Networks and Graphs

When you say the words networks, trees, graphs, what do you think about?

Network is a word that you are used to hear in concrete settings: a computer network, a network of friends, social network, social networking sites… Trees will surely make you think to what you see in your backyard, parks and forests but also to genealogical trees. And graphs probably invoke memories of algebra. In fact network, trees and graphs are very important concepts in computer science and the branch of mathematics that studies those concepts is called graph theory.

The words networks and graphs are often used interchangeably, the work networks being used more often in concrete applications and a graph being often viewed as the abstract representation of the network. We will see later on that trees are special graphs.

Let briefly discuss computer networks and social networks and see how they share some common ideas. A **computer network** is a collection of computers (or other computing devices such as printers, routers, etc.) that are connected. The computers or other computing devices are the nodes of the network and the connection cables are the links.

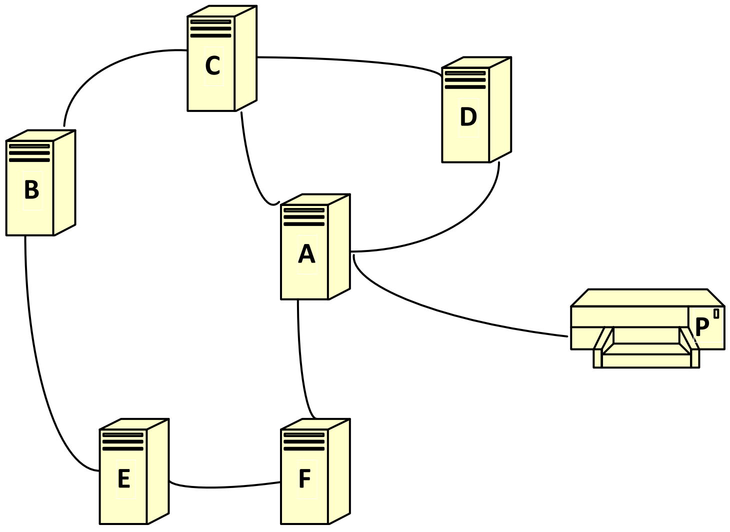


Figure 1- A simple computer network

The simple computer network depicted in Figure 1 could be the computer network of a small business where all computers share a common printer. Notice a few features of this network. First, as long as all links and all computers function, all computers can communicate with each other and they can all send documents to the printer. While computer A can send a document directly to printer P, a document issued from computer B must go through at least two other computers to get to printer P. If A malfunctions or is removed or if the connection between A and P malfunctions or is removed then none of the remaining computers can communicate with the printer P.

Social networks are used to study relationships between individuals or organizations. Social networks analysis was develop as an important technique in sociology. The use of modern mathematics techniques combined with powerful computers has helped spread the use of social networks to many other disciplines such as economics, biology, and information science. In a social network individuals or organizations depending of the type of interaction studied are the nodes. Nodes are linked according to a specific type of relationship or interdependence.

Let suppose that we want to model interactions at the beginning of group project assignment. A link between two individuals means that they have discussed the assignment together. We could interpret the network in Figure 2 in different ways. For example, P is the professor and has discussed the assignment with student A only, who in turn has discussed the assignment with students C, D, and F. We can see that if professor P gave instructions to A in his discussion, B will hear from them only through either A then C or A then F then E. Student A plays a major role, since she/he is the only connection between the other students and the professor.

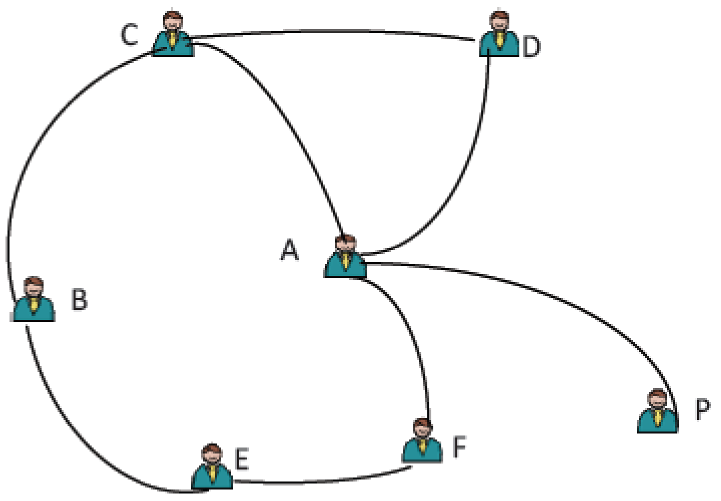


Figure 2- A simple social network

In both examples, the computer network and the social network, we can abstract a mathematical structure that is identical and that represent the fundamental concepts: the nodes and the links between the nodes. The mathematical structure is called a graph. Figure 3 shows the graph that would represent the networks of Figure 1 and Figure 2.

