
        if (*(arr + x) == n)
            return x;
    return pos;
}

int main()
{
    bool flag = false;
    int *arr, size, n;
    printf("Enter number of elements: ");
```

```c
    scanf("%d", &size);
    arr = malloc(size * sizeof(int));
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    printf("Enter element to search for: ");
    scanf("%d", &n);
    int pos = search(arr, size, n);
    if (pos == -1)
        printf("Element not found.");
    else
        printf("Element found at position %d.", pos + 1);
    free(arr);
}
```

Output

```
Enter number of elements: 10
10
50
30
70
80
60
20
90
40
0
Enter element to search for: 20
Element found at position 7.
```

b. Binary Search

Algorithm

```
search(array, start, end, n):
    Declare mid as (start + end)/2

    If start = end
        return -1

    If array[mid] = n
        Return mid

    If array[mid] > n
        search(array, start, mid, n)
    Else search(array, mid, end, n)
```

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int search(int *arr, int start, int end, int n)
{
    int mid = (start + end) / 2;
    if (start == end)
        return -1;
    if (*(arr + mid) == n)
        return mid;
    else if (*(arr + mid) > n)
        search(arr, start, mid, n);
    else
        search(arr, mid, end, n);
}

int main()
{
    bool flag = false;
    int *arr, size, n;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(size * sizeof(int));
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
```

```c
    printf("Enter element to search for: ");
    scanf("%d", &n);
    int pos = search(arr, 0, size, n);
    if (pos == -1)
        printf("Element not found.");
    else
        printf("Element found at position %d.", pos + 1);
    free(arr);
}
```

Output

```
Enter number of elements: 10
20
30
50
60
70
80
110
130
140
170
Enter element to search for: 110
Element found at position 7.
```

2. Sorting Algorithms

Aim

To implement insertion, selection, bubble, quick, merge and counting sorts.

a. Selection Sort

Algorithm

```
array (array, size):
    Declare mindex

    Loop from 0 to size as x:
        Find min element and assign mindex
        as its index
        Swap array [x] and array [mindex]
```

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void sort(int *arr, int size)
{
    int mindex;
    for (int i = 0; i < size; i++)
    {
        mindex = i;
        for (int x = i; x < size; x++)
            if (*(arr + mindex) > *(arr + x))
                mindex = x;
        swap(arr + mindex, arr + i);
```

```
    }
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, size);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
    free(arr);
}
```

Output

```
Enter number of elements: 5
64
25
12
22
11
11 12 22 25 64
```

b. Insertion Sort

Algorithm

```
sort(array, size):
    loop from 1 to size as x:
        loop from 0 to x as y:
            if array[x] < array[y]:
                shift all elements to the right
                that are greater than array[x]
                insert arr[x] into the free space.
```

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

```c
void sort(int *arr, int size)
{
    int temp;
    for (int x = 1; x < size; x++)
        for (int y = 0; y < x; y++)
            if (*(arr + x) < *(arr + y))
            {
                temp = *(arr + x);
                for (int i = x; i >= y; i--)
                    *(arr + i) = *(arr + i - 1);
                *(arr + y) = temp;
            }
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, size);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
    free(arr);
A}
```

Output

```
Enter number of elements: 5
12
11
13
5
6
5 6 11 12 13
```

c. Bubble Sort

Algorithm

```
sort(array, size):
    Loop from 0 to size-1 as x
        Loop from 0 to size-x-1 as y
            If array[y] > array[x]:
                swap them.
```

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void sort(int *arr, int size)
{
    for (int x = 0; x < size - 1; x++)
        for (int y = 0; y < size - x - 1; y++)
            if (*(arr + y) > *(arr + y + 1))
                swap(arr + y, arr + y + 1);
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, size);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
```

```
    free(arr);
}
```

Output

```
Enter number of elements: 7
3
5
2
11
4
7
13
2 3 4 5 7 11 13
```

d. Quick Sort

Algorithm

```
sort(array, left, right):
    Declare start = left, end = right
    Declare pivot = left
    If left >, right, return
    while left < right:
        If array[left] > array[right]:
            swap them
            If pivot = left, then pivot = right
            Else pivot = left
        If pivot = left, right = right - 1
        Else left = left + 1
    call sort(arr, start, ~~pivot~~, pivot - 1)
    call sort(arr, pivot + 1, end)
```

