# MEETING 12 - FUNDAMENTALS OF GRAPH THEORY

Graphs: Structures on sets. Computer science and software engineering can fundamentally be regarded as applied discrete mathematics. One of the best illustrations of this is that CS & SE utilize the concept of a *graph* in many situations. Now we have a *graph* whenever we have a situation where several entities, known as *vertices*, or *nodes*, that are related to each other in some way. We then say that the relation between two nodes or vertices constitute an *edge* and we can give this a pictorial representation by drawing the whole situation:
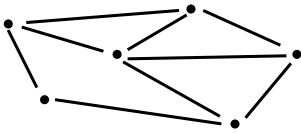
This could represent a range of different situations: each dot could be a router in a wide-area network, and each edge between the nodes could represent that there is a connection between the routers (they know of each other's IP-addresses). We will now use the words *vertex* and *edge* to denote the building blocks of a graph. Another instance is state-diagrams of digital automatons, each vertex would then represent a state that the automaton can assume and an edge between two vertices/states would then denote that there is a state-transition present between two states/vertices. If we draw a graph in the form of a tree (which has no closed circuits) we could draw it in the following manner:
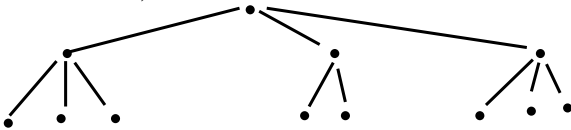

Here the vertices could represent folders or files in a file-system and an edge being present between two vertices/files/folders could represent that one folder/file is contained in the other, we can talk about parent vertices. The same graph could be used to represent hierarchies of executing processes in an a model of a running operating system. In another instance, the same model could be used to describe the architecture of a software systems built of components, where each component can be made up of other components (such as function or libraries of functions).

The possibilities are endless and studying graphs are of utmost importance for any student of discrete athematics and/or computer science and software engineering.

We will make a formal definition of what exactly a graph is:

Definition: A *graph* is a pair (V, E) of sets, V nonempty and each element of E a set of two distinct elements of V. The elements of V are called *vertices*; the elements of E are called *edges*. Thus, if e is an edge, then $e = \{v, w\}$, where v, w are different elements of V called the *end vertices* or *ends* of e. We say that an edge $e = \{v, w\}$ *joins* the veritces v, w and then we usually abandon set notation and write vw for e; and this is then the same edge as wv. The vertices v, w are said to be *incident* with the edge vw; the edge vw is *incident* with each vertex, v and w. Two vertices v, w are *adjacent* if they are vertices of the same edge; two edges are *adjacent* if they have a vertex in common. The number of edges incident with a vertex v is called the *degree* of v and it is denoted *deg* v. If *deg* v is an even number, then v is said to be an *even* vertex; otherwise v is said to be *odd*. A vertex of degree 0 is said to be *isolated*. A graph (V, E) is said to be *finite* if the set of vertices, V, is a finite set. When we say that $G(V, E)$ is a graph we mean that G is a set with vertex set V and edge set E.

We can now compare the set notation of a graph with the graphical (pictorial) representation of a graph. We study the graph $G(V, E)$, where $V = \{a, b, c, d, e\}$ and $E = \{ab, ac, ad, bd\}$. The pictorial representation of this graph is given below:
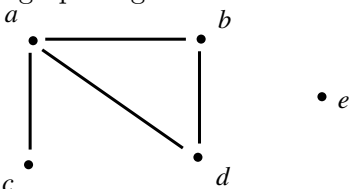
Here we see that the vertice e is an isolated vertex, that a is incident to b, c, d but that b is not incident to c. The degrees are given by *deg* $a = 3$, *deg* $b = 2$, *deg* $c = 1$, *deg* $d = 2$, and *deg* $e = 0$. Hence the set of

even vertices is $\{b, d, e\}$ and the set of odd vertices is $\{a, c\}$ and a number of similar statements can also be made.

So a graph is a pair of two sets. This means that a graph is a very general concept, as such the theory of graphs can span over many different applications in several different disciplines. As we saw above, graphs occur in many disciplines in computer science and software engineering.) Since graphs then can viewed as sets (or really a pair of two sets), we can introduce various set theoretic constructs with graphs. If we have an existing graph, G, we can talk about removing elements (vertices or edges) from G to form smaller graphs; we can talk about adding elements (vertices or edges) to G to form larger graphs.

