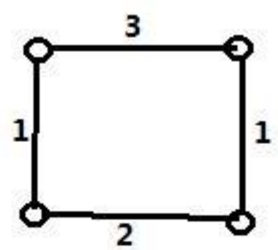
CPT_S 580 HW4

Yang Zhang

11529139 (graduate)

4.3



4.4.1

Vertex	Df#	low#
a	0	0
b	2	2
c	1	0
d	3	0
e	4	0
f	5	0
g	6	3
h	7	6
i	8	6

Vertex g and c are cut vertices, and dfnumber(c) <= low(b), dfnumber(g) <= low(h), which verifies the assertion of Corollary 4.4.12

4.4.2

Vertex	Df#	low#
a	2	1
b	0	0
c	1	1
d	4	1
e	3	1
f	6	2
g	5	2
h	7	5

i	8	5

Vertex g and c are cut vertices, and dfnumber(c) <= low(b), dfnumber(g) <= low(h), which verifies the assertion of Corollary 4.4.12

4.4.3

Vertex	Df#	low#
a	1	0
b	8	8
c	0	0
d	3	0
e	2	0
f	5	1
g	4	3
h	6	4
i	7	4

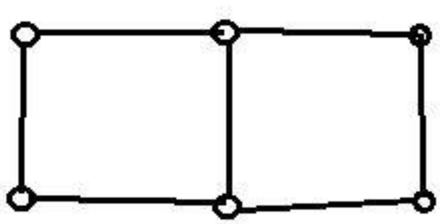
4.4.4

Vertex g is cut vertex, and dfnumber(g) <= low(h), which verifies the assertion of Corollary 4.4.12

Vertex	Df#	low#
a	1	0
b	3	3
c	2	0
d	4	0
e	0	0
f	6	0
g	5	3
h	6	6
i	7	6

Vertex g and c are cut vertices, and dfnumber(c) <= low(b), dfnumber(g) <= low(h), which verifies the assertion of Corollary 4.4.12

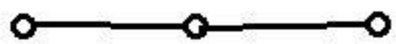
5.1.2



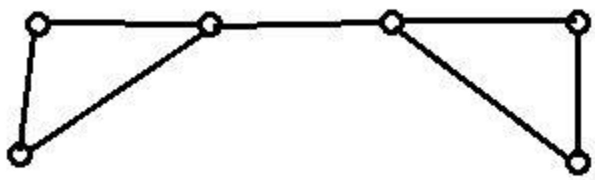
5.1.4

It is impossible, because if the 3 connected graph G only has 1 bridge, then $k_E(G) = 1$ and $k_V(G) = 3$, which violates the Corollary 5.1.6 ($k_V(G) <= k_E(G)$).

5.1.14

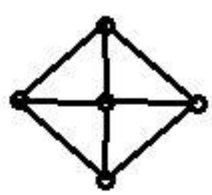


5.1.15



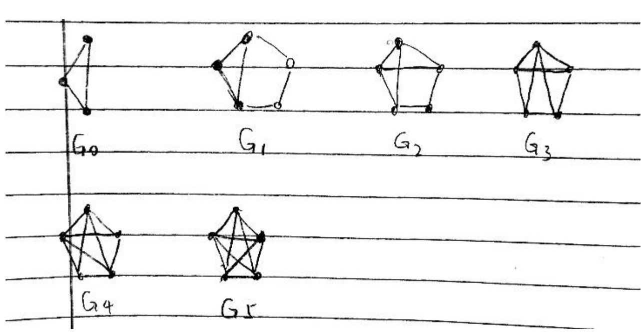
5.1.20

Any 3-connected simple graph must have at least 5 vertices and min degree number >= 3. Therefore, the smallest possible 3-connected graph is as shown below:

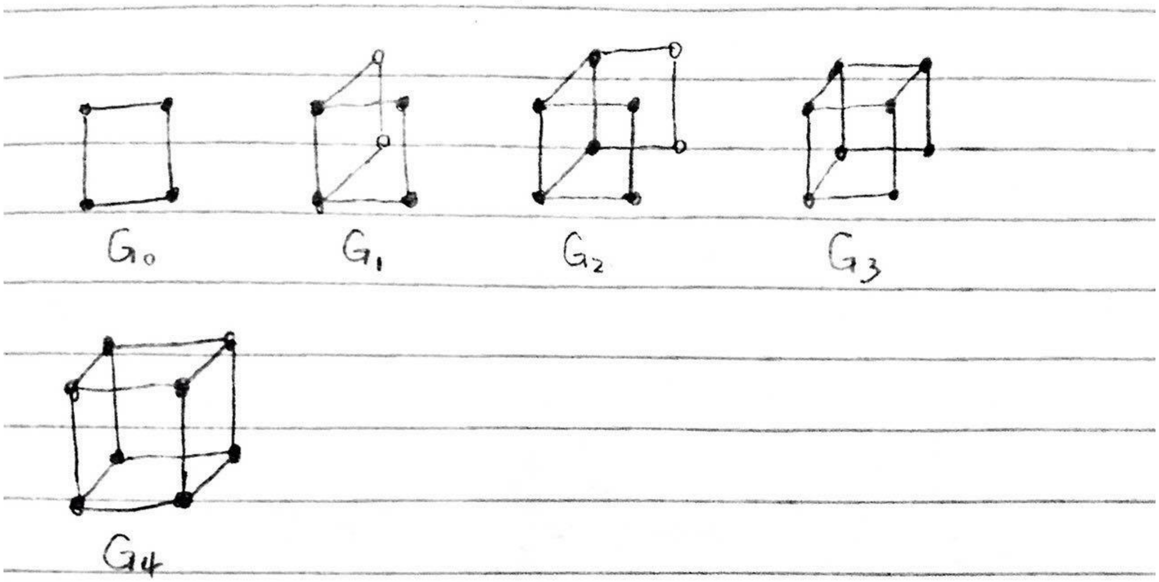


Which has 8 egdes. Therefore, there doesn't exist such a graph with 7 edges.

5.2.2



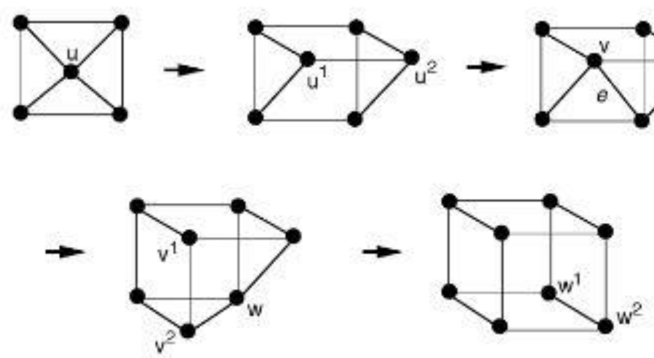
5.2.3



5.2.6

A 3-connected graph can be constructed from a wheel graph by applying Tutte Synthesis

Q_3 can be constructed from a wheel graph as follow steps:



Therefore, Q_3 is 3-connected graph.

5.3.2

One collection of two internally disjoint paths could be P={(u,t,y,v), (u,s,w,x,v), (u,a,b,v)}. The u-v separating vertex set {t,s,a} shows that P is maximum-size collection.

5.3.11

Since S_{UV} is a u-v separating set, each u-v path in P_{uv} must include at least one vertex of S_{uv}. Since the paths in P_{uv} are internally disjoint, no two of them can include the same vertex. Thus, the number of internally disjoint u-v paths in G is at most $|S_{uv}|$. Therefore, if $|P_{uv}| = |S_{uv}|$, P_{uv} has the maximum size.

Coding Part

The question is asking to implement the Prim Algorithm

The Prim' Algorithm will produce the MST. To implement the Prim:

1. Transfer a graph into its adjacent matrix
2. Initialize an array checklist to track on the vertex already visited
3. Initialize an array parent to track on the vertex' parent (i.e. the edge)
4. Initialize an array v to store the frontier edge weight of current vertex
5. Growing the tree:
   Pick the 1st vertex u as starting point, and set v[u] = 0; parent[u] = -1;
   #Vertices = n
   Iterates the building process n-1 times:
      Pick the vertex s with minimum v[s]
      Mark s as visited
      Iterates other vertices x:
         If s and x are adjacent to each other and x is unvisited and edge between s and x has smaller weight than other edge on s:
            Record the weight of edge between s and x (v[x])
            And mark s as the parent of x

The input is the adjacent matrix of a graph

The output is the list of edges that Prim would pick

Output result:

<terminated> MainTest (1) [Java Appl

Edge	Weight
s - a	6
a - b	5
b - c	4
g - d	4
c - e	4
e - f	3
f - g	6

4.3.1

Edge	Weight
c - a	2
c - b	4
s - c	5
e - d	1
f - e	3
a - f	4
f - g	2

4.3.2

<terminated> MainTest (1) [Java App

Edge	Weight
c - a	5
a - b	7
f - c	7
a - d	7
b - e	6
s - f	6
e - g	5
g - h	9

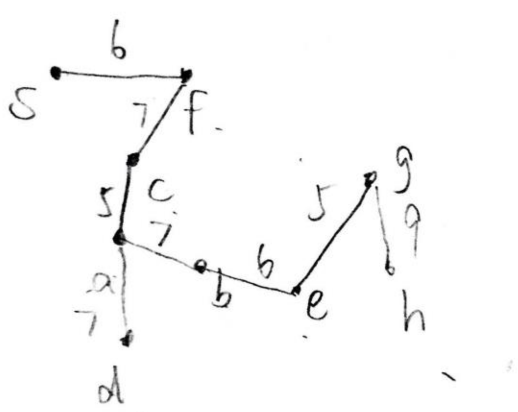
4.3.3

Edge	Weight
s - a	6
a - b	7
a - c	5
s - d	8
b - e	6
c - f	7
h - g	7
e - h	5

4.3.4

Verification of correctness:

Hand produced prim MST of 4.3.3:



Program output on 4.3.3:

Edge	Weight
c - a	5
a - b	7
f - c	7
a - d	7
b - e	6
s - f	6
e - g	5
g - h	9

The program pick the same edge set building the MST as doing it by hand

As a conclusion, the Prim algorithm would produce an MST successfully.