

Data Structures Day 3

T. Sai krishna (192211870)

31) Write C programme to implement Bubble sort.

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C program named 'bubble sort.cpp' with the following content:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int array[100], n, c, d, swap;
6     printf("Enter number of elements\n");
7     scanf("%d", &n);
8     printf("Enter %d integers\n", n);
9     for (c = 0; c < n; c++)
10    {
11        scanf("%d", &array[c]);
12    }
13    for (c = 0; c < n - 1; c++)
14    {
15        for (d = 0; d < n - c - 1; d++)
16        {
17            if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */
18            {
19                swap = array[d];
20                array[d] = array[d+1];
21                array[d+1] = swap;
22            }
23        }
24        printf("Sorted list in ascending order:\n");
25        for (c = 0; c < n; c++)
26        {
27            printf("%d\n", array[c]);
28        }
29    }
}
```

The terminal window shows the output of the program. It prompts for the number of elements (6), then six integers (12, 19, 5, 7, 15, 21). The sorted list is then printed as 5, 7, 12, 15, 19, 21. The status bar at the bottom indicates the process exited after 39.44 seconds with return value 0.

32) Write C programme to implement Insertion sort.

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C program named 'INSEETION SORT.cpp' with the following content:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int arr[100], n, i, j, key;
6     printf("Enter the number of elements: ");
7     scanf("%d", &n);
8     printf("Enter the elements: ");
9     for (i = 0; i < n; i++)
10    {
11        scanf("%d", &arr[i]);
12    }
13    printf("Original array: ");
14    for (i = 0; i < n; i++)
15    {
16        printf("%d ", arr[i]);
17    }
18    for (i = 1; i < n; i++)
19    {
20        key = arr[i];
21        j = i - 1;
22        while (j >= 0 && arr[j] > key)
23        {
24            arr[j + 1] = arr[j];
25            j = j - 1;
26        }
27        arr[j + 1] = key;
28    }
29    printf("\nSorted array: ");
30    for (i = 0; i < n; i++)
31    {
32        printf("%d ", arr[i]);
33    }
34    return 0;
35 }
```

The terminal window shows the output of the program. It prompts for the number of elements (6), then six integers (12, 19, 5, 7, 15, 21). The original array is printed as 12, 19, 5, 7, 15, 21. The sorted array is then printed as 5, 7, 12, 15, 19, 21. The status bar at the bottom indicates the process exited after 46.94 seconds with return value 0.

33) Write C programme to implement Merge sort.

The screenshot shows the Dev-C++ IDE interface. The code editor displays the following C program:

```
#include <stdio.h>
#define max 10
int a[10] = { 2, 6, 9, 7, 10, 4, 3, 1, 6, 8, 7 };
int b[10];
void merging(int low, int mid, int high) {
    int i1, i2, i;
    for(i1 = low, i2 = mid + 1, i = low; i1 <= mid && i2 <= high; i++) {
        if(a[i1] <= a[i2])
            b[i] = a[i1++];
        else
            b[i] = a[i2++];
    }
    while(i1 <= mid)
        b[i++] = a[i1++];
    while(i2 <= high)
        b[i++] = a[i2++];
    for(i = low; i <= high; i++)
        a[i] = b[i];
}
void sort(int low, int high) {
    int mid;
    if(low < high) {
        mid = (low + high) / 2;
        sort(low, mid);
        sort(mid+1, high);
    }
}
int main() {
    int i;
    printf("List before sorting\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
    sort(0, max);
    printf("List after sorting\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
}
```

The output window shows the list before sorting and the list after sorting. The list before sorting is: 2 6 9 7 10 4 3 1 6 8 7. The list after sorting is: 1 2 3 4 6 6 7 7 8 9 10.

The status bar at the bottom indicates the date and time as 15-05-2023 12:09.

The screenshot shows the Dev-C++ IDE interface. The code editor displays the following C program:

```
b[10] = a[12++];
for(i = low; i <= high; i++)
    a[i] = b[i];
}
void sort(int low, int high) {
    int mid;
    if(low < high) {
        mid = (low + high) / 2;
        sort(low, mid);
        sort(mid+1, high);
        merging(low, mid, high);
    }
    else
        return;
}
int main() {
    int i;
    printf("List before sorting\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
    sort(0, max);
    printf("List after sorting\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
}
```

The output window shows the list before sorting and the list after sorting. The list before sorting is: 2 6 9 7 10 4 3 1 6 8 7. The list after sorting is: 1 2 3 4 6 6 7 7 8 9 10.

