

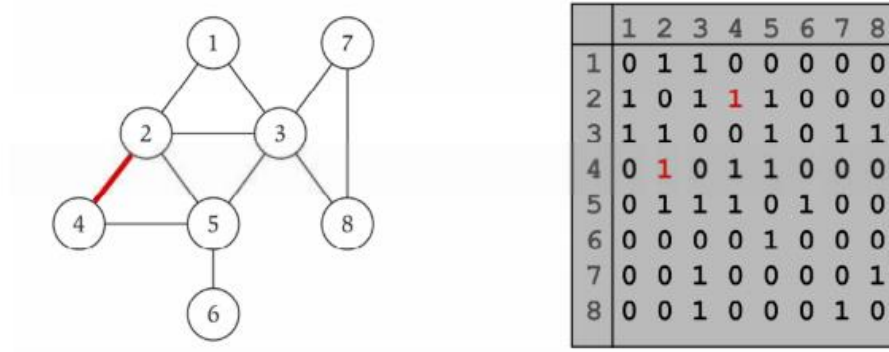
Bandana Irmal Abdillah

140810180025

Tugas 6

1.

Dengan menggunakan *undirected graph* dan *adjacency matrix* berikut, buatlah koding programnya menggunakan bahasa C++.



```
1.  /*
2.  Nama      : Bandana Irmal Abdillah
3.  NPM       : 140810180025
4.  Tugas 6 - No 1
5.  */
6.
7.  #include<iostream>
8.  using namespace std;
9.
10. int main(){
11.     int matrix[8][8] = {
12.         {0,1,1,0,0,0,0,0},
13.         {1,0,1,1,1,0,0,0},
14.         {1,1,0,0,1,0,1,1},
15.         {0,1,0,1,1,0,0,0},
16.         {0,1,1,1,0,1,0,0},
17.         {0,0,0,0,1,0,0,0},
18.         {0,0,1,0,0,0,0,1},
19.         {0,0,1,0,0,0,1,0}
20.     };
21.     for(int i = 0; i < 8; i++){
22.         for(int j = 0; j<8;j++){
23.             cout << matrix[i][j] << " ";
24.         }
25.         cout << endl;
26.     }
27. }
```

```

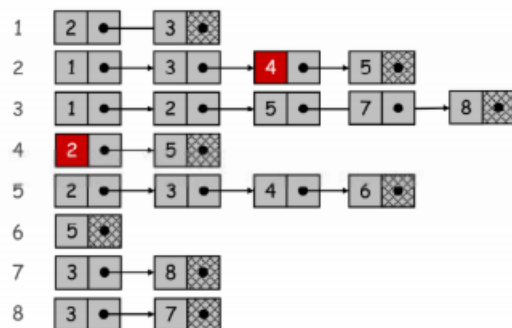
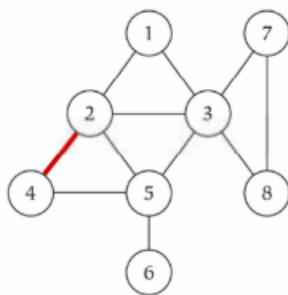
D:\_CobaanHidup\Semester 4 IP 4\Analisis Algoritma\Praktikum\Anal
1 1 0 0 0 0 0
0 1 1 1 0 0 0
1 0 0 1 0 1 1
1 0 1 1 0 0 0
1 1 1 0 1 0 0
0 0 0 1 0 0 0
0 1 0 0 0 0 1
0 1 0 0 0 1 0

-----
rocess exited after 0.267 seconds with return value 0
ress any key to continue . . .

```

2.

Dengan menggunakan *undirected graph* dan representasi *adjacency list*, buatlah koding programnya menggunakan bahasa C++.



```

1.  /*
2.  Nama      : Bandana Irmal Abdillah
3.  NPM       : 140810180025
4.  Tugas 6 - No 2
5.  */
6.
7.  #include<iostream>
8.  #include<windows.h>
9.  using namespace std;
10.
11. struct adjacent{
12.     int nodeAdj;
13.     adjacent* nextAdj;
14. };
15.
16. struct elemen{
17.     int node;
18.     elemen* next;
19.     adjacent* firstAdj;
20. };
21.
22. typedef elemen* pointerNode;
23. typedef adjacent* pointerAdj;
24. typedef pointerNode list;

