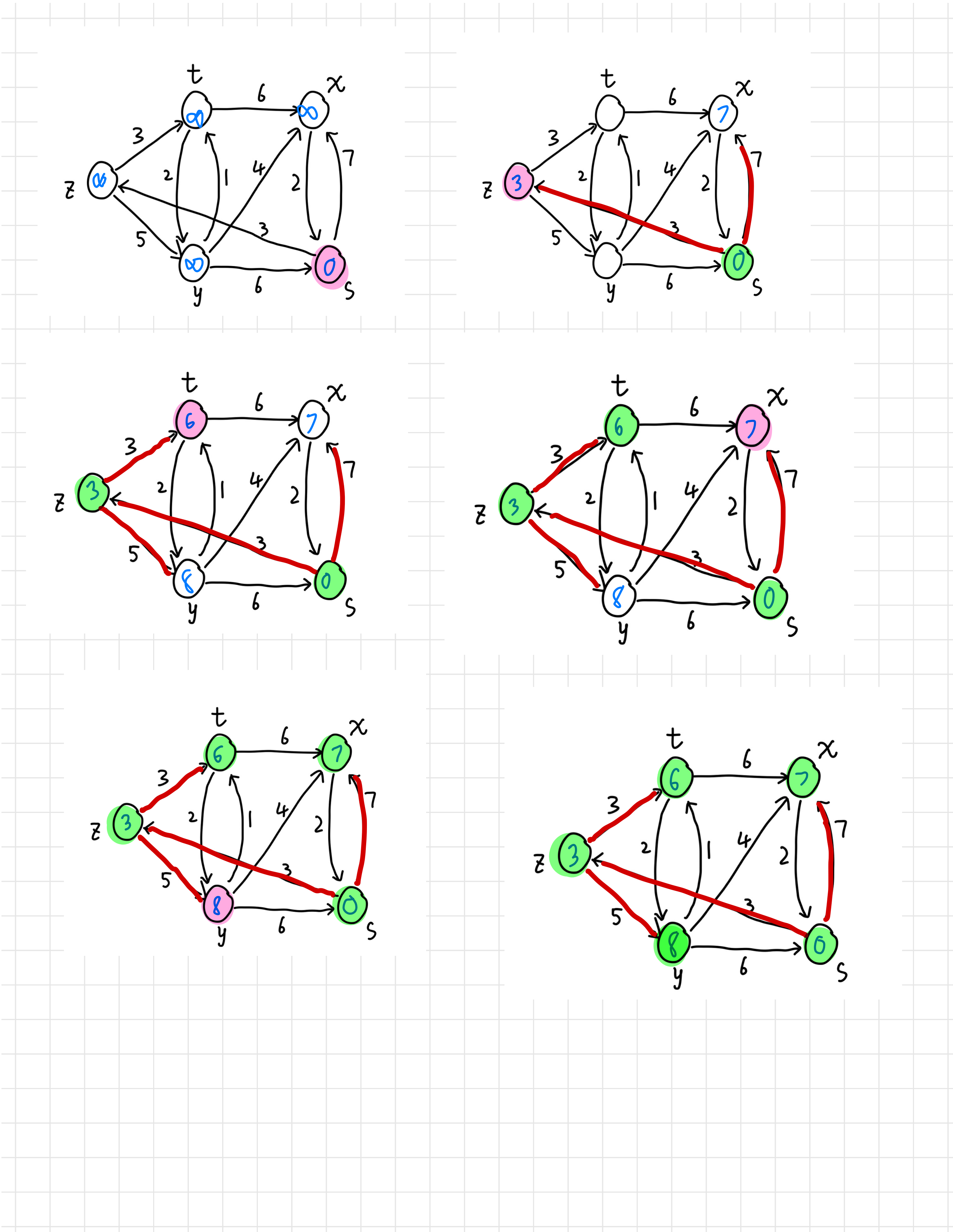
Algorithms Homework 5

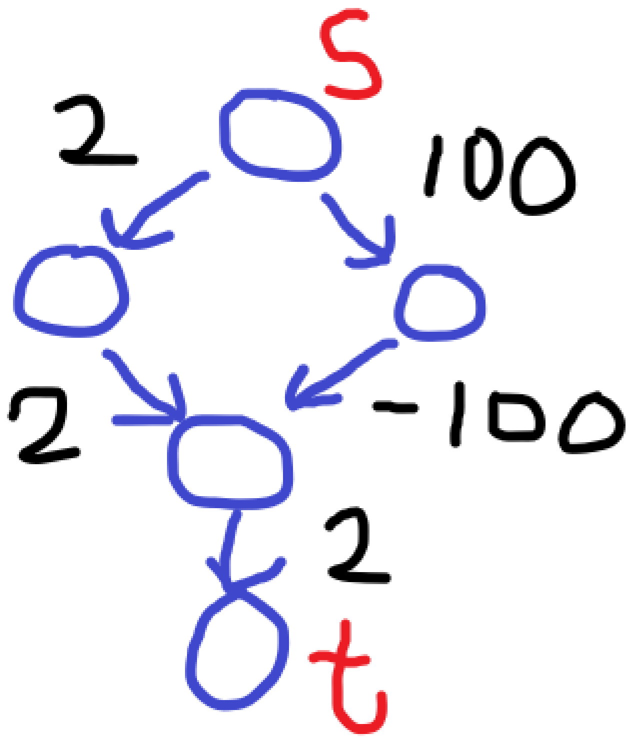
BRANDON YUEN, by161, Section 06
RYAN LEE, rsl82, Section 07
JAMES ALBA, jma390, Section 07

This is a joint assignment between Brandon Yuen, Ryan Lee, and James Alba
December 2022

Problem 1: Brandon Yuen, by161, section 06, Ryan Lee, rsl82, section07, and James Alba, jma390, section 07



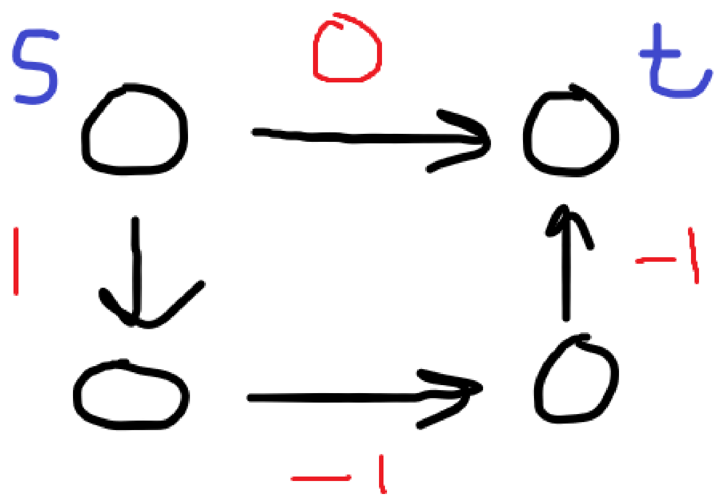
1.



2.

Actual Shortest: 2
Shortest by Dijkstra: 6

3. Graph G:



Actual shortest distance: -1
Distance using bad algorithm: 0

4. D.insert: O($|V|$) times * O(log($|V|$) = O($|V|$ * log($|V|$))
D.decrease-key: O($|E|$) times * O(1) = O($|E|$)
D.delete-min: O($|V|$) times * O(log($|V|$) = O($|V|$ * log($|V|$))

Total : O($|E|$ + $|V|$*log$|V|$) = O($|V|$*log$|V|$)
Comparing this to O($|E|$log($|V|$)):
Edge dense E = $|V|^2$
$|V|$*log$|V|$ = $|V|$*log$|V|$
$|E|$log($|V|$) = $|V|^2$ * log$|V|$
Therefore, in terms of a graph that is edge dense, our fancy data structure is more efficient.
Edge sparse E approx. $|V|$
$|V|$*log$|V|$ = $|V|$*log$|V|$
$|E|$log($|V|$) = $|V|$ * log$|V|$
Therefore, in terms of a graph that is sparse, our fancy data structure preforms the same.
We can then say that we find that both data structures will preform the same when $|E| = |V|$.
Additionally, we find that when $|E|$ is greater than $|V|$, then our fancy data structure preforms better.
The opposite is true: when $|V|$ is greater than $|E|$, then our fancy data structure preforms worse.