The status bar at the bottom indicates the date and time as 15-05-2023 12:09.

34) Write C programme to implement Quick sort.

The screenshot shows the Dev-C++ IDE interface. The code editor window displays the following C program:

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

int quickSort(int *arr, int low, int high)
{
    int i = low, j = high;
    int pivot = arr[(low + high) / 2];
    while (i <= j)
    {
        while (arr[i] < pivot)
            i++;
        while (arr[j] > pivot)
            j--;
        if (i <= j)
        {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
    }
    if (low < j)
        quickSort(arr, low, j);
    if (j < high)
        quickSort(arr, i, high);
    return 0;
}

int main(void)
{
    puts("Enter the number of elements in the array: ");
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        arr[i] = rand() % 21;
    quickSort(arr, 0, n - 1);
    puts("The sorted array is:");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

To the right of the code editor is a terminal window showing the output of the program. It prompts for the number of elements (6), then lists the array elements (10, 12, 15, 5, 7, 21), sorts them, and prints the result (5, 7, 12, 15, 19, 21). Below the terminal window is the compiler log, which shows a permission denied error for the output file.

At the bottom of the screen is the Windows taskbar with various pinned icons and system status indicators.

The screenshot shows the Dev-C++ IDE interface. The code editor window displays the following C program:

```
int arr[n];
for (int i = 0; i < n; i++)
{
    printf("arr[%d]: ", i);
    scanf("%d", &arr[i]);
}
int low = 0;
int high = n - 1;
int pivot = arr[high];
int k = low - 1;
for (int j = low; j < high; j++)
{
    if (arr[j] <= pivot)
    {
        k++;
        int temp = arr[k];
        arr[k] = arr[j];
        arr[j] = temp;
    }
}
int temp = arr[k + 1];
arr[k + 1] = arr[high];
arr[high] = temp;
int pi = k + 1;
quickSort(arr, low, pi - 1);
quickSort(arr, pi + 1, high);
puts("The sorted array is:");
for (int i = 0; i < n; i++)
{
    printf("%d ", arr[i]);
}
return 0;
```

To the right of the code editor is a terminal window showing the output of the program. It prompts for the elements of the array, sorts them, and prints the result. Below the terminal window is the compiler log, which shows a permission denied error for the output file.

At the bottom of the screen is the Windows taskbar with various pinned icons and system status indicators.

35) Write C programme to implement DIJKSTRA.

The screenshot shows the Dev-C++ IDE interface with the code editor containing the Dijkstra's algorithm implementation. The code uses structures for list nodes and table entries, and includes functions for building the graph, calculating shortest paths, and printing results. The compiler log shows no errors or warnings, and the output window displays the execution of the program, which asks for vertices and their connections, and then prints the shortest path from vertex 1 to vertex 0.

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<malloc.h>
4 #include<stdlib.h>
5 #define infinity 999
6
7 int j=0;
8 int adjacent[20];
9 int cost=0;
10
11 struct listnode
12 {
13     int v;
14     struct listnode *next;
15 };
16
17 struct listnode *list;
18
19 struct tableEntry
20 {
21     int known;
22     int dist;
23     int path;
24 };
25
26 typedef struct tableEntry Table[10];
27
28 Table T;
29
30 void buildadj(int v1,int v2,int c1)
31 {
32     list temp=(list)malloc(sizeof(struct listnode));
33     temp->v=v2;
34     temp->cost=c1;
35     temp->next=NULL;
36 }
37
38 void caladj(int s)
39 {
40     list temp;
41     temp->s=s;
42     j=0;
43     for(temp=T[s].head->next,temp!=NULL,temp->next)
44     {
45         adjacent[j]=temp->v;
46         j++;
47     }
48 }
49
50 void findcost(int s,int d)
51 {
52     list temp;
53     for(temp=T[s].head->next,temp->v!=d,temp->next)
54     {
55         cost=temp->cost;
56     }
57 }
58
59 void ReadGraph()
60 {
61     int Ed,v1,v2,c1,i;
62     printf("Enter the no of edges:\n");
63     scanf("%d",&Ed);
64     for(i=1;i<=Ed;i++)
65     {
66         printf("Enter the Vertices pair with cost:\n");
67         scanf("%d%d%d",&v1,&v2,&c1);
68         buildadj(v1,v2,c1);
69     }
70 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 132.2900390625 KiB
- Compilation Time: 0.20s

