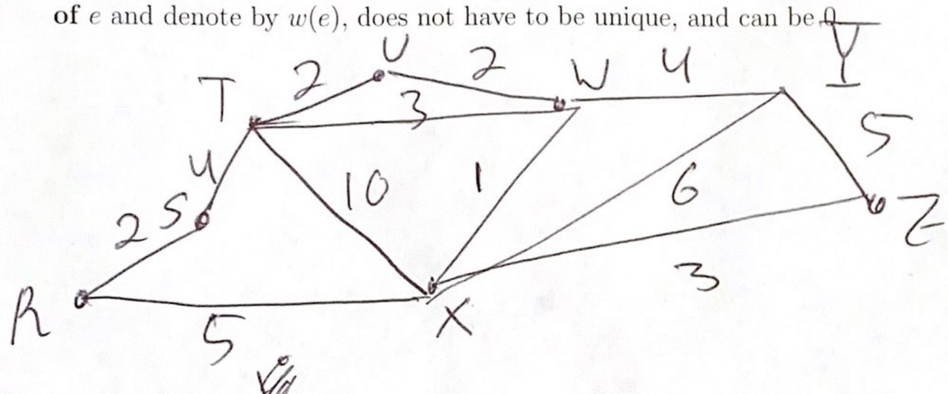
Chapter 12

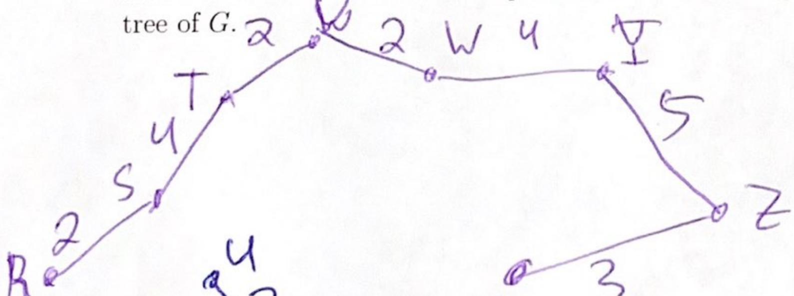
1 Section 12.1: Graph Algorithms

Definition 1.1. A **weighted graph** is a graph G in which every edge has been assigned a nonnegative integer. The number assigned to a particular edge e, which we call the **weight of** e and denote by $w(e)$, does not have to be unique, and can be 0.



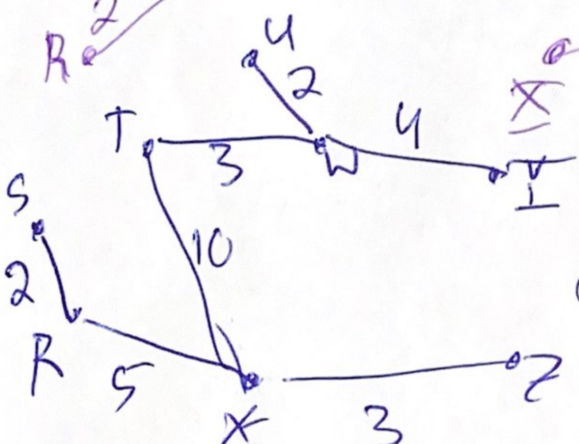
- Recall that a **spanning tree** T of a graph G is a subgraph of G that is both a **tree** (connected and no cycles) and a spanning subgraph of G (every vertex of G is included in T).

Problem 1.2. Given a weighted graph G, can we find a minimum weight spanning tree T of G? By minimum weight, we mean that if you summed up the weights of the edges of T, the sum would be smaller than or equal the sum of the weights of edges of any other spanning tree of G.



weight of this spanning tree is

$2+4+2+2+4+5+3=21$



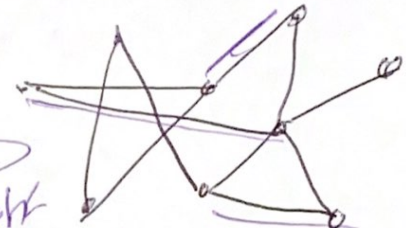
$10+2+5+2+4+3+3=29$ weight of this spanning tree is

~~Find the~~

Can we beat 21? Is there a tree with a smaller weight.

Remark: Every tree T on n vertices has n-1 edges.