```

```

25.
26. void createListNode(list& first);
27. void createNode(pointerNode& pBaru, int vertex);
28. void createAdjacent(pointerAdj& pBaru, int vertex);
29. void insertAdjacent(pointerNode& curNode, pointerAdj pBaruAdj);
30. void insertElement(list& first, pointerNode pBaruNode, int size);
31. void output(list first);
32.
33. int main(){
34.     list first;
35.     pointerNode node;
36.
37.     createListNode(first);
38.
39.     createNode(node,1);
40.     insertElement(first,node,2);
41.     createNode(node,2);
42.     insertElement(first,node,4);
43.     createNode(node,3);
44.     insertElement(first,node,5);
45.     createNode(node,4);
46.     insertElement(first,node,2);
47.     createNode(node,5);
48.     insertElement(first,node,4);
49.     createNode(node,6);
50.     insertElement(first,node,1);
51.     createNode(node,7);
52.     insertElement(first,node,2);
53.     createNode(node,8);
54.     insertElement(first,node,2);
55.     output(first);
56.     system("pause");
57. }
58.
59.
60.
61.
62.
63. void createListNode(list& first){
64.     first = NULL;
65. }
66.
67. void createNode(pointerNode& pBaru,int vertex){
68.     pBaru = new elemen;
69.     pBaru->node = vertex;
70.     pBaru->next = NULL;
71.     pBaru->firstAdj = NULL;
72. }
73.
74. void createAdjacent(pointerAdj& pBaru,int vertex){
75.     pBaru = new adjacent;
76.     pBaru->nodeAdj = vertex;
77.     pBaru->nextAdj = NULL;
78. }
79.
80. void insertAdjacent(pointerNode& curNode,pointerAdj pBaruAdj){
81.     pointerAdj last;
82.     if(curNode->firstAdj == NULL){
83.         curNode->firstAdj = pBaruAdj;
84.     }else{
85.         last = curNode->firstAdj;

```

```

86.         while(last->nextAdj != NULL){
87.             last = last->nextAdj;
88.         }
89.         last->nextAdj = pBaruAdj;
90.     }
91. }
92.
93. void insertElement(list& first, pointerNode pBaruNode, int size){
94.     pointerNode last;
95.     pointerAdj pBaruAdj;
96.     if(first == NULL){
97.         first = pBaruNode;
98.     }else{
99.         last = first;
100.        while(last->next != NULL){
101.            last = last->next;
102.        }
103.        last->next = pBaruNode;
104.    }
105.    if(size>0){
106.        cout<<"Masukan node yang berhubungan dengan "<<pBaruNode-
>node<<" : "<<endl;
107.    }
108.    for(int i = 0; i < size; i++){
109.        int vertex;
110.        cin>>vertex;
111.        createAdjacent(pBaruAdj,vertex);
112.        insertAdjacent(pBaruNode,pBaruAdj);
113.    }
114. }
115.
116. void output(list first){
117.     pointerNode pOut;
118.     pointerAdj pOutAdj;
119.     if(first == NULL){
120.         cout<<"Tidak ada Node"<<endl;
121.     }else{
122.         pOut = first;
123.
124.         while(pOut != NULL){
125.             cout<<"Parent = "<<pOut->node<<endl;
126.             if(pOut->firstAdj == NULL){
127.                 cout<<"Tidak ada adjacency"<<endl;
128.             }else{
129.                 pOutAdj = pOut->firstAdj;
130.                 cout<<"Child = ";
131.                 while(pOutAdj != NULL){
132.                     cout<<pOutAdj->nodeAdj<<" ";
133.                     pOutAdj = pOutAdj->nextAdj;
134.                 }
135.             }
136.             cout<<endl;
137.             pOut = pOut->next;
138.
139.         }
140.     }
141. }

```

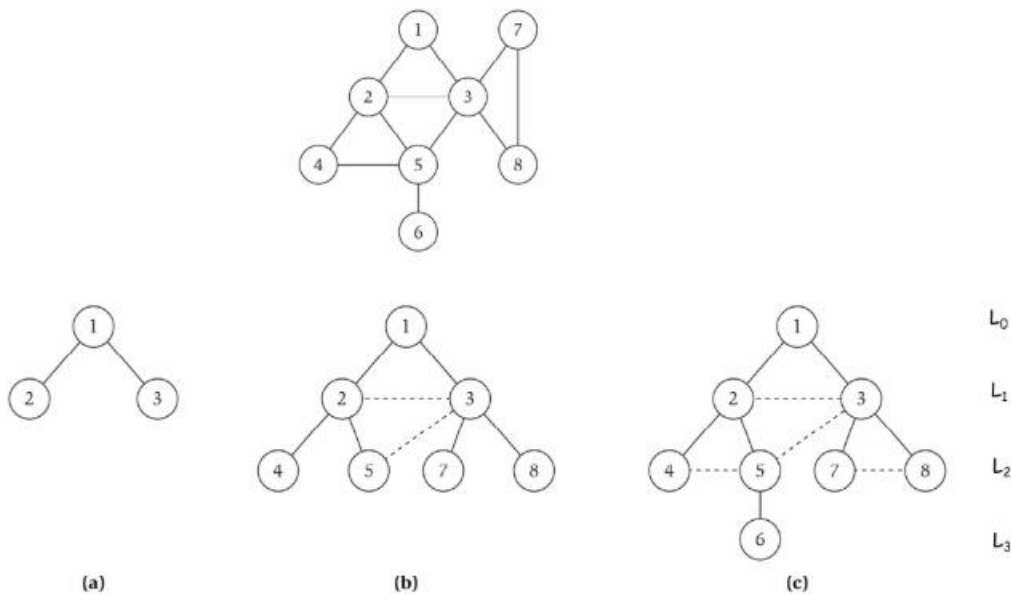
```

D:\CobaanHidup\Semester 4 IP 4\Analisis Algoritma\BFS
Masukan node yang berhubungan dengan 1 : 2 3
Masukan node yang berhubungan dengan 2 : 1 3 4 5
Masukan node yang berhubungan dengan 3 : 1 2 5 7 8
Masukan node yang berhubungan dengan 4 : 2 5
Masukan node yang berhubungan dengan 5 : 2 3 4 6
Masukan node yang berhubungan dengan 6 : 5
Masukan node yang berhubungan dengan 7 : 3 8
Masukan node yang berhubungan dengan 8 : 3 7
Parent = 1
Child = 2 3
Parent = 2
Child = 1 3 4 5
Parent = 3
Child = 1 2 5 7 8
Parent = 4
Child = 2 5
Parent = 5
Child = 2 3 4 6
Parent = 6
Child = 5
Parent = 7
Child = 3 8
Parent = 8
Child = 3 7
Press any key to continue . . .