Enter the source vertex:1
Enter the destination to which u want to findout shortpath:6
Enter the no of vertices:7
Enter the no of edges:12
Enter the Vertices pair with cost:
1 4 5
Enter the Vertices pair with cost:
1 2 3
Enter the Vertices pair with cost:
2 5 10
Enter the Vertices pair with cost:
2 4 3
Enter the Vertices pair with cost:
3 6 5
Enter the Vertices pair with cost:
4 5 2
Enter the Vertices pair with cost:
4 3 1
Enter the Vertices pair with cost:
1 1 4
Enter the Vertices pair with cost:
3 6 9
Enter the Vertices pair with cost:
4 3 5
Enter the Vertices pair with cost:
7 6 1
Enter the Vertices pair with cost:
5 7 8
Known dv Pv
1 0 0
1 2 1
1 3 4
1 1 1
1 5 6
1 8 3
1 9 5
SHORTEST PATH FROM 1 To 0
1->4->3->6->5->2->1

The screenshot shows the Dev-C++ IDE interface with the code editor containing the Dijkstra's algorithm implementation. The code is identical to the one in the first screenshot. The compiler log shows no errors or warnings, and the output window displays the execution of the program, which asks for vertices and their connections, and then prints the shortest path from vertex 1 to vertex 0.

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<malloc.h>
4 #include<stdlib.h>
5 #define infinity 999
6
7 int j=0;
8 int adjacent[20];
9 int cost=0;
10
11 struct listnode
12 {
13     int v;
14     struct listnode *next;
15 };
16
17 struct listnode *list;
18
19 struct tableEntry
20 {
21     int known;
22     int dist;
23     int path;
24 };
25
26 Table T;
27
28 void buildadj(int v1,int v2,int c1)
29 {
30     list temp=(list)malloc(sizeof(struct listnode));
31     temp->v=v2;
32     temp->cost=c1;
33     temp->next=NULL;
34 }
35
36 void caladj(int s)
37 {
38     list temp;
39     temp->s=s;
40     j=0;
41     for(temp=T[s].head->next,temp!=NULL,temp->next)
42     {
43         adjacent[j]=temp->v;
44         j++;
45     }
46 }
47
48 void findcost(int s,int d)
49 {
50     list temp;
51     for(temp=T[s].head->next,temp->v!=d,temp->next)
52     {
53         cost=temp->cost;
54     }
55 }
56
57 void ReadGraph()
58 {
59     int Ed,v1,v2,c1,i;
60     printf("Enter the no of edges:\n");
61     scanf("%d",&Ed);
62     for(i=1;i<=Ed;i++)
63     {
64         printf("Enter the Vertices pair with cost:\n");
65         scanf("%d%d%d",&v1,&v2,&c1);
66         buildadj(v1,v2,c1);
67     }
68 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 132.2900390625 KiB
- Compilation Time: 0.20s

Enter the source vertex:1
Enter the destination to which u want to findout shortpath:6
Enter the no of vertices:7
Enter the no of edges:12
Enter the Vertices pair with cost:
1 4 5
Enter the Vertices pair with cost:
1 2 3
Enter the Vertices pair with cost:
2 5 10
Enter the Vertices pair with cost:
2 4 3
Enter the Vertices pair with cost:
3 6 5
Enter the Vertices pair with cost:
4 5 2
Enter the Vertices pair with cost:
4 3 1
Enter the Vertices pair with cost:
1 1 4
Enter the Vertices pair with cost:
3 6 9
Enter the Vertices pair with cost:
4 3 5
Enter the Vertices pair with cost:
7 6 1
Enter the Vertices pair with cost:
5 7 8
Known dv Pv
1 0 0
1 2 1
1 3 4
1 1 1
1 5 6
1 8 3
1 9 5
SHORTEST PATH FROM 1 To 0
1->4->3->6->5->2->1