So graphs are like sets, we cannot talk about the empty graph, which would just be the graph $(\emptyset, \emptyset)$, this would be too boring, but graphs nevertheless exhibit similarities to sets. One very important aspect of this is the notion of subgraph, which we now introduce.

Definition: Let $G(V, E)$ be a graph. A graph $G_1(V_1, E_1)$ is a *subgraph* of G if and only if $V_1 \subseteq V$ and $E_1 \subseteq E$. This is written $G_1 \subseteq G$. (Which is a form of abuse of notation.)

In connection with this it is also natural to introduce the complete graph:

Definition: Let V be a set of n vertices. The *complete graph on n vertices*, denoted K_n, is the graph (V, E), where V is the set of edges that contain every possible pair of vertices from V.

So if we have a set of n vertices, we can form all manner graphs by associating to V a set of edges. We can form $(V, \emptyset)$, which would be a pretty boring graph, just consisting of isolated nodes, but then if we gradually add edges we would have larger and larger graphs and eventually we would come to the complete graph on n vertices, K_n. We could illustrate this by studying a set of 4 vertices, as above, and gradually add varying subsets of edges. Each of the resulting graphs, G, would then have $(V, \emptyset) \subseteq G \subseteq K_4$. In the figure below we give pictorial representations of $(V, \emptyset)$, which we call A, to the left, and K_4, which we call E to the right, and then we have three graphs in between, B, C, D. We have drawn rectangles around the graphs too, but these are merely to separate the various graphs from each other and are not to be considered part of the graphs themselves:
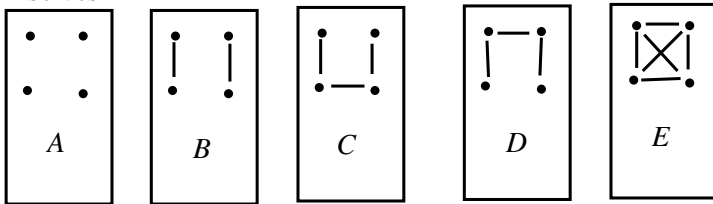
We then have the following subgraphrelations that hold: $A \subseteq B \subseteq C \subseteq E = K_4$ and $A \subseteq B \subseteq D \subseteq E = K_4$. Observe that we do *not* have $C \subseteq D$ or $D \subseteq C$ since the edgeset of C and the edgeset of D do not fulfill a subsetrelation.

By the definition we do not have to have the same number of vertices in a subgraph and a graph, one requirement is of course that the subgraph contains fewer vertices that that graph which it is subgraph of. We can illustrate this by giving pictorial representations of subgraphs of K_4, based on the immediately preceding situation, but we here replace the graphs A, B, C, D with other (similar-looking) graphs A', B', C', D':
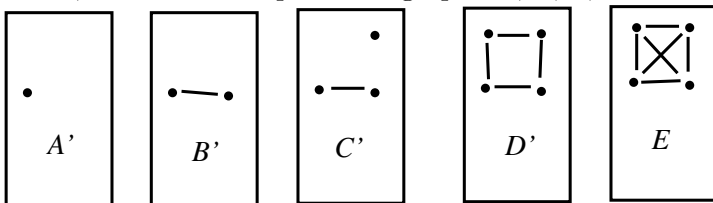
In this situation we do have each graph to the left being a subgraph of that to the right so that $A' \subseteq B' \subseteq C' \subseteq D' \subseteq E = K_4$. The set of graphs is indeed a partially ordered set under the subgraphrelation (prove this!) and a sequence of elements (a, b, c, d) in a partially ordered set that fulfill $a < b < c < d$ is called a *chain*. This means that the sequence $(A', B', C', D', E(= K_4))$ is a chain of graphs. A chain could of course have other than 4 or 5 elements, but we will not concern ourselves with chains further in this course.