Problem 2: Brandon Yuen, by161, section 06, Ryan Lee, rsl82, section07, and James Alba, jma390, section 07

1. s in the algorithm can be any node in the graph G.

Algorithm 1 CheckBipartite(G,s)

```
u ← BUILDDICTIONARY()
v ← empty queue
u.add(s,red)
for node n : all neighbors of s in graph G do
    v.insert((n,blue))
end for
while v is not empty do
    (key,value) ← v.pop()
    if u.search(key) != NIL AND u.search(key) != value then
        return false
    end if
    u.add(key,value)

    if value == blue then
        color = red
    else
        color = blue
    end if

    for node n : all neighbors of key in graph G do
        v.insert((n,color))
    end for
end while
return true
```

Problem 3: Brandon Yuen, by161, section 06, Ryan Lee, rsl82, section07, and James Alba, jma390, section 07

1. I Don't Know.

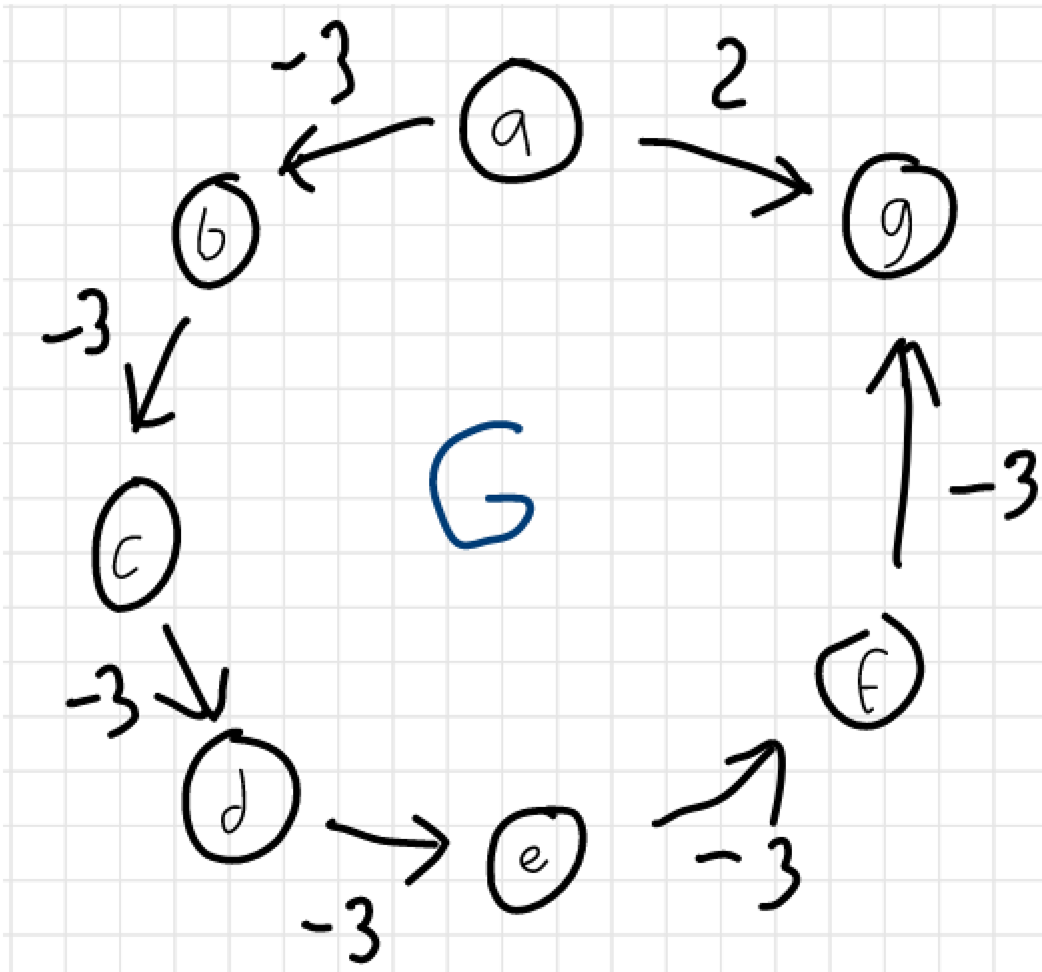
Problem 4: Brandon Yuen, by161, section 06, Ryan Lee, rsl82, section07, and James Alba, jma390, section 07