C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11

```
File Edit Search View Project Execute Tools ASyntax Window Help
TDM-GCC 4.9.2 64-bit Release
(globals) ▾
Project Classes Debug Untitled1.cpp
listnode: struct
tableEntry: struct
adjacency (int N) : void
build (int v1, int v2, int d)
calcd (int s) : void
Dijkstra (int N) : void
findcost (int s, int d) :
initTable (int s, int N):
main () :
printpath (int v) :
printTable (int N) : void
ReadGraph () : void
adjacent [20] : int
cost : int
j: int
T: Table
void printpath(int v)
{
    if(T[v].path!=0)
    {
        printpath(T[v].path);
        printf("%d->",v);
    }
}
int main()
{
    int s,N,d;
    printf("Enter the source vertex:");
    scanf("%d",&s);
    printf("Enter the destination to which u want to findout shortestpath:");
    scanf("%d",&d);
}

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 132.2900390625 KiB
- Compilation Time: 0.20s
35°C Haze
ENG IN 18:26 15-05-2023
```

C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11

```
File Edit Search View Project Execute Tools ASyntax Window Help
TDM-GCC 4.9.2 64-bit Release
(globals) ▾
Project Classes Debug Untitled1.cpp
listnode: struct
tableEntry: struct
adjacency (int N) : void
build (int v1, int v2, int d)
calcd (int s) : void
Dijkstra (int N) : void
findcost (int s, int d) :
initTable (int s, int N):
main () :
printpath (int v) :
printTable (int N) : void
ReadGraph () : void
adjacent [20] : int
cost : int
j: int
T: Table
void printpath(int v)
{
    if(T[v].path!=0)
    {
        printpath(T[v].path);
        printf("%d->",v);
    }
}
int main()
{
    int s,N,d;
    printf("Enter the source vertex:");
    scanf("%d",&s);
    printf("Enter the destination to which u want to findout shortestpath:");
    scanf("%d",&d);
    printf("Enter the No of vertices:");
    scanf("%d",&N);
    initTable(s,N);
    Dijkstra(N);
    printTable(N);
    printf("\nSHORTEST PATH FROM %d To %d",s,d);
    printf("\n%d->%d",s,d);
    printpath(d);
    getch();
}

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 132.2900390625 KiB
- Compilation Time: 0.20s
35°C Haze
ENG IN 18:26 15-05-2023
```

36) Write C programme to implement PRIMS

The screenshot shows the Dev-C++ IDE interface with two windows open. The left window displays the code for a Dijkstra's algorithm implementation in C. The right window shows the terminal output of the program being run.

Code (Untitled1.cpp):

```
#include<stdio.h>
#include<malloc.h>
#include<conio.h>
#define INFINITY 999
int j=0;
int adj[20][20];
int cost[20];
int Totalcost=0;
main()
{
    struct listnode
    {
        int v;
        int cost;
        struct listnode *next;
    };
    typedef struct listnode *list;
    struct tableEntry
    {
        list head;
        int known;
        int dist;
        int path;
    };
    typedef struct tableEntry Table[10];
    Table T;
    void buildd(int v1,int v2,int c1)
    {
        list temp=(list)malloc(sizeof(struct listnode));
        list temp1=(list)malloc(sizeof(struct listnode));
        temp->v=v2;
        temp1->v=v1;
        temp1->cost=c1;
        temp->cost=c1;
    }
}
```

Terminal Output:

```
Enter the source vertex:1
Enter the No of vertices:8
enter the no of edges:6
Enter the Vertices pair along with cost 1 4
2
Enter the Vertices pair along with cost 2 3
7
Enter the Vertices pair along with cost 5 6
5
Enter the Vertices pair along with cost 7 8
10
Enter the Vertices pair along with cost 1 6
6
Enter the Vertices pair along with cost 4 5
8
**FINAL TABLE**
-----
```

vertex	Known	dv	Pv
1	1	0	0
2	0	999	0
3	0	999	0

Compiler Results:

```
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 131.5830078125 Kib
- Compilation Time: 0.24s
```

System tray icons include: 35°C Haze, ENG IN, 15-05-2023, 12:58.

The screenshot shows the Dev-C++ IDE interface with two main windows. The left window displays the source code for `Untitled1.cpp`, which contains a C program implementing Dijkstra's algorithm for finding the shortest path between vertices in a weighted graph. The right window shows the terminal output of the compiled executable `Untitled1.exe`, demonstrating the algorithm's execution and output.