Kruskal's Algorithm:



- Two Algorithms for This:

grab the edge with the lowest weight and add it to tree-under-construction

Algorithm 1.3 (Kruskal's Algorithm). Construct the minimum weight spanning tree T as follows:

(I) Set T to be just the vertices of G.

(II) Sort the edges of G into an ordered list $e_1, e_2, \ldots, e_m$ such that $w(e_k) \leq w(e_{k+1})$ for $k = 1, 2, \ldots, m-1$.

(III) Set $i = 0$.

(IV) Repeat the following until T is a tree:

  (i) Find the smallest integer j such that $i < j$ and adding edge e_j to T will not create a cycle.

  (ii) Add e_j to T.

  (iii) Set $i = j$.

Algorithm 1.4 (Prim's Algorithm). Construct the minimum weight spanning tree T as follows:

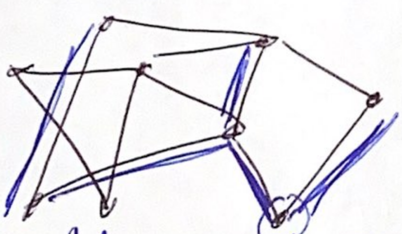
(I) Choose a starting vertex r, and set T to be just the vertex r with no edges.

(II) Repeat the following until T contains every vertex of T:

  (i) List all the edges of G that have as one endpoint a vertex in T, and the other endpoint a vertex that is not in T.

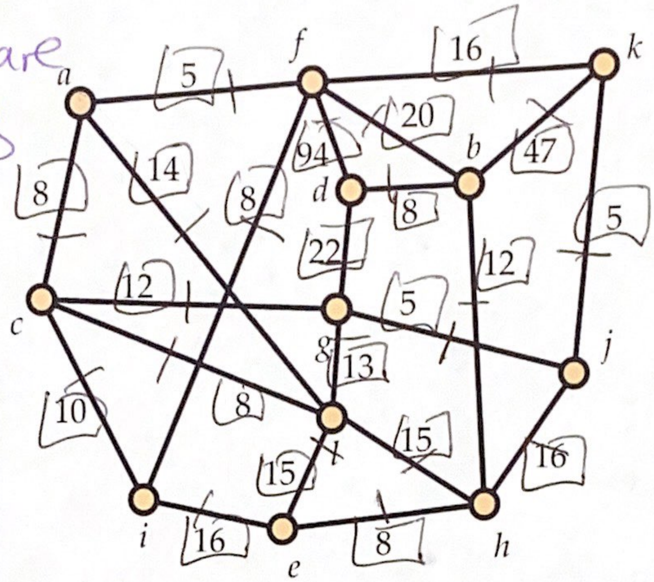
  (ii) From that list, choose the edge e with the smallest weight. Add v to to T, where v is the endpoint of e that is not in T, and add e to T.

Prim's Algorithm



build up a tree by adding edges that touch the tree

~~Rema.~~
Remark!
Minimum weight spanning trees are not necessarily unique



this graph has 12 vertices, so a spanning tree of it will have ~~eleven~~ 11 edges

~~list~~ in the event that two edges have the same weight, list in any order

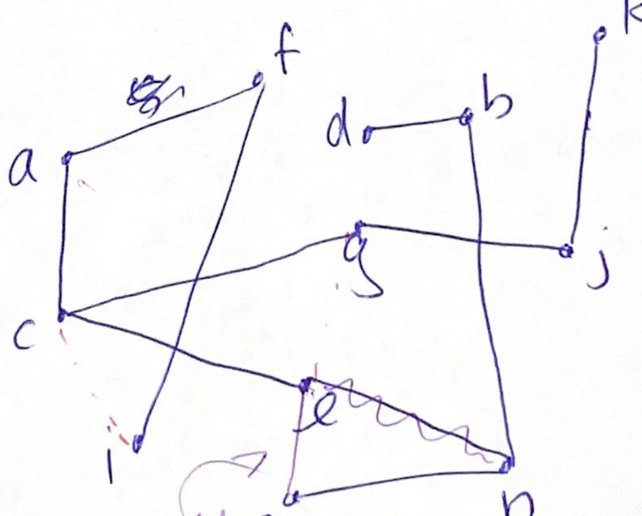
Exercise 1.5. Find a minimum weight spanning tree T of the graph shown in the figure (Figure 12.20 from the textbook) using Kruskal's Algorithm.