```

3.

Buatlah program Breadth First Search dari algoritma BFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan *undirected graph* sehingga menghasilkan tree BFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big- Θ !



```

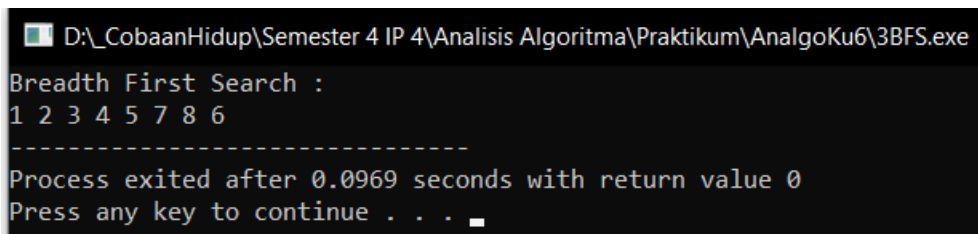
1.
2. /*
3.  Nama      : Bandana Irmal Abdillah
4.  NPM       : 140810180025
5.  Tugas 6 - No 3
6.  */
7. #include<iostream>
8. using namespace std;
9. int main(){
10.     int vertexSize = 8;
11.     int adjacency[8][8] = {

```

```

12.     {0,1,1,0,0,0,0,0},
13.     {1,0,1,1,1,0,0,0},
14.     {1,1,0,0,1,0,1,1},
15.     {0,1,0,0,1,0,0,0},
16.     {0,1,1,1,0,1,0,0},
17.     {0,0,0,0,1,0,0,0},
18.     {0,0,1,0,0,0,0,1},
19.     {0,0,1,0,0,0,1,0}
20. };
21. bool discovered[vertexSize];
22. for(int i = 0; i < vertexSize; i++){
23.     discovered[i] = false;
24. }
25. int output[vertexSize];
26.
27. //inisialisasi start
28. discovered[0] = true;
29. output[0] = 1;
30.
31. int counter = 1;
32. for(int i = 0; i < vertexSize; i++){
33.     for(int j = 0; j < vertexSize; j++){
34.         if((adjacency[i][j] == 1)&&(discovered[j] == false)){
35.             output[counter] = j+1;
36.             discovered[j] = true;
37.             counter++;
38.         }
39.     }
40. }
41. cout<<"Breadth First Search : "<<endl;
42. for(int i = 0; i < vertexSize; i++){
43.     cout<<output[i]<<" ";
44. }
45. }

```



```

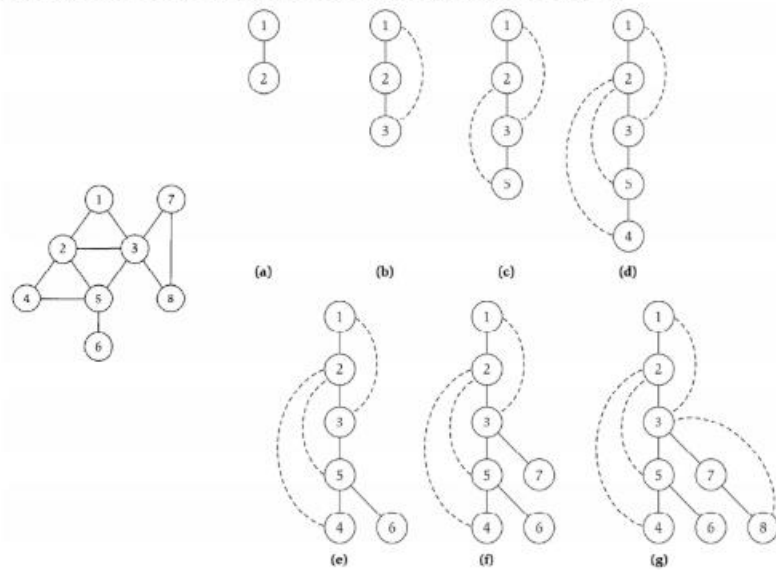
D:\_CobaanHidup\Semester 4 IP 4\Analisis Algoritma\Praktikum\AnalgoKu6\3BFS.exe
Breadth First Search :
1 2 3 4 5 7 8 6
-----
Process exited after 0.0969 seconds with return value 0
Press any key to continue . . .