Untitled1.cpp Content:

```
listnode struct
tableentry struct
node
builded(int v1, int v2)
calcd(int v)
findcost(node, int d)
initTable(int n, int NV)
main()
prims(int N)
printTable(int N)
ReadGraph()
adjacent[20]
cost: int
j: int
T: Table
Totalcost: int

listnode struct
{
    temp->cost=c1;
    temp->next=NULL;
    temp1->next=NULL;
}
calcd(int v)
{
    temp->next=T[v].head->next;
    T[v].head=>temp;
}
findcost(node, int d)
{
    temp1->next=T[v2].head->next;
    T[v2].head=>temp1;
}
initTable(int n, int NV)
{
    main();
}
prims(int N)
{
    void calcd(int s)
    {
        list temp;
        j=0;
        for(temp=T[s].head->next;temp!=NULL;temp=>next)
        {
            adjacent[j+=1]=temp->v;
        }
    }
    void findcost(int s, int d)
    {
        list temp;
        for(temp=T[s].head->next;temp->v!=d;temp=>next)
        {
            cost=temp->cost;
        }
    }
}
ReadGraph()
{
    int end,v1,v2,c1,i;
    printf("Enter the no of edges:");
    scanf("%d",&end);
    for(i=1;i<end;i++)
}

Compiler Results:
Compilation results...
Errors: 0
Warnings: 0
Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
Output Size: 131.5830078125 Kib
Compilation Time: 0.24s
```

Terminal Output (Untitled1.exe):

```
Enter the Vertices pair along with cost 5
5
Enter the Vertices pair along with cost 7
8
Enter the Vertices pair along with cost 1
6
Enter the Vertices pair along with cost 4
5
**FINAL TABLE**
vertex Known dv PV
-----
1 0 0 0
2 0 999 0
3 0 999 0
4 1 2 1
5 1 5 6
6 0 1 6 1
7 0 999 0
8 0 999 0
-----
Minimum cost of the tree is 33:
-----
```

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11
- Menu Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbar:** Standard Dev-C++ toolbar with icons for file operations, compilation, and debugging.
- Project Explorer:** Shows the project structure with files Untitled1.cpp and Untitled1.h.
- Code Editor:** The code for Dijkstra's algorithm is displayed. It includes functions for building a list node, finding the minimum cost vertex, and printing the final table. The code uses a priority queue represented by a linked list of nodes.
- Output Window:** Displays the execution of the program. It asks for vertex pairs and their costs, builds the graph, and then prints the final table and the minimum cost of the tree.
- Compiler Tab:** Shows compilation results with 0 errors and 0 warnings.
- Resources Tab:** Shows output file information: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe, size 131.5830078125 Kib, and compilation time 0.24s.

```
7
Enter the Vertices pair along with cost 5
5

Enter the Vertices pair along with cost 7
10

Enter the Vertices pair along with cost 1
6

Enter the Vertices pair along with cost 4
5

**FINAL TABLE**
vertex Known dv Pv
-----
1 1 0 0
2 0 999 0
3 0 999 0
4 1 2 1
5 1 5 6
6 1 6 1
7 0 999 0
8 0 999 0
-----
Minimum cost of the tree is 33;
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 131.5830078125 Kib
- Compilation Time: 0.24s

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - (Executing) - Dev-C++ 5.11
- Menu Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbars:** Standard, Advanced, Compiler, Resources, Compile Log, Find Results.
- Project Explorer:** Shows the file Untitled1.cpp with various functions and variables defined.
- Code Editor:** The code implements Dijkstra's algorithm. It includes structures for listnodes and tableentry, and functions for building the table, calculating distances, finding shortest paths, and printing results. It uses a priority queue represented by a linked list of nodes.
- Compiler Output:** Shows compilation results with 0 errors and 0 warnings. The output file is C:\Users\saikr\OneDrive\Desktop\Untitled1.exe, size 131.5830078125 KIB, and compilation time 0.24s.

C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11

