# MEETING 14 - MORE GRAPH THEORY

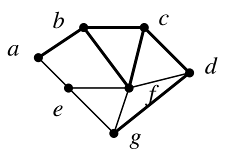
We continue with graph theory.

**Paths and circuits.** We will continue with graph theory and prepare the formulation of the second programming assignment.

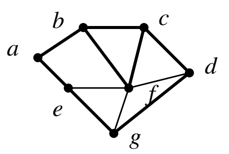
Graph theory abound with technical terms, so here comes another handful of them:

**Definition:** A *walk* in a graph is an alternating sequence of vertices and edges, beginning and ending with a vertex, in which each edge is incident with the edge immediately preceding it and the vertex immediately following it. The *length* of a walk is the number of edges in it. A walk is *closed* if the first vertex is the same as the last and otherwise it is called *open.* A *trail* is a walk in which all edges are distinct; a *path* is a walk in which all vertices are distinct. A closed trail is called a *circuit.* A circuit in which the first vertex appears exactly twice (at the beginning and at the end) and in which no other vertex appears more that once is called a *cycle.* An $n$-cycle is a cycle with $n$ vertices. It is *even* is $n$ is even and *odd* if $n$ is odd.

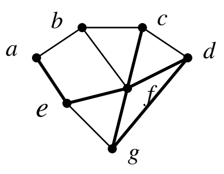
When encountering a definition like this, with no less than 11(!) new concepts it is absolutely necessary to draw a picture illustrating each concept. We will therefore draw eight graphs, the first four will illustrate a walk which is not a path or trail, then we have a closed walk which is not a path or trail, then we will have a trail which is not a path and lastly a path. Here are the first four figures:



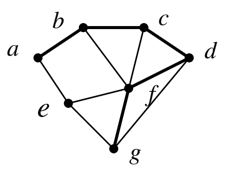
*abcfbcdg*



*abcfbcdgea*



*aefgdfc*

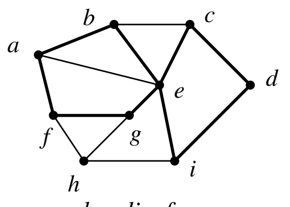


*abcdfg*

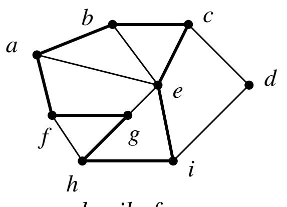
We choose to work in the same graph to illustrate more clearly the concepts. We denote a walk by naming the vertices it passes, but we do not name the edges that it passes, this is redundant since the two vertices that an edge has as endpoints specifies the edge. There are four walks illustrated above, the first one is an open walk and it is $abcfbcdg$, which is, as mentioned above, a walk that is neither a trail nor a path (since both edges and vertices repeat). The second walk is $abcfbcdga$ which is the same walk as before, but we have added content so that it becomes closed. Then comes $aefgdfc$, which is an open trail, it is a trail since the edges are distinct, they do not repeat, however, since the vertex $f$ repeats, it is not a path. Finally we give the path $abcdfg$ in which no vertices repeat, they are all distinct, that makes it a path. We can make a small observation here: in a path, no vertex appears twice. This means that also no edge can repeat, for if an edge repeats, then that edge is incident on a vertex that must also repeat, which means that it is not a path. We have therefore found that for all walks $W$ we have the implication $W is\ not\ a\ trail \Rightarrow W is\ not\ a\ path$, but this is the converse of $W is\ a\ path \Rightarrow W is\ a\ trail$, so all paths are trails. This means that if a walk is a path it is automatically a trail. It is harder for a walk to be a path than it is to be a trail. Make sure you study these concepts and understand each term well. Draw examples similar to those given above.