```

- BFS merupakan metode pencarian secara melebar sehingga mengunjungi node dari kiri ke kanan di level yang sama. Apabila semua node pada suatu level sudah dikunjungi semua, maka akan berpindah ke level selanjutnya. Dalam worst case BFS harus mempertimbangkan semua jalur (path) untuk semua node yang mungkin, maka nilai kompleksitas waktu dari BFS adalah $O(|V| + |E|)$.

4.

Buatlah program Depth First Search dari algoritma DFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan *undirected graph* sehingga menghasilkan tree DFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!



```

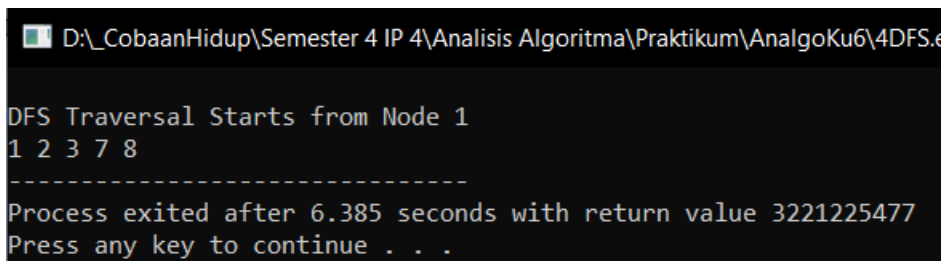
1.  /*
2.  Nama      : Bandana Irmal Abdillah
3.  NPM       : 140810180025
4.  Tugas 6 - No 4
5.  */
6.
7.  #include <iostream>
8.  #include <list>
9.
10. using namespace std;
11.
12. class Graph{
13.     int N;
14.
15.     list<int> *adj;
16.
17.     void DFSUtil(int u, bool visited[]){
18.         visited[u] = true;
19.         cout << u << " ";
20.
21.         list<int>::iterator i;
22.         for(i = adj[u].begin(); i != adj[u].end(); i++){
23.             if(!visited[*i]){
24.                 DFSUtil(*i, visited);
25.             }
26.         }
27.     }
28.
29. public :
30.     Graph(int N){
31.         this->N = N;
32.         adj = new list<int>[N];
33.     }
34.

```

```

35. void addEdge(int u, int v){
36.     adj[u].push_back(v);
37. }
38.
39. void DFS(int u){
40.     bool *visited = new bool[N];
41.     for(int i = 0; i < N; i++){
42.         visited[i] = false;
43.     }
44.     DFSUtil(u, visited);
45. }
46. };
47.
48. int main(){
49.     Graph g(8);
50.
51.
52.     g.addEdge(1,2);
53.     g.addEdge(1,3);
54.     g.addEdge(2,3);
55.     g.addEdge(2,4);
56.     g.addEdge(2,5);
57.     g.addEdge(3,7);
58.     g.addEdge(3,8);
59.     g.addEdge(4,5);
60.     g.addEdge(5,3);
61.     g.addEdge(5,6);
62.     g.addEdge(7,8);
63.
64.     cout << "\nDFS Traversal Starts from Node 1" << endl;
65.     g.DFS(1);
66.
67.     return 0;
68. }

```



The screenshot shows a terminal window with the following output:

```

D:\CobaanHidup\Semester 4 IP 4\Analisis Algoritma\Praktikum\AnalgoKu6\4DFS.e
DFS Traversal Starts from Node 1
1 2 3 7 8
-----
Process exited after 6.385 seconds with return value 3221225477
Press any key to continue . . .

```

- DFS merupakan metode pencarian mendalam, yang mengunjungi semua node dari yang terkiri lalu geser ke kanan hingga semua node dikunjungi. Kompleksitas ruang algoritma DFS adalah $O(bm)$, karena kita hanya perlu menyimpan satu buah lintasan tunggal dari akar sampai daun, ditambah dengan simpul-simpul saudara kandungnya yang belum dikembangkan.