```

Project Classes Debug Untitled1.cpp
File Edit Search View Project Execute Tools ASyle Window Help
File Project Properties Build Run View Tools Options Help
(globals) (Untitled1)
Project Classes Debug Untitled1.cpp
File Edit Search View Project Execute Tools ASyle Window Help
File Project Properties Build Run View Tools Options Help
(globals) (Untitled1)


```

listnode: struct
tableEntry: struct
build(int v1, int v2,
calcd(int s, void
findcost(int s, int d):
initTable(int s, int N,
main():
prims(int N) : void
printTable(int N) : void
ReadGraph(): void
adjacent[20]: int
cost: int
j: int
T: Table
Totalcost: int
126 +> c;
127 calcd(v);
128 for(i=1;i<=j;i++)
129 {
130 if(v==adjacent[i])
131 if(T[i].known==0)
132 {
133 findcost(v,w);
134 if(cost[T[i].dist])
135 {
136 T[w].dist=cost;
137 T[w].path=v;
138 }
139 }
140 min=99;
141 }
142 }
143 }
144 }
145 int main()
146 {
147 int s,N,d;
148 printf("Enter the source vertex:");
149 scanf("%d",&s);
150 printf("Enter the No of vertices:");
151 scanf("%d",&N);
152 initTable(s,N);
153 prims(N);
154 printTable(N);
155 printf("\n*****");
156 printf("\nMinimum cost of the tree is %d:",Totalcost);
157 printf("\n*****");
158 getch();
159 }

```


Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 131.5830078125 Kib
- Compilation Time: 0.24s

```

35°C Haze 13:01 15-05-2023 ENG IN

37) Write C programme to implement DFS.

C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11

```

Project Classes Debug Untitled1.cpp
File Edit Search View Project Execute Tools ASyle Window Help
File Project Properties Build Run View Tools Options Help
File Project Properties Build Run View Tools Options Help
(globals) (Untitled1)
Project Classes Debug Untitled1.cpp
File Edit Search View Project Execute Tools ASyle Window Help
File Project Properties Build Run View Tools Options Help
File Project Properties Build Run View Tools Options Help
(globals) (Untitled1)


```

dfs(int graph[MAX_NODES]
main(): int
1 #include <stdio.h>
2 define MAX_NODES 100
3 void dfs(int graph[MAX_NODES][MAX_NODES], int visited[MAX_NODES], int currentNode, int numnodes)
4 {
5 printf("Id ", currentNode);
6 visited[currentNode] = 1;
7
8 for (i = 0; i < numnodes; i++) {
9 if (graph[currentNode][i] && !visited[i]) {
10 dfs(graph, visited, i, numnodes);
11 }
12 }
13 }
14 int main()
15 {
16 int numnodes, i, j;
17 int graph[MAX_NODES][MAX_NODES];
18 int visited[MAX_NODES];
19 printf("Enter the number of nodes in the graph: ");
20 scanf("%d", &numnodes);
21 for (i = 0; i < numnodes; i++) {
22 visited[i] = 0;
23 }
24 printf("Enter the adjacency matrix:\n");
25 for (i = 0; i < numnodes; i++) {
26 for (j = 0; j < numnodes; j++) {
27 scanf("%d", &graph[i][j]);
28 }
29 }
30 printf("DFS traversal: ");
31 for (i = 0; i < numnodes; i++) {
32 if (!visited[i])
33 dfs(graph, visited, i, numnodes);
34 }
35 printf("\n");
36 return 0;
37 }