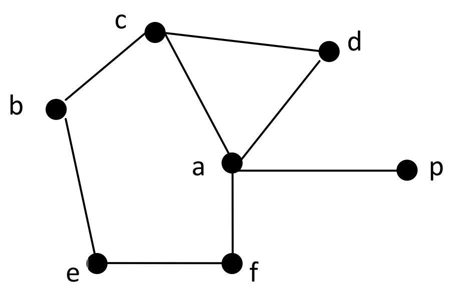


Figure 3- Graph representing simple networks

A graph is a mathematical structure that consists of a set (or collection) of nodes called vertices and a set of links between vertices that are called edges. For the graph shown on figure 3 we will denote the set of vertices by $\mathcal{V}$= {a, b, c, d, e, f, p} and the set of edges by $\mathcal{E}$ = {ac, bc, cd, da, ap, ef, af, be}. The order in which we list the vertices or the edges in the sets do not matter and the order in which we list the vertices of the edges do not matter either. For example ac and ca represent the same edge.

Studying properties of graphs help solve problems related to computer networks, diverse other computer science problems, social network analysis, and problems in many other seemingly unrelated applied fields.

Graphs: definitions and properties

A **graph** consists of a set of **vertices** and a set of **edges**. An edge is itself a set of two vertices called the **endpoints** of the edge.
We draw the graph by representing the vertices by dots and drawing lines (not necessarily straight) between endpoints of an edge. So for example, if the set of vertices of G is $\mathcal{V}$= {a, b, c, d, e} and the set of edges is $\mathcal{E}$ = {*ab, ce, de, be*} then the graph can been drawn as shown in Figure 4 and in Figure 5.
.

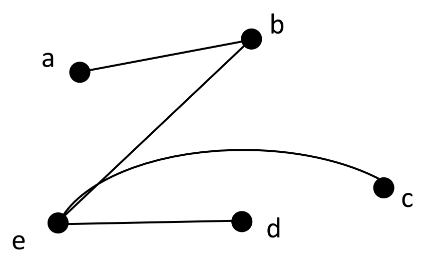


Figure 4- a representation of graph G

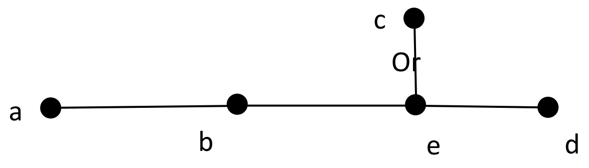


Figure 5- a representation of graph G

We can reposition the vertices in any ways we like, as long as two vertices are connected by a line if and only if they are endpoints of one edge. We then get a representation of the graph.

Remarks:
- There is at most one edge between any two vertices.
- The endpoints of an edge are always distinct.
- An edge has no direction. So we can write its endpoints in any order. The edge *ab* is the same as the edge *ba*.

In the social networking class activity, we introduced some definitions that we will repeat here and we will add a few more terms. All definitions are illustrated with examples using the graph G depicted in Figure 5.
Two vertices that are endpoints of the same edge are said to be **adjacent**. An edge is said to be **incident** with its endpoints and a vertex is **incident** with and edge for which it is an endpoint.
Example: In graph G vertices a and b are adjacent, but b and c are not because ab is an edge of the graph while bc is not. Edge *be* is incident with b and vertex c is incident with edge *ec*.

The **degree** of a vertex is the number of edges incident with that vertex. We will denote the degree of the vertex v as deg(v).
Example: deg(a) = a, deg(b) = 2. Find the degrees of the other vertices of G.

A **path** is a sequence of adjacent vertices. If the first vertex in the sequence is u and the last one is v, we say that the **path is a path between u and v**.
Example: *abec* is a path between a and c. *bed* is a path between b and d.

A graph is **connected** if there is path between any two vertices of the graph.
Example: is the graph G connected? To answer this question, try to list a path between any pairs of vertices.

Path between a and b:
Path between a and c:
Path between a and d:
Path between a and e:
Path between b and c:
Path between b and d:
Path between b and e:
Path between c and d:
Path between c and e:
Path between d and e:
Conclusion: is the graph G connected:

Remark: If there is a path between a and b then there is a path between b and a. How do we find it?
Example: the graph H shown in Figure 6 is not connected. Why?

