P1

Given: $n \times n$ Matrix A, $A_{ij} = 2^{l}$, $l \in \mathbb{Z}$

Want: $a_1, a_2, \dots, a_k$ s.t. $a_k = a_1$,

$$\text{and } \prod_{i<k} A_{a_i, a_{i+1}} \geq 1$$

$$\iff \sum_{i<k} \log A_{a_i, a_{i+1}} \geq 0$$

We then construct a graph with n nodes.

$V := [1, n]$

E: def adj(u):
    return V

W: def w(u,v):
    return $-\log A_{a_i, a_{i+1}}$

Then problem reduced to:

find cyclic path $a_1 \sim a_k$ s.t.
sum of weight < 0.

So we can use Bellmen-Ford on any s to find the existence of such negative weight cycle.

Time: $O(VE) = O(n \cdot n^2) = O(n^3)$

P2

1. Change all weights > L to ∞. run Dijkstra.

2. Find Min L(G, w, s, t):

    $Q \leftarrow$ min-heap(V)

    $d[s] \leftarrow 0$; set others to ∞

    while $|Q| > 0$:

        u = extract-min(Q)

        if u is t: return $d[u]$

        for v in adj(u):

Time: O(|V|+|E|) log(|V|))

            $d[v] = MIN(d[v], MAX(d[u], w(u,v)))$

Claim: when u is extracted, $d[u] = \delta(s,u)$ (where $\delta(a,b)$ denotes max-weight of minimum-capacity path from a to b)

Proof: Let $u \neq s$ be the first violation, and Q right before extract.

Let $s \to \dots \to x \to y \to \dots \to u$ be a path that requires minimum capacity. where $s \notin Q$ and y is first in Q

Note $d[x] = \delta(s,x)$ ($\because$ u is first violation)

$d[y] = MIN(w(x,y), \delta(s,x)) = \delta(s,y)$

But we have: $d[u] < d[y]$

$MAX(\delta(y,u), \delta(s,y)) < \delta(s,y)$

(contraction)

End Proof

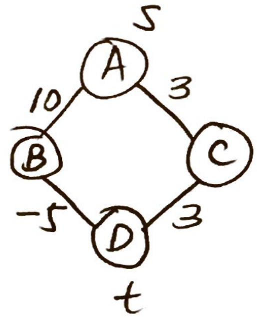
P3

```
d[c][|V|] ← all ∞
d[*][s] ← all 0
Q = min-heap( (c, V), key=d )
while |Q| > 0 :
    (i, u) = extract-min(Q)
    if (i,u) is (c,t):
        return d[c][t]
    for v ∈ adj(u):
        d[i][v] = MIN(d[i][v], d[i][u]+w(u,v))
        if i < c :
            d[i+1][v] = MIN(d[i+1][v], d[i][u])
```

Idea: just adding extra demension to the Graph, essentially still Dijkstra Algorithm.

Time: O((|V| + |E|) c log(|V| c))

P4



d[D] will be 6

whereas $\delta(D)$ is 5