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void sort(int *arr, int left, int right)
{
    int start = left, end = right;
    int pivot = left;
    if (left >= right)
        return;
    while (left < right)
    {
        if (*(arr + left) > *(arr + right))
        {
            swap(arr + left, arr + right);
            if (pivot == left)
                pivot = right;
            else
                pivot = left;
        }
        if (pivot == left)
            right--;
        else
            left++;
    }
    sort(arr, start, pivot - 1);
    sort(arr, pivot + 1, end);
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, 0, size - 1);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
```

```
    free(arr);
}
```

Output

```
Enter number of elements: 7
10
80
30
90
40
50
70
10 30 40 50 70 80 90
```

e. Merge Sort

Algorithm

```
merge(array, s1, e1, s2, e2):
    Declare temp
    Declare i, j = s1, s2
    while i ≤ e1 and j ≤ e2
        If array[i] < array[j]
            Append array[i] to temp
        Else append array[j] to temp
    Put remaining indices of i's elements in temp
    Do the same for j's elements

sort(array, start, end):
    If start = end, return
    Declare mid = (start + end)/2
    Call sort(array, start, mid)
    Call sort(array, mid+1, end)
    Call merge(array, start, mid, mid+1, end)
```

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void merge(int *arr, int s1, int e1, int s2, int e2)
{
    int *temp = malloc((e2 - s1) * sizeof(int));
    int tempdex = 0;
    int i = s1, j = s2;
    while (i <= e1 && j <= e2)
        if (*(arr + i) < *(arr + j))
            *(temp + tempdex++) = *(arr + i++);
        else
            *(temp + tempdex++) = *(arr + j++);
    while (i <= e1)
        *(temp + tempdex++) = *(arr + i++);
    while (j <= e2)
        *(temp + tempdex++) = *(arr + j++);
    tempdex = 0;
    for (int x = s1; x <= e2; x++)
        *(arr + x) = *(temp + tempdex++);
}

void sort(int *arr, int start, int end)
{
    if (start == end)
        return;
    int mid = (start + end) / 2;
    sort(arr, start, mid);
    sort(arr, mid + 1, end);
    merge(arr, start, mid, mid + 1, end);
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, 0, size - 1);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
    free(arr);
}
```

Output

```
Enter number of elements: 7
38
27
43
3
9
82
10
3 9 10 27 38 43 82
```

f. Counting Sort

Algorithm

sort(array, size):

Declare max as max value in array

Declare array count with max+1 spaces

Get no. of elements in array and put the corresponding values in count

Loop from left to right in count:

count = count + prev count index

If count - count's next index > 1,

then add the index to array

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void sort(int *array, int size)
{
    int max = *array;
    for (int x = 0; x < size; x++)
        if (max < *(array + x))
            max = *(array + x);
    int *count = calloc(max + 1, sizeof(int));
    for (int x = 0; x < size; x++)
        (*(count + *(array + x)))++;
    for (int x = 1; x < max + 1; x++)
```

```
        *(count + x) += *(count + x - 1);
    int arrindex = 0;
    if (*count == 1)
        *(array + arrindex++) = 0;
    for (int x = 1; x < max + 1; x++)
        for (int y = 0; y < *(count + x) - *(count + x - 1); y++)
            *(array + arrindex++) = x;
    free(count);
}

int main()
{
    int *arr, size;
    printf("Enter number of elements: ");
    scanf("%d", &size);
    arr = malloc(sizeof(int) * size);
    for (int x = 0; x < size; x++)
        scanf("%d", arr + x);
    sort(arr, size);
    for (int x = 0; x < size; x++)
        printf("%d ", *(arr + x));
    free(arr);
}
```

Output

```
Enter number of elements: 7
1
4
1
2
7
5
2
1 1 2 2 4 5 7
```