First, list all ~~the~~ edges and order them from smallest to largest weight

af: 5 ✓
gj: 5 ✓
kj: 5 ✓
fi: 8 ✓
cl: 8 ✓
~~af~~ el: 8 ✓
eh: 8 ✓
db: 8 ✓
ci: 10 ✗ creates a cycle, so skip over
cg: 12 ✓
bh: 12 ✓
gl: 13 ✗
al: 14 ✗
eh: 15 ✓ ← added
le: 15

ie: 16
hj: 16
fk: 16
fb: 20
dg: 22
bk: 47
fd: 94

go down the list, check to see that adding the edge won't create a cycle; add it to tree-in-construction if it doesn't

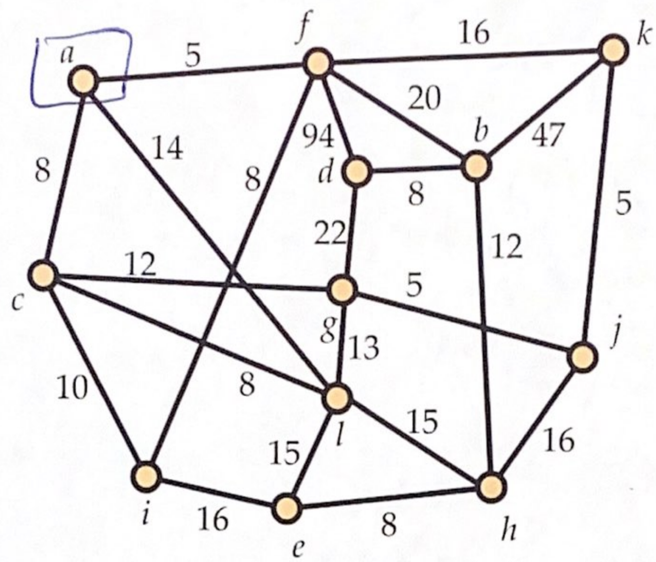


would have this edge instead of le

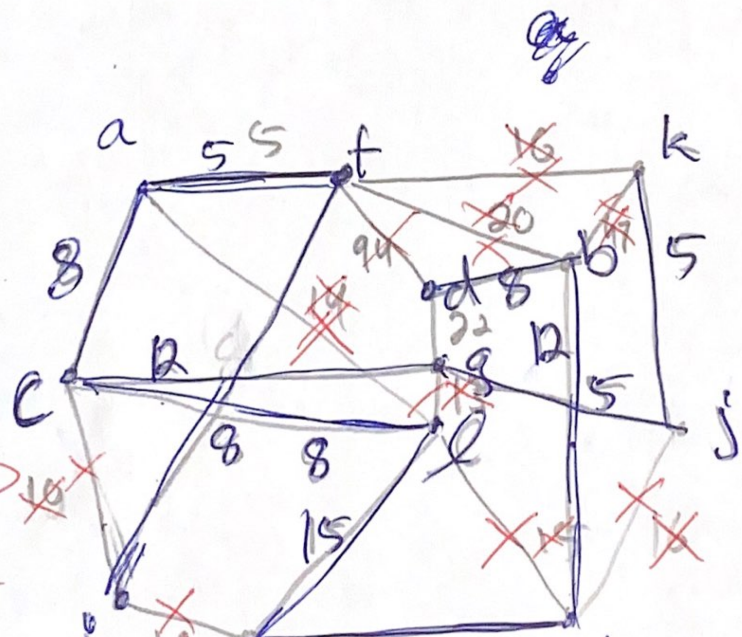
if I had listed le before lh

11 edges, so we can stop here

weight of this ~~min. span~~ min. weight spanning tree is
$5 \cdot 3 + 8 \cdot 5 + 12 \cdot 2 + 15 = \boxed{94}$



Exercise 1.6. Find a minimum weight spanning tree T of the graph shown in the figure (Figure 12.20 from the textbook) using Prim's Algorithm.



after adding fi, we see that c and i are both in the tree, so we can forget ci

Record of Order we added edges!