The bipartite graph. A structure of particular interest in the study of database systems is the *bipartite graph.* This relationship arises for example if we wish to create a way to manage one type of entities that can relate to another type of entities and that elements each of theses types of entities cannot relate to each other. For example consider the situation where we wish to study the side-effects of medical drugs in order to create a database for use by medical personnel and researchers. Let us restrict ourselves to four drugs and three side effects. Let us also make both the drugs and the side effects fictitious so as to not encounter any liability

problems. Let us assume that the three drugs' names are a=*Miracle drug 1*, b=*Miracle drug 2*, c=*Miracle drug 3*, and d=*Miracle drug 4* and that the three side effects are x=*Loss of arms*, y=*Loss of legs* and z=*Bad breath*. Let us then assume that a has the side effects x, z, b has the side effects y, z and c has x, z and that d has all side effects but that they happen very rarely (at least according to the drug company). Then these relationships could be described in a graph with the following pictorial representation:
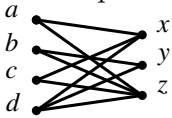

Note the crucial property of the bipartite graph: the vertices fall into two classes, here, it is the drugs that is one type of vertex and the side effects is the other class of vertices. The vertices appear in two distinct columns, which is the usual way to draw a bipartite graph. (We can of course also draw a pictorial representation of a bipartite graph in two rows above each other.) Let us make a definition before deepening this application:

Definition: A *bipartite graph* is a graph whose vertices can be partitioned into two sets, V_1 and V_2, called *bipartition sets*, in such a way that every edge joins a vertex in V_1 and a vertex in V_2. There are *no* edges of the type vw where both v, w are in V_1 or both are in V_2. A *complete bipartite graph* is a bipartite graph in which every vertex in V_1 is joined to every vertex in V_2. The complete bipartite graph on sets of m vertices and n vertices, respectively, is denoted $K_{m,n}$.

In the example above we had $V_1 = \{a, b, c, d\}$ and $V_2 = \{x, y, z\}$.

To conceive of an interesting application we also assume that the four medications has desirable effects for which they were designed. Let us assume that the medications has the following good effects: a has the effect of h=*Removing headache*, b has the effect of x'=*Regrowing lost arms*, c has the effect of y'=*Regrowing lost legs* and d has the effect of w=*Removing bad breath*. Then if we combine the previous graph with a new bipartite graph we can combine the information in both bipartite graphs like this:
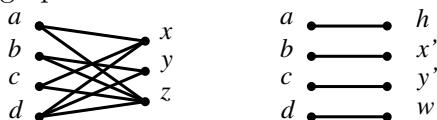
This information can be encoded in a database to help a doctor make a decision of what could be an appropriate course of action with a patient with a certain affliction. Let us say that we have a headache and wish to consult a doctor about it. The doctor opens the database and enters "headache" and then gets the following list of directions:

The patient is to take Miracle Drug 1, Miracle Drug 2, Miracle Drug 3, and Miracle Drug 4

The patient asks the doctor if there are any side effects of these drugs, and the doctor says, "No, the side effects are very rare". The patient asks which are the side effects and the doctor says, "Well, very rarely patients have lost their arms and legs and incurred bad breath from *Miracle Drug 4*". "Oh, that is terrible, can't we skip *Miracle Drug 4* then?", "No, the doctor replies, *MD4* will actually very likely *remove* bad breath, it is very rarely that it will *cause* bad breath (and loss of arms and legs)". The patient then enquires, "But why do I need to remove bad breath?", "Well, bad breath is a side effect of *MD3* and it is needed regrow your lost legs". "Lost legs! I have two legs!", "Yes, but you will lose your legs as a side effect of *MD2* but we still need to give you *MD2* to counter the side effects of *MD1* which is loss of arms, and we still need to give you *MD1* because it will remove your headache which is of course the thing you are here for isn't it? After all it is only very rarely that *Miracle drug 4* causes any problems."

Reassured by the doctor's knowledge and authority and trusting the statistics of the drug company that produced the miracle drugs, the patient accepts the treatment.

This example is a bit morbid, however it also illustrates how graphs can be used to model knowledge about the real world. It also shows that we might have to be a bit careful in trusting the wonders of technology: while the described technology may present a sound way to treat headache, there may also be so-called interactions between medicines, but this is another story.