```


Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4775390625 Kib
- Compilation Time: 0.36s

```

35°C Haze 13:05 18-05-2023 ENG IN

38) Write C programme to implement BFS.

The screenshot shows the Dev-C++ IDE interface with the following details:

- Project:** Untitled1
- File:** Untitled1.cpp
- Code Content:** C code for Breadth-First Search (BFS) using an adjacency matrix representation and a queue. The code includes functions for initializing graphs, adding edges, dequeuing vertices, and performing BFS traversal starting from a given vertex.
- Output Window:** Displays the command-line output of the program execution. It prompts for the number of vertices (6), edges (5), and edge details (1-2, 2-3, 3-4, 4-5, 5-6). It then starts the BFS traversal from vertex 6, listing the visited vertices (6, 1, 2, 3, 4, 5).
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings, outputting Untitled1.exe to C:\Users\saikr\OneDrive\Desktop.
- System Taskbar:** Shows system icons like Start, Search, Task View, File Explorer, Edge, and others, along with the date (18-05-2023) and time (13:08).

The screenshot shows the Dev-C++ IDE interface with the following details:

- Project:** Untitled1
- File:** Untitled1.cpp
- Code Content:** C code for Breadth-First Search (BFS) using an adjacency matrix representation and a queue. This version includes detailed enqueue and dequeue logic using pointers to front and rear of the queue.
- Output Window:** Displays the command-line output of the program execution. It prompts for the number of vertices (6), edges (5), and edge details (1-2, 2-3, 3-4, 4-5, 5-6). It then starts the BFS traversal from vertex 6, listing the visited vertices (6, 1, 2, 3, 4, 5).
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings, outputting Untitled1.exe to C:\Users\saikr\OneDrive\Desktop.
- System Taskbar:** Shows system icons like Start, Search, Task View, File Explorer, Edge, and others, along with the date (18-05-2023) and time (13:09).

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11
- Menu Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbar:** Includes icons for New, Open, Save, Build, Run, Stop, and others.
- Compiler Status Bar:** TDM-GCC 4.9.2 64-bit Release
- Project Explorer:** Shows the project structure with files: Graph.h, Queue.h, BFS.c, and main.c.
- Code Editor:** Displays the C code for BFS. The code initializes a graph, creates a queue, and performs a traversal starting from vertex 'src'. It prints the traversal path and the number of vertices visited.
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings. The output file is C:\Users\saikr\OneDrive\Desktop\Untitled1.exe, size 131.212890625 Kib, and compilation time 0.34s.
- System Tray:** Shows the date (18-05-2023), time (13:09), battery level (35°C Haze), and system icons.

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11
- Menu Bar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help
- Toolbar:** Includes icons for New, Open, Save, Build, Run, Stop, and others.
- Compiler Status:** TDM-GCC 4.9.2 64-bit Release
- Project Explorer:** Shows the file Untitled1.cpp under the Graph project. The code implements a BFS algorithm using a queue.
- Code Editor:** The main function reads vertices and edges from input, initializes the graph, and performs a BFS starting from a specified vertex.
- Compiler Tab:** Shows compilation results with 0 errors and 0 warnings.
- Status Bar:** Displays system information (35°C Haze), battery level (ENG IN), and the date/time (18-05-2023 13:10).