af
ac
fi
cl
cg
gj
kj
le
eh
bh
bd

added 11 edges, we are done

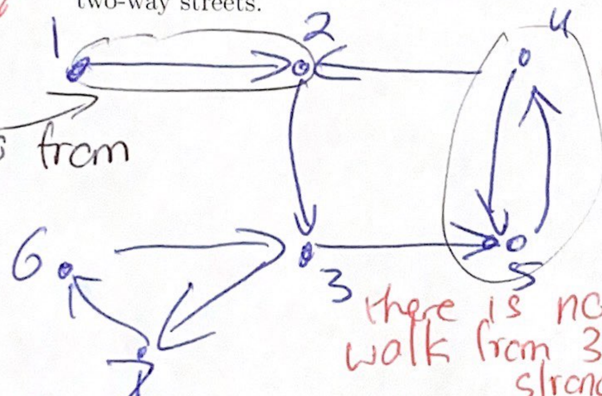
2 Section 12.2: Directed Graphs

for directed graphs:
strongly connected: any ~~two~~ pair of vertices a and b, there is a walk from a to b
weakly connected: If we forget about directions and pretend it's a regular graph, it's connected

Definition 2.1.
- A directed graph, also called a **digraph**, D is exactly like a graph G except that instead of edges, D has **directed edges**.
- A **directed edge** is like an edge, in that it touches two vertices u and v, but there is now an orientation from one vertex to the other. So formally, a directed edge is an ordered pair (u, v) of vertices.

edge
directed edge

Remark 2.2.
- In a graph G, there can be at most one edge between two vertices u and v, the edge $\{u, v\}$. In a directed graph D, there can be at most two directed edges between vertices u and v, (u, v) and (v, u).
- A digraph D can have walks just like a graph G, but with the restriction that when moving from a vertex u to a vertex v, the walk has to move along an edge whose direction is from u to v. Think of directed edges as one-way streets, and edges and two-way streets.



can go from 1 to 2, but not from 2 to 1

we can have two directed edges between the same pair of vertices, if they are going in different directions

there is no walk from 3 to 1, so this graph is not strongly connected, but it is weakly connected

Definition 2.3. The **directed walk**, **directed path**, **directed cycle**, **directed circuit**, and **directed trail** are exactly the same as there non-directed counterparts with the condition that when moving from vertex to vertex, you have to move in the direction specified for the directed edge.

directed walk: A sequence of vertices $v_1, v_2, v_3, \ldots, v_n$ such that for each v_i, v_{i+1} there is ~~an edg~~ a directed edge from v_i to v_{i+1}

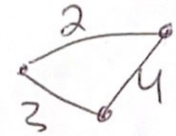
In the graph above: $3 \to 7 \to 6 \to 3 \to 5 \to 4 \to 5$ is a directed walk

$3 \overset{\times}{\to} 2 \overset{\times}{\to} 1 \to 2 \overset{\times}{\to} 4$
that's not a directed walk

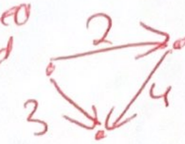
in all these cases "driving the wrong way down a 1-way street"

Ex: looks like there is a cycle here, but there isn't

weighted graph



weighted directed graph



3 Section 12.3: Dijkstra's Algorithm for Shortest Paths

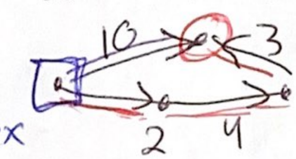
We are now working with a directed graph D in which every directed edge has been assigned a nonnegative number called a **length**.

Problem 3.1. Given a directed graph D with assigned edge lengths and a specific vertex r, for every $x \neq r$, find

- The shortest directed path from r to x,
- The **distance** from r to x, which is the sum of the lengths of the edges in the shortest path

- Intuition behind Dijkstra's Algorithm:
  - We are going to maintain a list of the paths we currently have from r to each vertex x and and what our current distance is.
  - If we don't have a path yet, we say the distance is ∞.
  - At each step, we choose a vertex y, and see if the path from r to y and then going along the edge from y to x is shorter than our current path from r to x.
  - If this new path involving y is shorter, we replace the old path and distance with this new one.

we have a root vertex



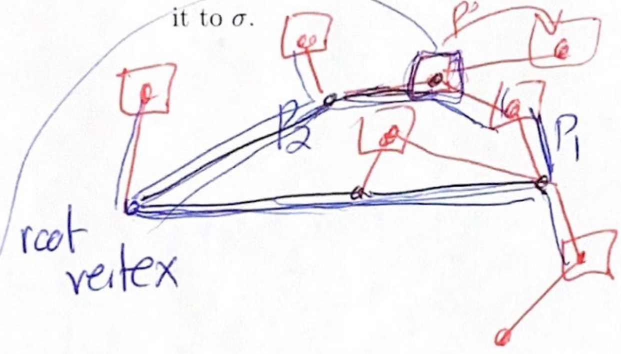
what is the shortest path, and the length of the shortest path, from root vertex to any other vertex

Here, by length of path, we mean the sum of the weights of the edges in that path.