1. We assume that there are no cycles that have less than 10 edges, to prove this, we would keep traversing with BFS, to check if there are any cycles in each layer (adjacent nodes), based on our assumption we would have to keep traversing until the 5th layer (This gives us a cycle with at most 10 edges since $2 * 5 = 10$ because one child branch has a total of 5 vertices connected with 4 edges, the 5th edge would be from connecting one child branch to another child branch with 4 edges. The other branch would get their 5th edge by connecting to the other side of the previous child branch, summing up to a total of 10 edges). But this gives us $100^5|V|$ nodes that cannot be adjacent to each other. Which is impossible since we only have $|V|$ amount of vertices and $100^5|V| > |V|$.

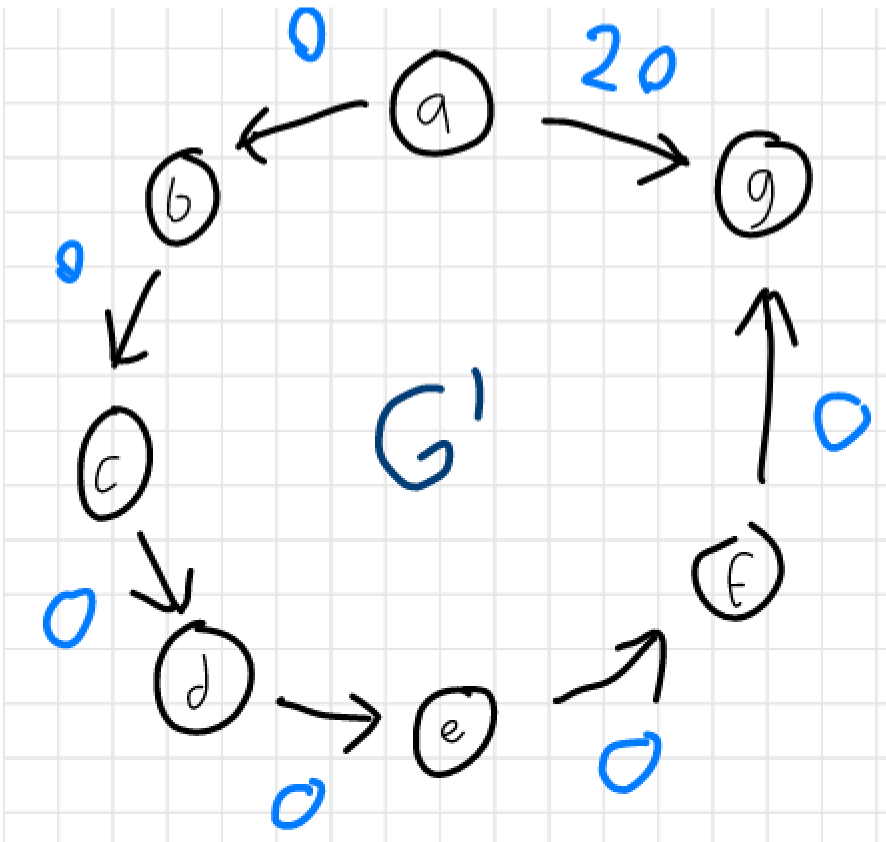
2. Assume that there is graph G that has n vertices.
   V0 is outgoing all of the other vertices. (V1,V2...Vn)
   V1 is outgoing all of the other vertices except V0. (V2,V3...Vn)
   V2 is outgoing all of the other vertices except V0,V1. (V3,V4...Vn)
   .
   .
   .
   Vn has no outgoing edge but has n-1 incoming edges.

   Every vertex has n-1 number of out-neighbor+in-neighbor but it cannot make cycle. Because there is no way to go to smaller number, it just goes down.

Problem 5: Brandon Yuen, by161, section 06, Ryan Lee, rsl82, section07, and James Alba, jma390, section 07

1. 

Price functions:
$\phi(a) = 0$
$\phi(b) = -3$
$\phi(c) = -6$
$\phi(d) = -9$
$\phi(e) = -12$
$\phi(f) = -15$
$\phi(g) = -18$



Maximum edge weight: 20