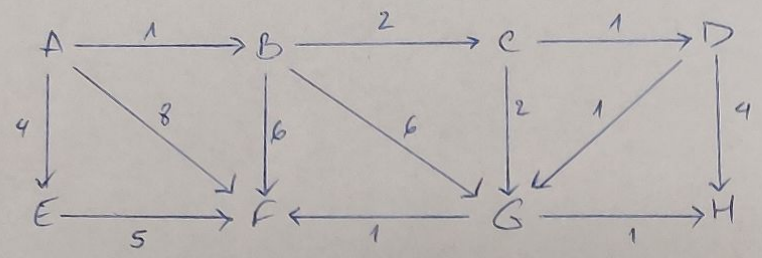
CS 216

Quan Nguyen

HW 7: graph paths

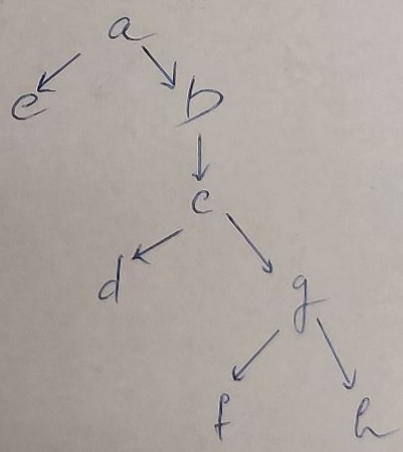
A.1



	Dist						Prev					
	0	1	2	3	4	5	0	1	2	3	4	5
a	0						∅					
b	∞	1					∅	a				
c	∞		3				∅		b			
d	∞			4			∅			e		
e	∞	4					∅	a				
f	∞	8	7			6	∅	d	b			g
g	∞		7	5			∅		b	c		
h	∞				8	6	∅				d	g

Loop	Priority Queue
0	~~a~~ b c d e f g h
1	~~b~~ e f c d g h
2	~~e~~ f g d h
3	~~d~~ e g f h
4	~~e~~ g f h
5	~~g~~ f h
6	~~f~~ h
7	~~h~~
8	∅

b)



1.3

```
Method():
    HashTable<key; Boolean> u_neigh, v_neigh
    for u in V:
        u_neigh.put(u, True if it is u's neighbor else False)
        count = 0
        for v in V:
            v_neigh.put(v, True if it is v's neighbors else False)
            u_neigh.put(v, False)
            v_neigh.put(u, False)
            for (Key k : V):
                if u_neigh.get(k) && v_neigh.get(k):
                    count++
            if count == 2:
                return True
    return False
```

/* How it works:

A — B
| |
D — C

Randomly choose A and C ($O(|V|^2)$)
so that they share exactly 2 neighbors (B, D)
(loop through takes $O(|V|)$

$\Rightarrow \Sigma = O(|V|^3)$

*/

4.5

// using Dijkstra

```
Method (s, t): from s to t
    for u in V:
        dist(u) = ∞
        prev(u) = ∅

    HashTable < Key, Integer> num_path
    for u in V:
        num_path.put(u, 0)

    dist(s) = 0

    PQ = pq(V)

    while (PQ ≠ ∅):
        u = PQ.deleteMin

        for (u, v) ∈ E:
            if dist(v) > dist(u) + len(u, v): // find shorter path
                dist(v) = dist(u) + len(u, v)
                num_path.put(v, num_path.get(u)) // num_path stays same
            elif dist(v) == dist(u) + len(u, v)
                num_path.put(v, num_path.get(v) + 1) // add num_path when find another path

            PQ.decreaseKey(v, dist(v))

    return num_path.get(t)
```

4.12

```
Method (s, t)    // edge e connects s and t
    if t is not s's neighbor:
        return
    tmpGraph = copy(this Graph)
    tmpGraph.vertices.get(s).remove(t)    // remove edge e

    // use Dijkstra
    V' = vertices of tmpGraph
    E' = edges of tmpGraph

    for u in V':
        dist(u) = ∞
        prev(u) = ∅
    dist(s) = 0
    PQ = pq(V')

    While PQ ≠ ∅:
        u = PQ.deleteMin
        for (u, v) in E':
            if dist(v) > dist(u) + len(u, v)
                dist(v) = dist(u) + len(u, v)
                prev(v) = u
                PQ.decreaseKey(v, dist(v))    → O(1)

    return dist(t) + len(s, t)
```

$\sum = O(|V|^2) + O(|V| + |E|)$

worst case: $|E| \approx |V|^2$

$\Rightarrow \sum = O(|V|^2)$

$\rightarrow O(|V|) \approx O(|V'|)$

$\rightarrow O(|V|)$

$\rightarrow O(|E_u'|) \approx O(|E_u|)$