So, in the above, red path has length 2+4+3=9
purple path has length 10, so red path is shorter

- Description of Update Step from Dijkstra's Algorithm:

  - We will have a sequence of vertices $\sigma = (r = v_1, v_2, \ldots, v_i)$. Vertices in this list are called the **permanent vertices**. Vertices not in this list are called **temporary vertices**.
    - The permanent vertices are the vertices we have figured out. We know a shortest path from r to each permanent vertex, and we know the distance from r to that permanent vertex.
    - The sequence σ is not generally a path, it's an ordering of the vertices in how close they are to r.
  - For each vertex x, we will have a path $P(x)$ from r to x, and $\delta(x)$ the length of that path.
  - Go to v_i, the last vertex in σ. For each temporary vertex x, see if the path $P(v_i)$, and then the edge v_i to x, is shorter than what we currently have for $P(x)$.
    - If it's not shorter or there is no edge from v_i to x, do nothing.
    - If $P(v_i) + v_ix$ is shorter than $P(x)$, $P(v_i) + v_ix$ is the new $P(x)$ and $\delta(x)$ is the length of the new $P(x)$.
  - Finally, choose the temporary vertex x whose distance from r $\delta(x)$ is the smallest among temporary vertices. This is our newest permanent vertex v_{i+1}, and we append it to σ.



permanent vertices: vertices that we have "completed"; we know the distance from root to those vertices, and we have a shortest path

temporary vertices: everybody else, a.k.a. the ones we have not figured out the shortest path yet

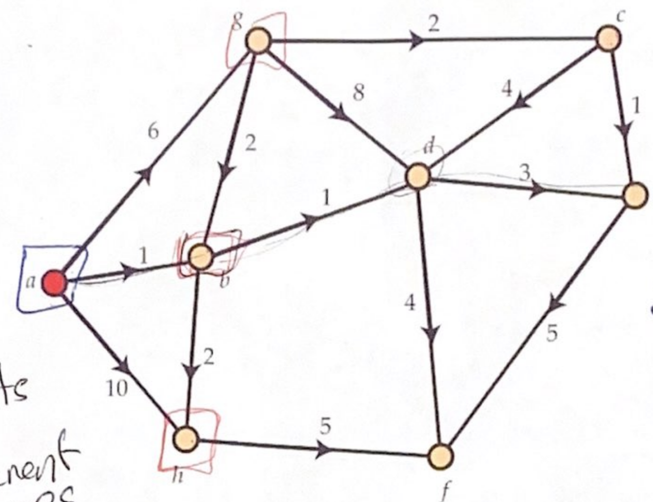
- we have for the temporary vertices, a current distance from the root vertex, which is the path going along permanent vertices to that temporary vertex

- the temporary vertices with a $\square$ have such a path

Algorithm works by alternating these two steps!

1) ~~look at the~~ find the $\square$ vertex with the shortest path to it, and make it permanent

2) with this new permanent vertex, we update the shortest paths to our temporary vertices



When do we finish?
when either
1) All vertices are permanent, or
2) All temporary vertices have $\delta = \infty$

σ represents list of permanent vertices

Exercise 3.2 (Exercise 12.11 from the textbook). Use Dijkstra's Algorithm to find the distance from a to each other vertex in the given directed graph, and a directed path of that length.

1) $\sigma = \{a\}$

$P(a) = (a)$ $\quad \delta(a) = 0$
$P(b) = (a,b)$ $\quad \delta(b) = 1$
$P(c) = \sim$ $\quad \delta(c) = \infty$
$P(d) = \sim$ $\quad \delta(d) = \infty$
$P(e) = \sim$ $\quad \delta(e) = \infty$
$P(f) = \sim$ $\quad \delta(f) = \infty$
$P(g) = (a,g)$ $\quad \delta(g) = 6$
$P(h) = (a,h)$ $\quad \delta(h) = 10$

choose the temp. vertex with the shortest path, which is b

2) $\sigma = \{a,b\}$

$P(a) = (a)$ $\quad \delta(a) = 0$
$P(b) = (a,b)$ $\quad \delta(b) = 1$
$P(c) = \sim$ $\quad \delta(c) = \infty$
$P(d) = (a,b,d)$ $\quad \delta(d) = 2$ update
$P(e) = \sim$ $\quad \delta(e) = \infty$
$P(f) = \sim$ $\quad \delta(f) = \infty$
$P(g) = (a,g)$ $\quad \delta(g) = 6$
$P(h) = (a,b,h)$ $\quad \delta(h) = 3$ update