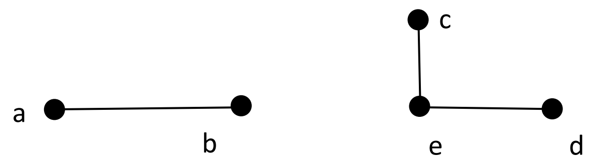


Figure 6- Graph H

The **length** of a path is one less than the number of vertices in the path. This corresponds to the number of edges the path goes through. This is illustrated in Figure 7 below.

The edges in the path *abed* are highlighted in red (thick lines if you see this in black and white). The path has 4 vertices and 3 edges. The length is 3.

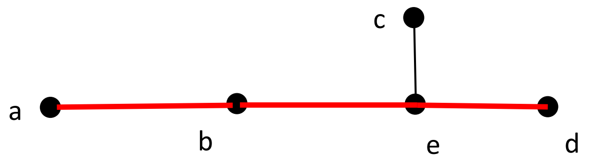


Figure 7- Length of path

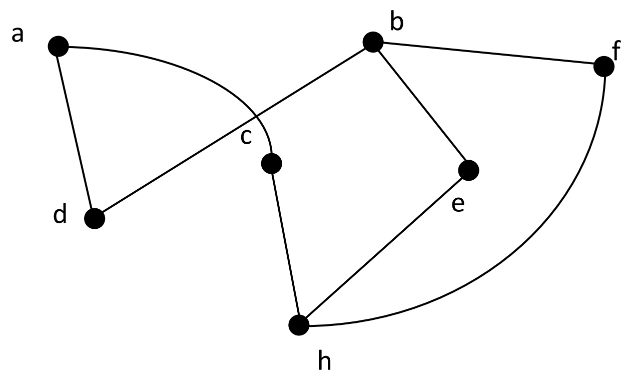
The **distance** between two vertices is the length of the shortest path between those two vertices. The distance between vertices u and v will be denoted d(u,v).
Example: d(a, e) = 2, d (a, d) = 3.

If there is no path between two vertices u and v then the distance between them is infinite. We write d(u, v) = ∞. The distance between a vertex and itself is 0: d(u, u) = 0.
Example: in graph H from Figure 6, d(b, e) = ∞. Can you give other pairs of vertices in graph H that are at infinite distance from each other?

Exercises

1. Draw the graph with vertex set $\mathcal{V}$= {a, b, c, d, e, f} and set of edges is $\mathcal{E}$ = {*ac, be, de, cd, , ba, fb*}.

2. Consider the graph G drawn below.



a) Give the vertex set of the G.
b) Give the set of edge of G.
c) Find the degree of each of the vertices of the graph G.
d) Add the degrees of all the vertices of the graph G and compare that sum to the number of edges in G, what do you find?
e) Give a path of length 1, of length 2, and of length 3 in G.
f) Find the longest path you can in G?

(Remember that you cannot repeat vertices).

3. Draw a graph with 6 vertices and 6 edges that is connected and a graph with 6 vertices and 6 edges that is not connected.

4. Consider the graph G from exercise 2. Suppose that this graph represents the relationship "is friend of" among 7 students that are part of the same social network.
Suppose you want to propagate a news item among this group of student, but you can communicate the item only to one student. You want the news to propagate as fast as possible (assume that the time it takes to reach a student is proportional to the number of students it has to go through). To which student should you communicate the item, assuming that the item will then be propagated within this network? Explain how you select the student.

5. Draw a graph that represents a social network where you think news would propagate very slowly. What are some of the characteristics of this graph?

Cycles and Trees

A **cycle** is a sequence of adjacent vertices that are distinct except the first and last vertices which are the same.

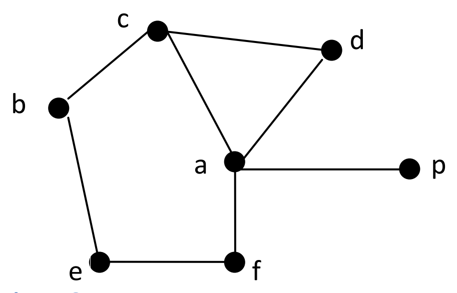


Figure 8

For example, acda , bcafeb, or ebcdafe are cycles of the graph depicted in Figure 8. The length of a cycle is the number of distinct vertices in it. So acda has length 3, bcafeb has length 5, and ebcdafe has length 6. Note the length of a cycle is the number of edges that link the vertices in the cycle.

