Design and Analysis of Algorithms
CS-535
Assignment 1

1.(a)
We will use BFS Algorithm to find the shortest path from Node S to Vertex V.
BFS uses a queue to maintain the nodes with a time complexity of O(|V| + |E|).

The Algorithm is :
Lo = {s}
L1 = all neighbours of Lo
L2 = all nodes that do not belong to Lo or L1, and that have an edge to a node in L1
Li+1 = all nodes that do not belong to an earlier layer, and that have an edge to s node in Li.

1. (b)
To obtain Inclusion-wise maximal edge-disjoint shortest path in O(|V| + |A|) time we first use the BFS algorithm. We calculate the total number of edges present in the shortest path from s to t. This gives us a running time complexity of O(|V| + |A|).
Inclusion-wise maximal set is a set which is not a superset of any other set in the collection and therefore, we apply DFS to get the edge disjoint shortest path from Node S to t. By doing this we get a running time complexity of O(|V| + |A|).

2.

Approach 1:

2. To obtain a k-circuit decomposition with $k \leq m$ we can use bellman ford algorithm where we assume that all the circuits are negative. This way we can satisfy the given condition.

Approach 2:

We are given ⇒

$D = (V, A; l)$ has positive edge lengths satisfying that for each $v \in V$

$$\sum_{(u,v) \in A} l(u,v) = \sum_{(v,u) \in A} l(v,u)$$

we know that $k > 0$ and consists of k pairs (C_i, ε_i) of circuit and positive real for $1 \leq i \leq k$ satisfying that for each $a \in A$.

$$l(a) = \sum_{1 \leq i \leq k : a \in C_i} \varepsilon_i$$

We can decompose circuits using Karp's Algorithm with $O(mn)$ time complexity.

We decompose the circuits as minimum mean circuit. This will give us shortest walk ending from s to t, with the required number of edges.

3.

3. We know, $C^{0}(u,v) = 0$
$C^{\lceil \log C \rceil}(u,v) = w(u,v)$
$C \leftarrow$ maximum weight of an edge.

Let,
$c^{i}(u,v) \leftarrow$ weight resulting from i most significant bits of $c(u,v)$.

$d^{i}(u) \leftarrow$ distance of s to u wrt c^{i} which are i most significant bits of the costs.

$\Rightarrow$ Let $p(u) = 0$ for all nodes $u \in V$ and $i = 0$.
From the above, we get
$c_p^{0}(u,v) = p(u) - p(v) + c^{0}(uv) = 0$

$\Rightarrow$ for all $uv \in E$ (edges) and $d^{0}(u) = 0$ for all u go from i to $i+1$.

* $p(u) = d^{i}(u)$ and calculate reduced costs $c_p^{i}(u,v)$ wrt $p(u) = d^{i}(u)$.

* From $d(u)$, we can obtain $c_p^{i}(u,v) = p(u) + c^{i}(u,v) - p(v) = d^{i}(u) + c^{i}(u,v) - d^{i}(v) \geq 0$

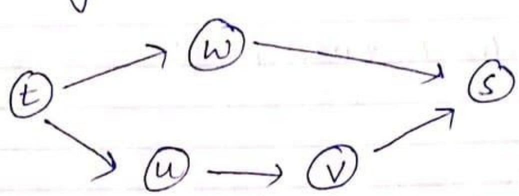
We can observe that if uv is on a shortest path, $d^{i}(v) = d^{i}(u) + c^{i}(uv)$ which gives $c_p^{i}(uv) = 0$.

$\Rightarrow$ Hence, we can change the weights if p is the arbitrary potential function, and L_p is the edge length. Then for any $a \in C \Rightarrow L_p(a) = 0$.
Hence Proved.

4.

4. We are given $D = (V, A)$ with two distinct nodes s & t in V.

assuming that



We know that $p(s) = n$ and $p(t) = 0$.
Also, $\forall (u,v) \in A$; $p(u) - p(v) \leq 1$.
where u, v are the starting and ending.
we can also say
$p(u) - p(v) \leq 1$; $(u,v) \in A$
$\Rightarrow p(s) - p(t) \leq 1$
$p(s) - 0 \leq 0$; [because $p(t) = 0$]

In the above condition, the loop will check for all possibilities of s which will never be true.

Hence proved that D has no $s-t$ path if and only if there exists a non negative integer-valued labelling p of the nodes satisfying the given conditions using the concept of contradiction.

5.

- A shortest cycle containing the node S must be a path from the node S to the vertex V.
- We run Dijkstra's algorithm to find the shortest distance d(s,v) from node S to each vertex V.
- The shortest cycle containing S is found by calculating min v∈V{s(s,v) + ‘(v,s)}.
- The time complexity is the running time of Dijkstra which is O(m + n log n) and the time to calculate d(s,v) + ‘(v,s)
- We can use the adjacency list of Vertex V to compute all the values and then take the minimum of O(m+n).

Therefore, the total time taken is O(m + n log n).

6.

6. Given:

$\Delta = (V, A; L)$ in which all but one arc (u,v) have non-negative lengths.

$u \to v$ = unique arc with negative weight.

Inorder to test whether Δ has a negative circuit we remove edge $u \to v$ from Δ and let Δ' denote the resulting graph. Now, we know that Δ' has no negative length edges.

If Δ has a negative length circuit then it must contain the arc $\cancel{u \to v}$ $x \to y$; here for any nodes x and y, let $\Delta(x,y)$ and $\Delta'(x,y)$ distance denote the distances from x to y in Δ and Δ' respectively.

The shortest length circuit containing this arc can be seen to consist of a shortest path P from y to x in Δ' together with the arc $x \to y$.

The length of this circuit is $dist'(y,x) + w(x \to y)$. Δ has a negative length circuit iff this quantity is negative. Thus, we can check if Δ has a negative ciruit by computing $dist(y,x)$ in Δ' via Dijkstra's Algorithm.
This will give us a running time complexity of $O(m+n \log n)$.