We will now produce examples of a circuit (closed trail) that is not a cycle and a cycle. It is not possible to give a "closed path" because a path must not contain repeated vertices. But a cycle will really be what we will be interested in. (In a sense a cycle is almost like a closed path; more precisely: a cycle would have been a path because it requires that vertices do not repeat, but only the beginning vertex, which coincide with the end vertex is the only one that repeats.) It will also become clear that the two concepts that we are really interested in are *cycle* and *path*, the earlier concepts, *walk*, *trail* and, to some extent, *circuit*, are really there mainly to help us develop clarity of the concepts *cycle* and *path.*

Here are the promised examples:



*abecdiegfa*



*abceihgfa*

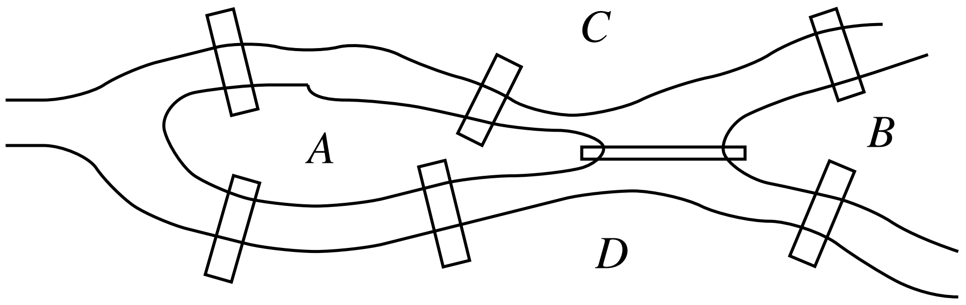
The first example gives a circuit $abecdiegfa$ that is not a cycle because in a cycle no repeated vertices are allowed. As the circuit passes through the vertex $e$, twice, this means that it cannot be a cycle. The other example $abceihgfa$ however does indeed illustrate a cycle precisely because no vertex (other than the starting vertex $a$) appears twice.

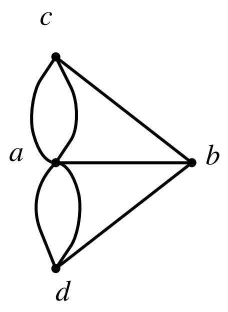
In these examples we have omitted to discuss the length or parity (whether an $n$-cycle is odd or even), but these concepts are very easy to grasp so we concentrate on the more important activity of making clear just what is a *trail*, a *path*, a *circuit*, and a *cycle*.

We will now start to introduce some more concepts (Hooray!)

**Defintion:** An *Eulerian* circuit in a graph is a circuit that contains every vertex and every edge. A graph is called *Eulerian* if it has an Eulerian circuit. Similarly an *Eulerian trail* is a trail between two vertices that contain every vertex and every edge. A graph is *connected* if there exists a walk between any two vertices.

*The Bridges of Königsberg.* The town of Köningsberg (contemporary Kaliningrad) had 7 bridges and the outline of the city is given to the left of the figure below:





To the right is a pictorial representation of a graph (a pseudograph, to be correct) that captures the relationship between the four land masses, $A, B, C, D$, these fours land masses are represented by the four vertices $a, b, c, d$ in the graph and the bridges are edges in the graph. It was a Sunday pleasure (of the wealthy I guess) to stroll around the city and try to find a way to walk across all bridges exactly once and return to the starting point. No one ever succeeded in doing this and a reident of this city, Leonhard Euler, realised that it is indeed impossible to do this. In terms of our recently defined concepts in graph theory, being able to do the Sunday walk just described would be equivalent to finding an Euler circuit in the graph to the right. Let us see why this is impossible.