make d permanent

3) $\sigma = \{a,b,d\}$

$P(a) = (a)$ $\quad \delta(a) = 0$
$P(b) = (a,b)$ $\quad \delta(b) = 1$
$P(c) = \sim$ $\quad \delta(c) = \infty$
$P(d) = (a,b,d)$ $\quad \delta(d) = 2$
$P(e) = (a,b,d,e)$ $\quad \delta(e) = 5$ update
$P(f) = (a,b,d,f)$ $\quad \delta(f) = 6$ update
$P(g) = (a,g)$ $\quad \delta(g) = 6$
$P(h) = (a,b,h)$ $\quad \delta(h) = 3$ $\leftarrow$ make h permanent

4) $\sigma = \{a,b,d,h\}$ no updates

$P(a) = (a)$ $\quad \delta(a) = 0$
$P(b) = (a,b)$ $\quad \delta(b) = 1$
$P(c) = \sim$ $\quad \delta(c) = \infty$
$P(d) = (a,b,d)$ $\quad \delta(d) = 2$
$P(e) = (a,b,d,e)$ $\quad \delta(e) = 5$ make e permanent
$P(f) = (a,b,d,f)$ $\quad \delta(f) = 6$
$P(g) = (a,g)$ $\quad \delta(g) = 6$
$P(h) = (a,b,h)$ $\quad \delta(h) = 3$

5) $\sigma = \{a, b, d, h, e\}$ no updates

$P(a) = (a)$ $\delta(a) = 0$

$P(b) = (a, b)$ $\delta(b) = 1$

$P(c) = \sim$ $\delta(c) = \infty$

$P(d) = (a, b, d)$ $\delta(d) = 2$

$P(e) = (a, b, d, e)$ $\delta(e) = 5$

$\boxed{P(f) = (a, b, d, f) \quad \delta(f) = 6}$ make f permanent (could have chosen g if we wanted to)

$P(g) = (a, g)$ $\delta(g) = 6$

$P(h) = (a, b, h)$ $\delta(h) = 3$

6) $\sigma = \{a, b, d, h, e, f\}$ no updates

$P(a) = (a)$ $\delta(a) = 0$

$P(b) = (a, b)$ $\delta(b) = 1$

$P(c) = \sim$ $\delta(c) = \infty$

$P(d) = (a, b, d)$ $\delta(d) = 2$

$P(e) = (a, b, d, e)$ $\delta(e) = 5$

$P(f) = (a, b, d, f)$ $\delta(f) = 6$

~~$P(g) = (a, g)$ $\delta(g) = 6$~~

~~$P(h) = (a,$~~

$P(g) = (a, g)$ $\delta(g) = 6$ make g permanent

$P(h) = (a, b, h)$ $\delta(h) = 3$

7) $\sigma = \{a, b, d, h, e, f, g\}$

$P(a) = (a)$ $\delta(a) = 0$

$P(b) = (a, b)$ $\delta(b) = 1$

~~$P(c) = (a$~~

$P(c) = (a, g, c)$ $\delta(c) = 8$ update

$P(d) = (a, b, d)$ $\delta(d) = 2$

$P(e) = (a, b, d, e)$ $\delta(e) = 5$

$P(f) = (a, b, d, f)$ $\delta(f) = 6$

$P(g) = (a, g)$ $\delta(g) = 6$

$P(h) = (a, b, h)$ $\delta(h) = 3$

make c permanent

8) complete, we now have a list of the shortest path from a to each vertex and the length of such a path.

$\sigma = \{a, b, d, h, e, f, g, c\}$

$P(a) = (a)$ $\delta(a) = 0$

$P(b) = (a, b)$ $\delta(a) = 1$

$P(c) = (a, g, c)$ $\delta(c) = 8$

$P(d) = (a, b, d)$ $\delta(d) = 2$

$P(e) = (a, b, d, e)$ $\delta(e) = 5$

$P(f) = (a, b, d, f)$ $\delta(f) = 6$

$P(g) = (a, g)$ $\delta(g) = 6$

$P(h) = (a, b, h)$ $\delta(h) = 3$