We will prove a proposition, due to Euler, about graphs:

Proposition: Let $G = (V, E)$ be a graph. Then the sum of the degrees of all vertices is twice the number of edges. In mathematical symbols this can be expressed

$$\sum_{v \in V} deg\ v = 2|E|.$$

Proof: If we add all the degrees of all vertices, for a particular vertex, this number is the number of incident edges. This means that each edge incident on a aprticular vertex contributes 1 to this number, but every edge is incident to two vertices so when the sum is extended over all vertices, each edge contributes 2 to the total sum. When the sum is extended to all vertices, this is equivalent to extending it to all edges so that the value of the sum will be

$$\sum_{v \in V} deg\ v = \sum_{e \in E} 1 + 1 = \sum_{e \in E} 2 = 2 \sum_{e \in E} 1 = 2|E|$$

which is what we wanted to prove.

Corollary: The number of odd vertices in a graph $G(V, E)$ is even.

Proof: The relation $\sum_{v \in V} deg\ v = \sum_{v \in V, v\ odd} deg\ v + \sum_{v \in V, v\ even} deg\ v = 2|E|$ holds for G. The degree of an even vertex in G is even $\sum_{v \in V, v\ even} deg\ v$ must be an even number so that $\sum_{v \in V, v\ odd} deg\ v$ also must be even. But since each term in this sum is odd, the number of terms, which is the number of odd vertices of G, must be an even number. That is the desired conclusion.

As an application of the proposition we count the number of edges in the graph above with the vertices $\{a, b, c, d, x, y, z\}$, the proposition says that twice the number of edges is

$$\sum_{v \in V} deg\ v = deg\ a + deg\ b + deg\ c + deg\ d + deg\ x + deg\ y + deg\ z = 2 + 2 + 2 + 3 + 3 + 2 + 4 = 18$$

which gives that the total number of edges is $18/2 = 9$ and this can also be easily seen by visually counting the edges.

Degree sequence. Study this concept independently and use the proposition and corollary to draw conclusions about degree sequences.

Isomorphism. Study the two following graphs:
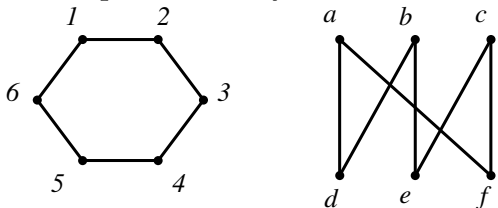
Are they different? Actually not. They are drawn differently, but if we form the sets of vertices and edges we see that the underlying sets will only differ in the names we give the elements. We will see this shortly, but first we will see how this can be understood in a more pictorial way. We can see that two pictorial representations of two graphs actually represent the same graph if and only if we can transform one into the other by imagining that one is made of rubber and then imagining stretching and bending one so that it fits onto the other, with each vertex in one graph fitting together with one vertex in the other graph and the same for all edges. To apply this procedure applied to the graphs above, we let the right one turn into rubber. The stretching and bending can then look like this:
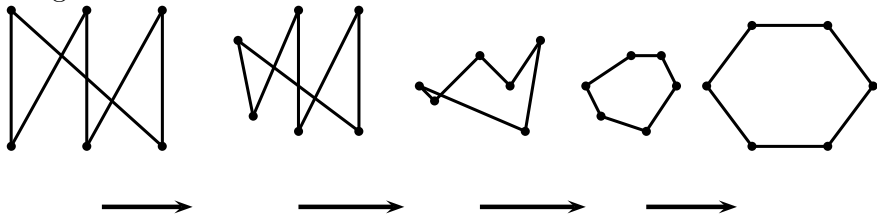
That this is possible means that the two graphs are essentially exactly the same graphs. The technical term for this is *isomorphism*, "iso" means "the same" and "morph" means "form". We cannot however have the talk about making one graph into rubber and stretching and bending as a definition, we will give the precise mathematical definitions using functions:

Definition: Given graphs $G_1(V_1, E_1), G_2(V_2, E_2)$, we say that G_1 is *isomorphic* to G_2 and write $G_1 \cong G_2$ if and only if there exists a one-to-one function φ from V_1 onto V_2 such that:

1. $vw \in E_1 \Rightarrow \varphi(v)\varphi(w) \in E_2$, and
2. $\forall ru \in E_2 : \exists vw \in E_1 : ru = \varphi(v)\varphi(w)$ (which just means that $r = \varphi(v)$ and $u = \varphi(w)$).

We call the function φ an *isomorphism* from G_1 to G_2, and, abusing notation, we say that $\varphi : G_1 \to G_2$ is an isomorphism.