It is a standard proof by contradiction: assume that there is an Euler circuit starting at one vertex and ending at the same vertex. Traversing a graph via an Euler circuit requires us to pass each edge exactly once. What would this mean for the degree of a vertex in the graph? If we complete the Euler circuit we must arrive at every vertex (land mass) using one edge (bridge) and then we must leave this vertex using another edge. We might visit the same vertex several but each time we arrive using one edge, we must leave the vertex using another edge. For this to be possible there must always be an even number of edges incident on every vertex. This means that every vertex in the graph must have even degree. What are the degrees of the vertices of the graph? Why, they are $5, 3, 3, 3$, they are *all* odd, not a single one of them is even. But the existence of an Eulerian circuit implies that they must *all* be even, this is a contradiction, hence there cannot be any Eulerian circuit and we have reproduced the argument that Euler himself probably used to destroy the Sunday amusement of the (probably more well-to-do) people in the city of Königsberg.

In fact this is a generally valid theorem:

**Theorem:** A graph is Eulerian (has an Eulerian circuit) if and only if it is connected and the degree of every vertex is even.

We omit the proof, it is not hard, but contains many details which you are welcome to study but we regard it as being outside of the course.

Finding an Eulerian circuit is certainly not only a matter of pleasure or entertainment, it is a very important procedure when planning large infrastructure maintenance, for example if the government of a country with a vast network of roads wishes to maintain those roads accessing them must be done in an economical way. If all the roads could be travelled and passing each road exactly once, then that would be the most desirable way. However, this may not be possible, because it is equivalent to finding an Eulerian circuit, such a circuit may not exist. Well, we can then disregard a well-chosen set of roads so that our network of roads in fact constitute a graph with each node having an even degree. When we have found an Eulerian circuit in this reduces network of roads, we can reintroduced the disregarded roads again and present a plan with which to access all the roads in a fairly economical way.

We will revisit this particular application when we introduce so-called *weighted graphs.*

To close this section we just state a variation of the above theorem concerning Eulerian *trails*, it can be studied when the Eulerian circuits have been somewhat mastered:

**Theorem:** A graph $G$ contains an Eulerian trail between two different vertices $u, v \Leftrightarrow G$ is connected and all vertices except $u, v$ are even.

**Proof:** The proof rests on the preceding theorem on Eulerian circuits and is not hard at all, but it is very long and but best studied when an understanding of Eulerian circuits is more solid.

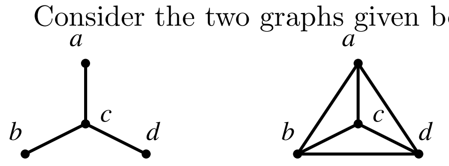
*Hamiltonian Cycles.* We will not study this in much detail, but we will focus on the following:

1. To know what a Hamiltonian Cycle is and to be able of finding one.
2. To understand and sometimes produce proofs of when a Hamiltonian cycle does not exist.
3. Gray Codes.

We will start by laying down precisely what it is:

**Definition:** A *Hamiltonian cycle* in a graph is a cycle that contains every vertex of the graph. A *Hamiltonian graph* is a graph that has a Hamiltonian cycle.

Consider the two graphs given below:



The one on the right is the complete graph on 4 vertices, as such it is definitely Hamiltonian. (Are all $K_n$ Hamiltonian? Why? Why not?) Let us call the graph on the left $G$. We wish to prove that it is not Hamiltonian, that is that it has no Hamiltonian cycle. This is a typical situation in which one resorts to proof by contradiction. We assume that there is a Hamiltonian cycle $H$ in $G$. The we consider the central vertex $c$, as this vertex is a part of the cycle and $a, b$ also must be part of $H$ (a Hamiltonian cycle must contain all vertices), then since the only edge that connects these vertices are $bc$ and $ca$, a segment of $H$ must be $acb$ or $bca$. Now $d$ needs to fit into the picture too, but since the only edge that connects $d$ to the other vertices is $cd$ and $c$ already is in the cycle we cannot use this edge, this would make the vertex $c$ appear twice in $H$ which is a contradiction. We conclude that there cannot be any Hamiltonian cycle in $G$, hence $G$ is not Hamiltonian.