A **tree** is a connected graph which has no cycle. The graph shown in Figure 9 is a tree. This tree has 7 vertices and 6 edges. If we remove the edge af, without removing the vertices, the graph is no longer connected. This would be true for the removal of any other edge.

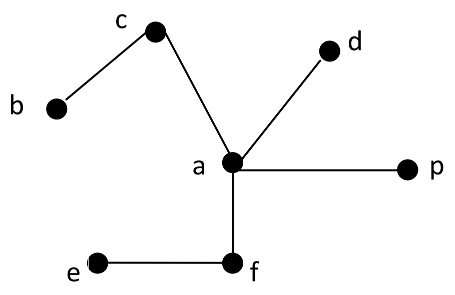


Figure 9- a tree T

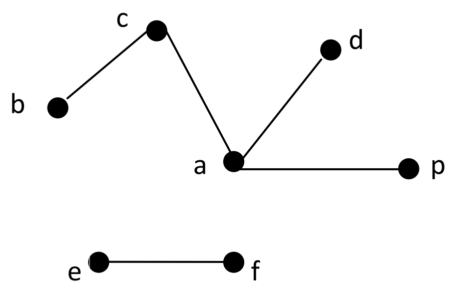


Figure 10- T minus edge af

Some properties of trees

- A tree has one fewer edges than vertices.
- Any tree has at least two vertices of degree one. We call the vertices of degree one leaves.
- If we remove one edge from a tree (without removing the vertices), the resulting graph is not connected.
- There is exactly one path between any two vertices in a tree.

Exercises

6. Draw the following trees or say why they do not exist.
   a) A tree with 8 vertices and 7 edges.
   b) A tree with 6 vertices. The degrees of the vertices are 3, 2, 2, 2, 2, 1.
   c) A tree with 6 vertices. The degrees of the vertices are 3, 2, 2, 1, 1, 1.

7. A club with 15 members establishes a "telephone tree" to propagate information through its membership. This can be represented by a tree, let call it T, in the graph theory sense of the word, where two members are joined by an edge if one of them is charged to call the other one in the telephone tree. Information will come from the club president, A.
   a) The telephone tree will satisfy the rule that no member is in contact with more than three other members through the telephone tree (i.e. is called or call another member). What does it say about the tree T?
   b) Draw the tree T so that information from A will reach any club member in the least possible number of phone calls.
   c) For your tree T, what is the maximum number of phone calls necessary for information from A to reach a member?

Breadth-First Search Algorithm

An **algorithm** is an unambiguous step-by-step finite set of instructions to solve a problem. Some of our more precise Alice storyboards such as the example shown in Figure 11 were algorithms.

```
Parameter: who
Do in order
        dragon takes off
        dragon turns to face who
        Repeat Distance of who in front of dragon number of times
                dragon fly
```

Figure 11- Alice Storyboard

The problem-solving process of in dealing with computer problems will frequently involve either designing an algorithm and/or choosing one or more known algorithms, then implement the algorithms. We will study one such algorithm.

The **breadth-first search** algorithm finds a spanning tree in a graph. A spanning tree T of a graph G is a tree that has the same vertices as the graph G and such that all the edges of T are also edges of G (some edges of G might not be edges of T). If a graph is connected then it has at least one spanning tree.
The breadth-first search algorithm can be used for different purposes:

- verify that a graph is connected;
- find a spanning tree of a graph;
- find the distances from a given vertex to any other vertex in the graph.

Distance: the distance between two vertices in a graph is the length of the shortest path between those two vertices. If there is no path between the two vertices, we say that the distance is infinite.

The idea of the breadth-first search algorithm is simple. The input is a graph G and an initial vertex. We will consider three lists:

1. A list U of unlabeled vertices: This list will consist of all the vertices of G at the beginning of the process and, if the graph is connected, it will be empty at the end of the process.
2. A list L of labeled vertices: This list will be empty at the beginning and, if the graph is connected, it will contain all the vertices of G at the end of the process.
3. A list E of edges: This list is empty at the beginning and, at the end of the process, if the graph is connected, it will contain all the edges of the spanning tree.