The primary way of showing that two graphs are isomorphic is of course to exhibit an isomorphism, let us do this with the example above. After some pondering it turns out that the one-to-one function φ defined on the set $\{1, 2, 3, 4, 5, 6\}$ onto the set $\{a, b, c, d, e, f\}$ defined by $\varphi(1) = a$, $\varphi(2) = f$, $\varphi(3) = c$, $\varphi(4) = e$, $\varphi(5) = b$, and $\varphi(6) = d$ is an isomorphism between the given graphs.

An isomorphism between graphs is hence a one-to-one correspondence between the different vertex-sets of the two graphs in questions, but also fulfilling the two demands on the edges given by item 1 and 2 in the definition of isomorphism. These two demands means that the mapping $E_1 \ni vw \mapsto \varphi(v)\varphi(w) \in E_2$ is also a one-to-one correspondence between the edges of the two graphs.

Two isomorphic graphs are essentially the same graph, they have the same number of vertices, the same number of edges, and very importantly: the same *structure*, that is, the same number of vertices of every degree, and they are related in the same way in both graphs, the same number of loops and cycles, everything is exacly the same. There is no need to distinguish between two isomorphic graphs. Indeed, the relation $\cong$ is an equivalence relation on the set of all graphs and when we consider a certain graph and draw conclusions about that particular graph, the same conclusions hold for all graphs in the same equivalence class, that is all graphs that are isomorphics to the given graph we are studying. The isomorphism must preserve the properties of the graph, and we can see this is the example above. The degree of 1, a vertex in V_1, is 2. This vertex is mapped at the vertex a (because $a = \varphi(1)$) and we can see that the degree of this vertex is also 2. This goes for all vertices, they all happen to have the same degree, namely 2, but if another vertex say 2 would have had say degree 17, then the vertex that it maps to, that is $\varphi(2)$ must also have degree 17 for the graphs to be isomorphic.

To show that two graphs are isomorphic, we exhibit an isomorhpism. The primary way to show that two graphs are *not* isomorphic is to exhibit a property that both should have if they were isomorphic, but also show that one graph has this property and the other graph lacks this property. We will take a very simple example.

Example: Show that the two graphs given below are not isomorphic.
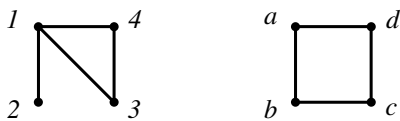
We can instantly see that there can never be any isomorphism between these two graphs. If we name the graph to the left G_1 and the one to the right G_2 we see that even though both graphs contain four vertices and fours edges, G_1 has a vertex labeled 2 which has degree 1. If an isomorphism φ existed, then the vertex $\varphi(2)$ would also have to have degree 1. But $\varphi(2)$ needs to be one of a, b, c, d and *each* of these vertices has degree 2. Hence the existence of in isomorphism is impossible, which is exactly the statement that the graphs are *not* isomorphic. We could apply the same reasoning to the vertex 1 or 3, each of which has the degree 3, and no vertex in G_2 has degree 3.

We will conclude this section by giving another example of a proof that two given graphs are not isomorphic. It will involve an appeal to a more detailed observation about the structure of the two graphs.

Example: Show that the two graphs given below are not isomorphic:

So how do we prove that these two graphs are not isomorphic? The degrees of the vertices of both graphs are $4, 1, 1, 1, 1, 1$ so they match ... what does not match? Well, if we assume that φ is an isomorphism, then since both graphs contain exactly one vertex of degree 4, the isomorphism must map the vertex of degree four onto the vertex of degree four. However, if we call the graph on the left G_1 and the graph on the right G_2, we see that the vertex that has degree four in G_1 has (of course) four adjacent vertices, the degrees of these vertices are $2, 1, 1, 1$. The degrees of the vertices adjacent to the vertex of degree four in G_2 however are $2, 2, 1, 1$ and this is a property that must be preserved by the isomorphism, and it evidently is not. For denote the four adjacent vertices in G_1 with a, b, c, d, then $\varphi(a), \varphi(b), \varphi(c), \varphi(d)$ must also have these degrees in some order, but, as we have seen, the degrees of $\varphi(a), \varphi(b), \varphi(c), \varphi(d)$ are $2, 2, 1, 1$, in some order. Hence these graphs are not isomorphic.