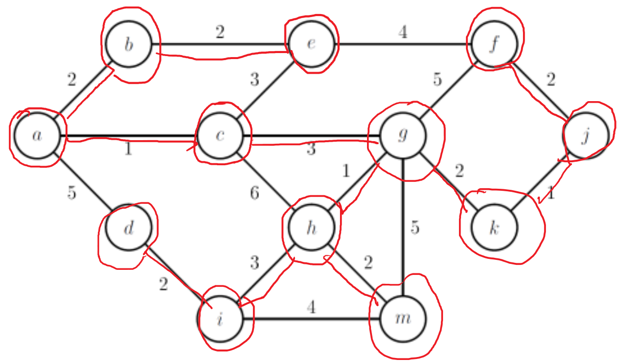
CSCB63 TUT2 WEEK11
andrewyk.leung@mail.utoronto.ca
https://snoopysnipe.github.io/ta/b63f20/

1) Prim's Algorithm

2) Adjacency Matrix Uses

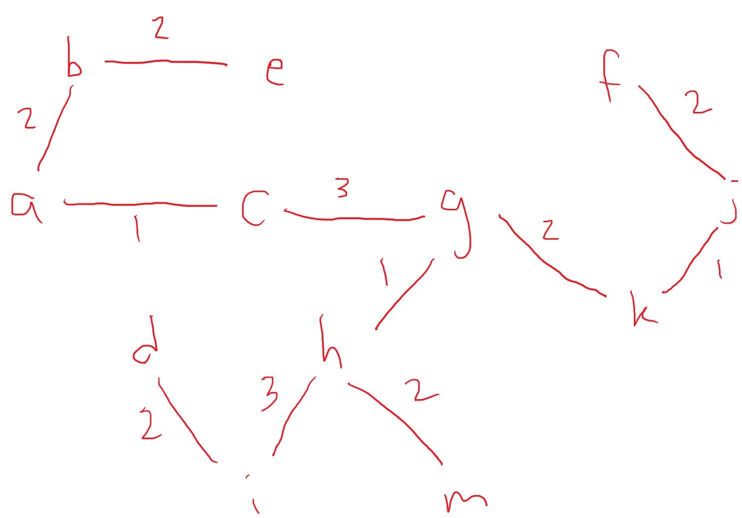
1

G	adjacency list
a	$(b,2)$, $(c,1)$, $(d,5)$
b	$(a,2)$, $(e,2)$
c	$(a,1)$, $(e,3)$, $(g,3)$, $(h,6)$
d	$(a,5)$, $(i,2)$
e	$(b,2)$, $(c,3)$, $(f,4)$
f	$(e,4)$, $(g,5)$, $(j,2)$
g	$(c,3)$, $(f,5)$, $(h,1)$, $(k,2)$, $(m,5)$
h	$(c,6)$, $(g,1)$, $(i,3)$, $(m,2)$
i	$(d,2)$, $(h,3)$, $(m,4)$
j	$(f,2)$, $(k,1)$
k	$(g,2)$, $(j,1)$
m	$(g,5)$, $(h,2)$, $(i,4)$



min-priority queue

~~(a, c, 1)~~
~~(a, b, 2)~~
~~(b, e, 2)~~
~~(c, g, 3)~~
~~(g, h, 1)~~
~~(g, k, 2)~~
~~(k, j, 1)~~
~~(h, m, 2)~~
~~(j, f, 2)~~
~~(h, i, 3)~~
~~(i, d, 2)~~



Discussion: Why use adjacency matrices instead of adjacency lists

Given how efficient an adjacency list representation can be for the algorithms we've seen so far, it may not be obvious how the matrix representation of a graph is useful. But it turns out that the matrix representation can let us do a lot of interesting things. In assignment 2 we're looking at using the matrix representation of a graph to determine whether it has a universal source, but we'll look at another aspect of this representation here.

Once we write a graph as a matrix we can start using linear algebra on it – it turns out that many interesting properties of a graph are encoded in its eigenvalues and eigenvectors, for example. We won't go into that here, but we'll look at what happens when we start doing multiplication on this matrix.

ex: Suppose that A is the adjacency matrix of an undirected graph. Show that the i^{th} entry on the diagonal of A^3 is twice the number of triangles (length-three cycles) in G that pass through vertex v_i.

$$A_{uv} = \begin{cases} 1 & G \text{ has an edge from } u \text{ to } v \\ 0 & G \text{ does not have an edge from } u \text{ to } v \end{cases}$$

$$= \# \text{ of one-edge paths from } u \text{ to } v$$

If you take A^2, the multiplication formula for the value at A_{uv} is $A_u \cdot A_v$. This is just the sum of vertices w where there is an edge from u to w and w to v. So the entries A^2_{uv} is the number of two-edge paths from u to v. (You can read the vertex degrees off of the diagonal of A^2).

You can proceed inductively to show that the entry A^n_{uv} of A^n is the number of n-edge paths from u to v. Note that these paths are not necessarily simple.

This means that an entry A^3_{uu} of A^3 gives the number of 3-edge paths from u to itself. Since we don't allow self-loops in a graph G, all these paths must contain 3 vertices and must be simple (so they represent triangles). Since the graph is undirected, each triangle is counted twice.