The label of a vertex mentioned in 2 corresponds to the distance between the initial vertex and the labeled vertex. We will label the initial vertex 0 and remove this vertex from U and place it in list L. Then we will visit, one at a time all the vertices adjacent to the initial vertex, label those vertices with a 1, remove them from U and place them in L. Also for each vertex adjacent to the initial vertex, we place the edge joining it to the initial vertex in E. We will illustrate the algorithm on the graph G shown in

Figure 12. The first stage of the algorithm is shown in Figure 13. The dashed red edges are the edges placed in E.

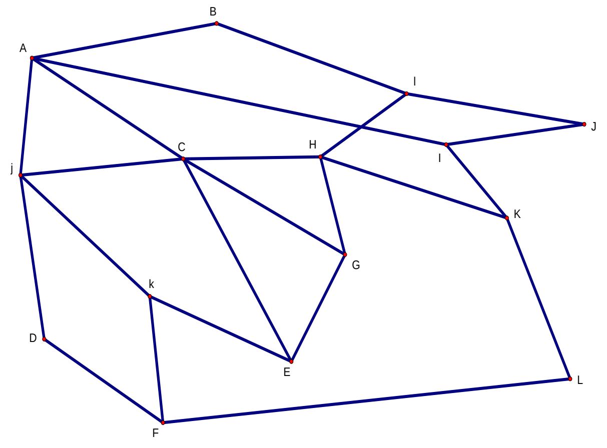


Figure 13- graph G

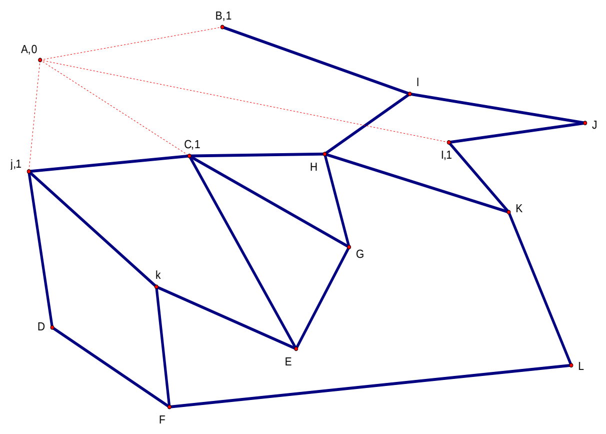


Figure 12- Breadth-first search algorithm, first labeling stage

Once we have visited all the vertices adjacent to the initial vertex (labeled 0), we visit, one at a time, all vertices that are still in U and are adjacent to some vertices labeled 1. For such a vertex, we first consider one edge from a vertex labeled 1 in U to that vertex and add it to E. Then we label the vertex 2, remove it from U and add it to L. The result from this stage is shown in Figure 14. The edges added to E in this stage are shown as thin green lines.

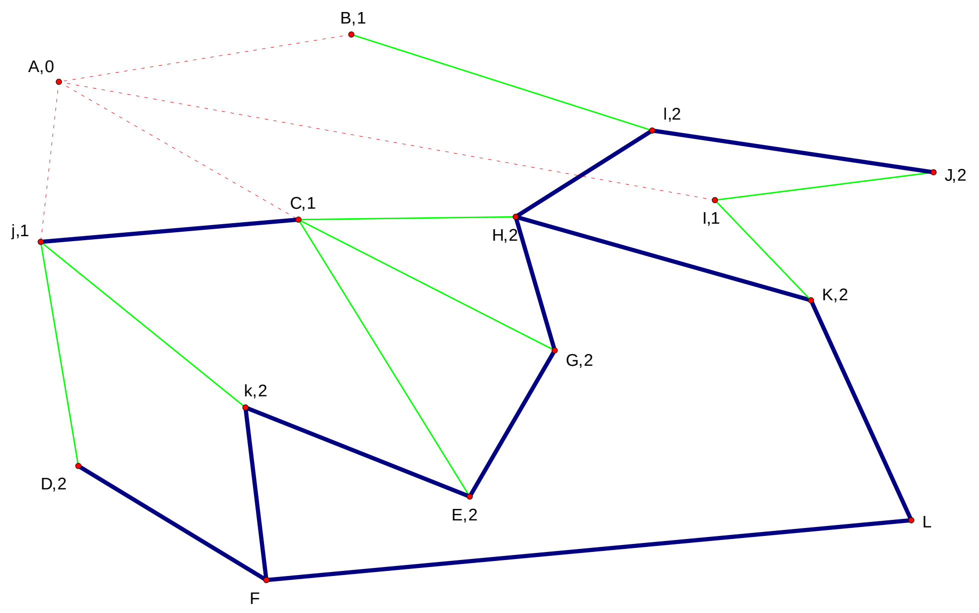


Figure 14- Breadth-first search, second labeling stage

When all vertices adjacent to vertices labeled 1 have been labeled 2, we move to label 3, and so on, until we run out of vertices or there is no edge between the vertices in L and the vertices in U.
There is a lot of imprecision in our algorithm description as illustrated by the last sentence above. We need to write the algorithm more precisely.