In general, it is of course easier for a graph to be Hamiltonian the more edges there are, hence it would be a very adequate suspicion that $K_n$, that contains *all* possible vertices is indeed Hamiltonian. An earlier theorem enabled us to decide precisely which graphs are Eulerian (and which are not), but in general there is no easy method to decide whether there a graph is Hamiltonian or not. The easiest case is of course if the graph is Hamiltonian and we are able to actually find a Hamiltonian cycle as with $K_4$ above.

Hamiltonian graphs that has usage in the robust implementation of asynchronous automatons with digital circuitry, the state diagram is built on an $n$-dimensional cube and enables coding of adjacent states in binary sequences differing only in one bit. Another application is related to this and it is called *Gray Codes* and enables the creation of digital fault tolerant detection of spatial placements of machine elements that are digitally controlled. A Gray ocde of length $n$ is a list of all $2^n$ sequences of 0's and 1's with the property that that each sequence in the list differs from the next in precisely 1 digit (bit), and the last differs from the first in precisely one bit. For example $00, 01, 11, 10$ is a Gray code of length 2.

*The Adjacency Matrix.* Study this independently. It is also supported in the laborative work.

*Shortest Path Algorithms.* We will study the so-called Dijkstra's Algorithm. But first a concept:

**Definition:** A *weighted graph* is a graph $G(V, E)$ together witha function $w : E \to [0, \infty)$. If $e$ is an edge, the non-negative real number $w(e)$ is called the *weight* of $e$. The *weight of a subgraph of G* (for example a path or a trail) is the sum of the weights of the edges of the subgraph.

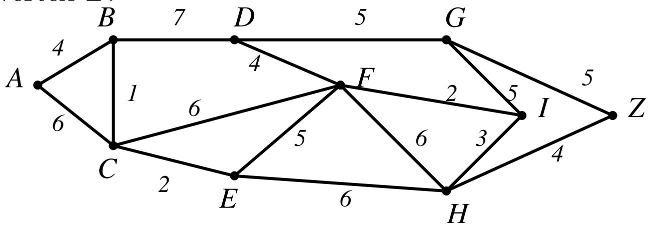
A classical problem is the *Travelling Salesperson's Problem* which attempts to find, in a Hamiltonian graph, a Hamiltonian cycle of least weight. This could representent finding a way to visit a number of cities (represented by vertices) which is the cheapest way for a merchant to visit all the cities he/she wants to.

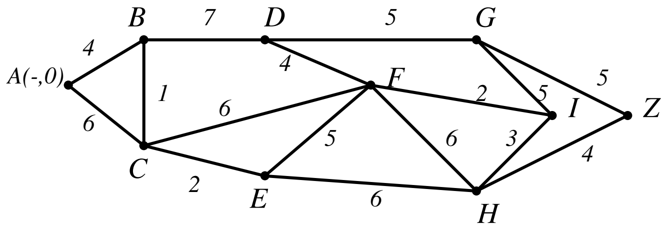
Another classical problem dealing with weighted graphs is to find a path between two edges that uses a set of edges that has a minimal weight. Instead of weight we could also talk about "cost" or "length", the weight could be thought of as representing the length of edges, and thus we are interested in finding the shortest path between two vertices.

The Dutch mathematician Edsger Dijkstra (1930-2002) found a way to do this and we will study his algorithm which is of course named *Dijkstra's Algorithm.*

We will give an analogy of the algorithm - the algorithm is about finding the shortest path between two given vertices, we can call one vertex the *starting vertex* and the other vertex (towards which we are headed) can be called the *finishing vertex.* In fact the algorithm proceeds in a way that will give the shortest path between any vertice in the graph and the starting vertex, so which vertex we choose as finishing vertex is not so important. The algorithm will proceed by labeling all the vertices in the graph and the label each vertex will receive will determine the shortest path (along with it's length) from the starting vertex to that vertex which has the label. The course of the algorithm can be thought of as water rising in a vessel whose sides are determined by the vertices and edges of the graph itself, with the starting vertex at the bottom, and water coming in through the starting vertex. Water has this curios ability to always find the shortest path between any two points. In Dijkstra's algorithm, the water can be thought of as a pool that grows and eventually engulfs all the vertices in the graph. We will illustrate this with an example.