```
Graph struct
Queue struct
BFS (struct Graph* graph)
dequeue (struct Queue* queue)
enqueue (struct Queue* queue, int vertex)
initializeGraph (struct Graph* graph)
initializeQueue (struct Queue* queue)
isQueueEmpty (struct Queue* queue)
main () {
    struct Graph graph;
    int vertices, edges, src, dest;
    printf("Enter the number of vertices: ");
    scanf("%d", &vertices);
    initializeGraph(&graph, vertices);
    printf("Enter the number of edges: ");
    scanf("%d", &edges);
    for (int i = 0; i < edges; i++) {
        printf("Enter edge %d (source destination): ", i + 1);
        scanf("%d %d", &src, &dest);
        addEdge(&graph, src, dest);
    }
    int startvertex;
    printf("Enter the starting vertex for BFS: ");
    scanf("%d", &startvertex);
    BFS(&graph, startvertex);
}
return 0;
}
```

39) Write C programme to implement Min Heap.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEAP_SIZE 100
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void heapify(int arr[], int n, int i, int isMinHeap) {
    int largest = i; // Initialize largest as root
    int left = 2 * i + 1; // left child index
    int right = 2 * i + 2; // Right child index
    if (isMinHeap) {
        if (arr[i] > arr[largest])
            largest = i;
        if (left < n && arr[left] < arr[largest])
            largest = left;
        if (right < n && arr[right] < arr[largest])
            largest = right;
    } else {
        if (arr[i] < arr[largest])
            largest = i;
        if (left < n && arr[left] > arr[largest])
            largest = left;
        if (right < n && arr[right] > arr[largest])
            largest = right;
    }
    if (largest != i) {
        swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest, isMinHeap);
    }
}
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    scanf("%d", &isMinHeap);
    buildHeap(arr, n, 0, isMinHeap);
    printHeap(arr, n);
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KiB
- Compilation Time: 0.41s

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEAP_SIZE 100
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void buildHeap(int arr[], int n, int isMinHeap) {
    int i;
    int startIdx = (n / 2) - 1;
    for (i = startIdx; i >= 0; i--) {
        heapify(arr, n, i, isMinHeap);
    }
}
void printHeap(int arr[], int n) {
    int i;
    printf("Heap: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int n;
    int headType;
    int arr[MAX_HEAP_SIZE];
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    scanf("%d", &headType);
    buildHeap(arr, n, headType);
    printHeap(arr, n);
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KiB
- Compilation Time: 0.41s

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEAP_SIZE 100
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printHeap(int arr[], int n) {
    printf("Heap: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int n;
    int headType;
    int arr[MAX_HEAP_SIZE];
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    scanf("%d", &headType);
    buildHeap(arr, n, headType);
    printHeap(arr, n);
    return 0;
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KiB
- Compilation Time: 0.41s

40) Write C programme to implement Max Heap.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEAP_SIZE 100
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void heapify(int arr[], int n, int i, int isMinHeap) {
    int largest = i; // Initialize largest as root
    int left = 2 * i + 1; // Left child index
    int right = 2 * i + 2; // Right child index
    if (isMinHeap) {
        if (arr[i] > arr[largest])
            largest = i;
        if (left < n && arr[left] < arr[largest])
            largest = left;
        if (right < n && arr[right] < arr[largest])
            largest = right;
    } else {
        if (arr[i] < arr[largest])
            largest = i;
        if (left < n && arr[left] > arr[largest])
            largest = left;
        if (right < n && arr[right] > arr[largest])
            largest = right;
    }
    if (largest != i) {
        swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest, isMinHeap);
    }
}
void buildHeap(int arr[], int n, int isMinHeap) {
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i, isMinHeap);
}
void printHeap(int arr[], int n) {
    int i;
    printf("Heap: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int i;
    int n, heapType;
    int arr[MAX_HEAP_SIZE];
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    if (scanf("%d", &heapType) == 1)
        buildHeap(arr, n, 1);
    else
        buildHeap(arr, n, 0);
    printHeap(arr, n);
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KiB
- Compilation Time: 0.34s

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEAP_SIZE 100
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void heapify(int arr[], int n, int i, int isMinHeap) {
    int largest = i; // Initialize largest as root
    int startIdx = (n / 2) - 1;
    for (int i = startIdx; i >= 0; i--) {
        heapify(arr, n, i, isMinHeap);
    }
}
void buildHeap(int arr[], int n, int isMinHeap) {
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i, isMinHeap);
}
void printHeap(int arr[], int n) {
    int i;
    printf("Heap: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int i;
    int n, heapType;
    int arr[MAX_HEAP_SIZE];
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    if (scanf("%d", &heapType) == 1)
        buildHeap(arr, n, 1);
    else
        buildHeap(arr, n, 0);
    printHeap(arr, n);
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KiB
- Compilation Time: 0.34s

C:\Users\saikr\OneDrive\Desktop\Untitled1.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug Untitled1.cpp

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

void buildHeap(int arr[], int n);
void heapify(int arr[], int i);
void printHeap(int arr[], int n);
void swap(int *a, int *b);

int main() {
    int n, heapType;
    int arr[MAX_HEAP_SIZE];
    int i;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter the elements: ");
    for (i=0;i<n;i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter 1 for Min Heap or 2 for Max Heap: ");
    scanf("%d", &heapType);

    buildHeap(arr, n, heapType == 1);
    printHeap(arr, n);
}

void buildHeap(int arr[], int n, int heapType) {
    int i;
    printHeap(arr, n);
    for (i=n/2;i>0;i--) {
        heapify(arr, i, heapType);
    }
}

void heapify(int arr[], int i, int heapType) {
    int l = 2*i;
    int r = 2*i + 1;
    int largest = i;
    if (l < n && arr[l] > arr[i]) {
        largest = l;
    }
    if (r < n && arr[r] > arr[largest]) {
        largest = r;
    }
    if (largest != i) {
        swap(&arr[i], &arr[largest]);
        heapify(arr, largest, heapType);
    }
}

void printHeap(int arr[], int n) {
    int i;
    for (i=0;i<n;i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\saikr\OneDrive\Desktop\Untitled1.exe
- Output Size: 130.4033203125 KB
- Compilation Time: 0.34s

Abort Compilation Shorten compiler paths

33°C Haze

Search

12:15 22-05-2023