Breadth-first search algorithm

```
Input: graph G, initial vertex: a vertex in G
Initialization:
    Set lists L and E to be empty lists.
    Set list U to a list containing all vertices of G.
    Set label k to 0.
Step 1:
    1.a.  Label the initial vertex with label k (0).
    1.b.  Remove the initial vertex from U.
    1.c.  Add the initial vertex to L.
Step 2:
    Repeat while there is a vertex in U adjacent to a vertex in L
        2.a.  Increase k by 1
        2.b.  Repeat while there is vertex in U adjacent to a vertex labeled (k-1) in L
                2.b.i.    Choose a vertex in U adjacent to a vertex labeled (k-1) in L. Call this vertex u.
                2.b.ii.   Choose a vertex labeled (k-1) in L that is adjacent to u. Call this vertex v.
                2.b.iii.  Add the edge uv to E. Label the vertex u with the label k.
                2.b.iv.   Remove u from U.
                2.b.v.    Add u to L.
Step 3:
    If U is empty
        The graph with vertex set L and edge set E is a spanning tree of G
        The label of each vertex gives the distance between that vertex and the initial vertex
    else
        The graph G is disconnected and does not have a spanning tree
        The label of each labeled vertex gives the distance between that vertex and the initial vertex
```

Notes:

- In step 2, when there are several choices for the vertices u and/or v we will always select the vertices that are lowest in the alphabetical order.
- The initialization and step 1 starts the process and step 3 gives the conclusion of the process. Most of the “work” of the algorithm is done in step 2.
- Indented statements are statements that are executed sequentially. For example, when the condition in step 2.b. is true, i.e. there is vertex in U adjacent to a vertex labeled (k-1) in L, statements 2.b.i. to 2.b.v. are executed sequentially then the condition of the repeat while statement 2.b. is checked again. If it is again true we repeat statements 2.b.i. to 2.b.v. otherwise we go back to check the condition “there is a vertex in U adjacent to a vertex in L”.

Figure 15 illustrates the entire process. The spanning tree is shown in Figure 16.

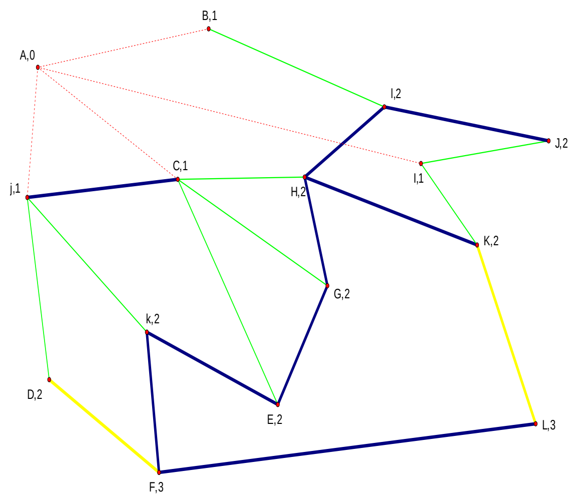


Figure 16- Complete breadth-first search algorithm

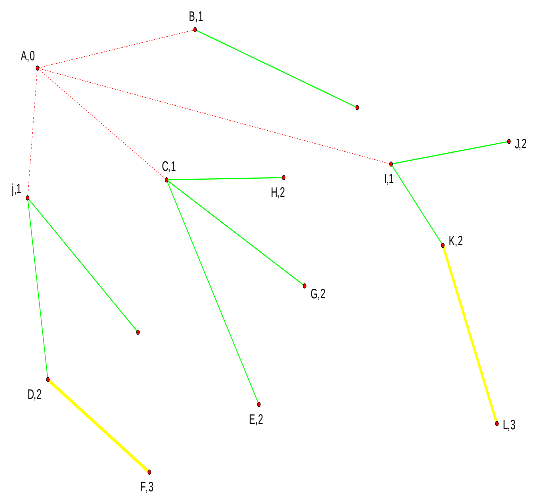


Figure 15- Spanning tree