**Example:** Consider the following weighted graph and find the shortest path between the vertex $A$ and the vertex $Z$:



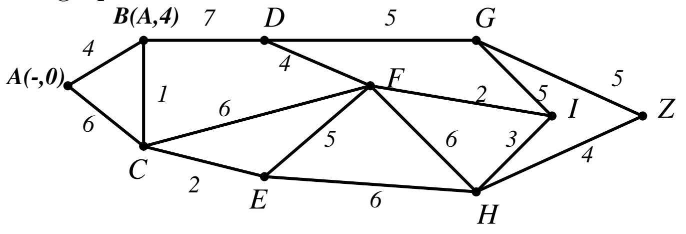


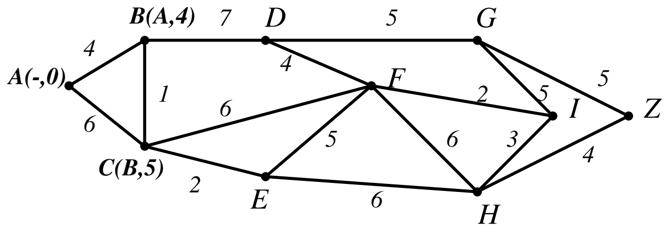
So when we imagine water starting to flow in at $A$, we label that vertex with $(-, 0)$, signifying that it is "at the bottom of the pool" and then we introduce the sets $U = \{A\}$ and $V = \{B, C\}$. The set $U$ will grow and when a vertex gets into $U$ it will get a label and the label will measure the distance to $A$ along a shortest path. Since $A$ is at distance 0 from itself, $A$'s label will have the measure 0. The set $V$ will vary and indicate those vertices that are next to any vertex in $U$, in the beginning $V$ consists of the vertices $B, C$ since they are next to $A$. The general steps of Dijkstra's algorithm are like this:

1. For every pair of elements $u \in U$ and $v \in V$ note the weight/distance of the edge $uv$.
2. Find the edge for which the distance/weight is the smallest and add the destination vertex $v$ to $U$ and give $v$ the label $(u, d)$ where $d$ is the distance/weight weight of the path from the starting vertex to $v$.
3. Repeat step 1 and 2 until all the vertices of the graph are contained in the set $U$. Then the labels will specifiy the shortest paths to each and every vertex in the graph.

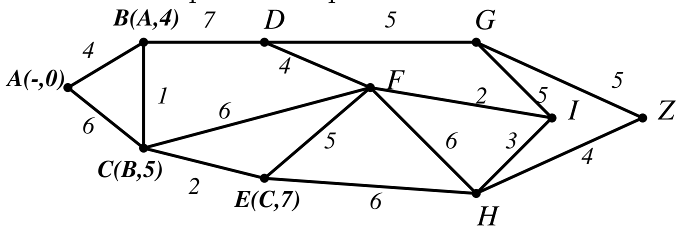
We will walk through these steps with the above graph. It is helpful to think of the set $U$ as a growing pool of water that step by step engulfs all the vertices of the graph.

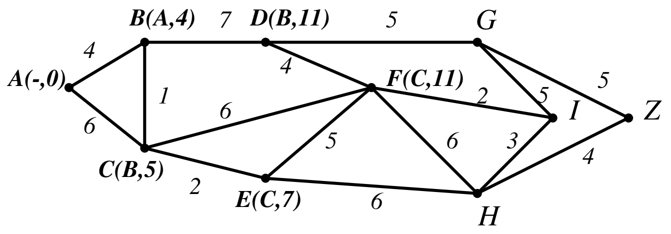
Now we have $U = \{A\}$, and $V = \{B, C\}$. Note all distances between all pairs $u, v$ where $u$ is chosen from $U$ and $v$ is chosen from $V$. There are only two combinations: $(u, v) = (A, B)$ giving the distance/weight $d = 4$, and $(u, v) = (A, C)$ givign the distance $d = 6$. Of these, 4 is smallest so we add $B$ to $U$ with the label $(A, 4)$. The graph with the new label is shown below on the left:



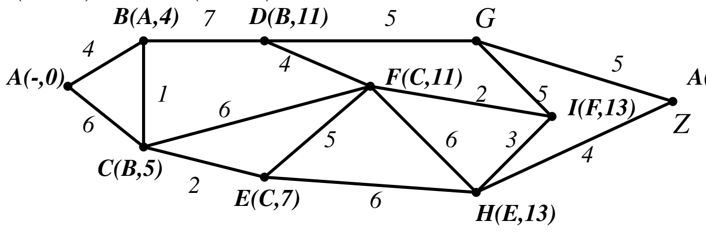


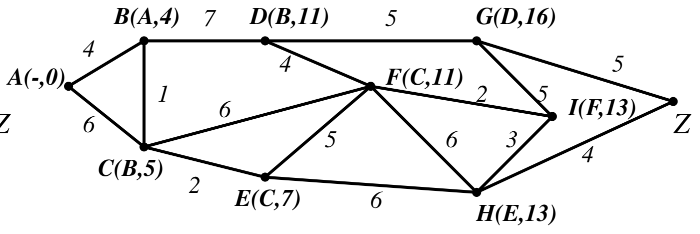
For the next step, we have $A = \{A, B\}$ and $V = \{C, D\}$, and the different choices of $(u, v)$ will then be $(u, v) = (A, C) \Rightarrow d = 6$, $(u, v) = (B, C) \Rightarrow d = 5$, and $(u, v) = (B, D) \Rightarrow d = 11$. Of these choices, the smallest distance/weight is $d = 5$ which corresponds to the choice of $(u, v) = (B, C)$ so $C$ gets the label $(B, 5)$ and a pictorial representation of this is shown above to the right. The water has now engulfed $A, B, C$ so that $U = \{A, B, C\}$ and $V = \{D, F, E\}$. Now the set of pairs $(u, v)$ will be $(B, D) \Rightarrow d = 11$, $(C, F) \Rightarrow d = 11$, and $(C, E) \Rightarrow d = 7$ of which $(C, E)$ gives the smallest $d = 7$, resulting in the addition of the label $(C, 7)$ to the vertex $E$. The pictorial representation of this situation is shown below to the left:



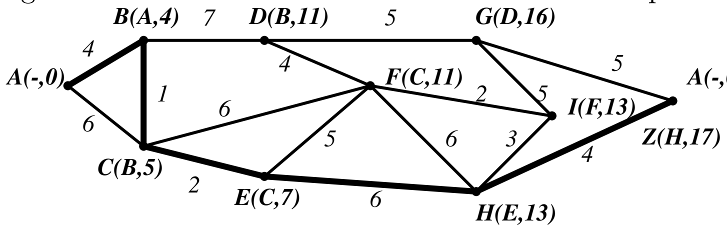


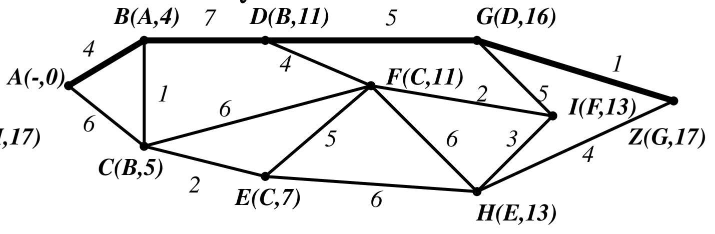
So now we have $U = \{A, B, C, E\}$ and $V = \{D, F, H\}$. Now the choices of $(u, v)$ are $(B, D) \Rightarrow d = 11$, $(C, F) \Rightarrow d = 11$, $(E, F) \Rightarrow d = 12$, and $(E, H) \Rightarrow d = 13$. We now come to an interesting situation. We have two candidates for the smallest value. If each of these candidates had common $u$'s or common $v$'s, we could add either of them, but now both the first and the second components differ, so we can actually add *both* $D$ and $F$ to $U$ with labels $(B, 11)$ and $(C, 11)$ respectively. That is illustrated above to the right. Again, consider the analogy of a rising water level. Now $U = \{A, B, C, D, E, F\}$ and $V = \{G, I, H\}$. The choices of $(u, v)$ now are $(D, G) \Rightarrow d = 16$, $(F, I) \Rightarrow d = 13$, $(F, H) \Rightarrow d = 17$, and $(E, H) \Rightarrow d = 13$ again giving two more labels $I(F, 13)$ and $H(E, 13)$. The situation is illustrated below on the left:



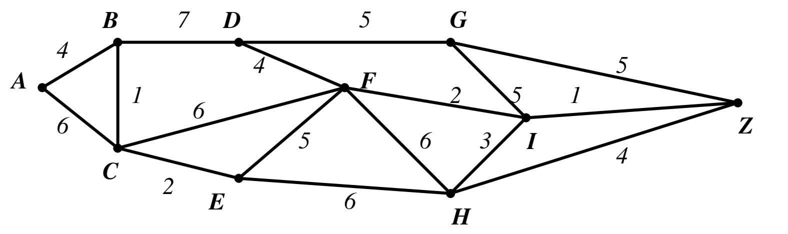


Now we are approaching the end, the water level has risen to engulf all the vertices $A, B, C, D, E, F, I, H$ and only $G, Z$ are outside. We can now reach $Z$ (the goal) which is expressed by $Z$ being a member of the set $V$. Indeed we have $U = \{A, B, C, D, E, F, I, H\}$ and $V = \{G, Z\}$. The $(u, v)$ pairs now are $(D, G) \Rightarrow d = 16$, $(I, G) \Rightarrow d = 18$ and $(H, Z) \Rightarrow d = 17$ giving the label $G(D, 16)$ which is drawn in the figure above to the right. Finally it is easy to find the last label for $Z$ which will be $(H, 17)$. Following the labels back to $A$ will then define a shortest path between $A$ and $Z$. And indeed, we could choose any vertex $Q$ in the whole graph and by following the labels back to $A$ it would indicate a shortest path between $A$ and $Q$. This is illustrated to the left below:





To the right we illustrate a slightly different situation, we have altered the weight of the edge $GZ$ to 1 so that a new shortest path appears. This is a shortest path also with the length 17 so it is a fully viable alternative to the one we already found. Another example giving an entirely different shortest path would be if we would have added the edge $IZ$ of weight 1, try Dijkstra's algorithm on this graph. It is showed below:



We can now state Dijkstra's Algorithm in full: To find a shortest path from vertex $A$ to vertex $Z$ in a weighted graph, carry out the following procedure:

1. Assign to the vertex $A$ the label $(-,0)$ and form the set $U = \{A\}$.
2. Until $Z$ is labeled of no further labels can be assigned, do the following:
    (a) Form the set $V$ of all vertices adjacent to all the vertices of the set $U$. For each pair of vertices, $(u, v)$, where $u \in U$ and $v \in V$, calculate the distances/weights from $A$ to $v$ based on the labels.
    (b) Find the minimum value of all these distances/weights.
    (c) Add the corresponding vertices (there may be more than one) to the set $U$ and label them to document the shortest paths so far to $A$.

If we cannot label vertices any more and we we have not yet reached the destination $Z$, then this means that the graph is not connected and there is no path at all between $A$ and $Z$, specifically then there cannot of course be any shortest path between $A$ and $Z$.