(part c) Define a function ComputeAFromT(T) which returns the adjacency n by n matrix A of G, by performing BFS on n by n matrix T.
ComputeAFromT(T) takes O(n^2) since BFS on a n by n matrix(T) to find all entries less than 1 takes O(n^2). And the function returns a correct adjacency matrix A of G, since every (i,j) ∈ E has T[i,j]<1, and every (i,j) ∉ E has T[i,j]≥1, so we can find all (i,j) ∈ E to build A.
Define a function ComputePFromD(D) which takes an n by n distance matrix D of G and returns a n by n matrix P, by performing "mod 2" on each of D's entries.
ComputePFromD(D) takes O(n^2) since perform "mod 2" on each of D's entries takes n^2 times. And the function returns a correct matrix P, by the definition of P.

```
Def ComputeP(Dsq,T):
        A := ComputeAFromT(T);
```

1 Question 2.3

Claim:
T[i,j]<1 iff (i,j) ∈ E.
First,if T[i,j]¡1, then (i,j) ∈ E.
prove by contradiction:
If (i,j) ∉ E, then i $\notin \{k : (k, j) \in E\}$. So, ∀ k in $\{k : (k, j) \in E\}$, D_{sq}[i,k] ≥ 1. And since deg(j) = $|\{k : (k, j) \in E\}|$, we now have $\sum_{k:(k,j)\in E} D_{sq}[i, k] \geq |\{k : (k, j) \in E\}| = deg(j)$, so T[i,j] ≥ 1. Then, by contradiction, if T[i,j]¡1, then (i,j) ∈ E.
Second,if (i,j) ∈ E, then T[i,j]¡1. proof:
(i,j) ∈ E, then ∀ k ∈ $\{k : (k, j) \in E\}$, since δ(k,j)=1, and δ(i,j)=1, we have δ(i,k) ≤ 2,implies D_sq(i,k) ≤ 1. And, i ∈ $\{k : (k, j) \in E\}$, means ∃ k ∈ $\{k : (k, j) \in E\}$, such that δ(k,i)=0. So, $\sum_{k:(k,j)\in E} D_{sq}[i, k] < deg(j)$, which implies T[i,j]¡1.
We are done.
When δ(i,j) ≥ 2, let b ∈ ℕ.
Claim: When δ(i,j) is even, assume δ(i,j)=2b. When δ(i,j) is odd, assume δ(i,j)=2b+1. We have $b \leq T[i, j] < b + 1$. Proof:
D_{sq}[i,j]=$\lceil \delta(i, j)/2 \rceil$, so D[i,j]=b if δ(i,j) is even, and so D[i,j]=b+1 if δ(i,j) is odd.For all k such that (k,j)∈E ,$\delta(i, j) - 1 \leq \delta(i, k) \leq delta(i, j) + 1$ since $\delta(k, j)$=1, so
when δ(i,j) is even,

$$D_{sq}[i, k] = \lceil \delta(i, k)/2 \rceil == \begin{cases} b & \delta(i, k) = (\delta(i, j) - 1) \ or \ \delta(i, j), \\ b + 1 & \delta(i, k) = \delta(i, j) + 1 \ . \end{cases}$$

when δ(i,j) is odd,

$$D_{sq}[i,k] = \lceil \delta(i,k)/2 \rceil == \begin{cases} b & \delta(i,k) = \delta(i,j) - 1, \\ b+1 & \mid \delta(i,k) = \delta(i,j) \ or \ (\delta(i,j)+1) \ . \end{cases}$$

Furthermore, there exist a k in the shortest path from i to j such that $\delta(i,k)$=$\delta(i,j) - 1$ (since δ(i,j) $\geq$ 2), which implies $D_{sq}[i,k]$=b for such k (from above piecewise functions). By the justification above,since T[i,j] = $\frac{1}{deg(j)} \sum_{k:(k,j)\in E} D_{sq}[i,k]$, b $\leq$ T[i,j] (since $D_{sq}[i,k] \geq b$) and T[i,j] $<$ b+1 (since $D_{sq}[i,k] \leq b+1$ and there exist a k such that $D_{sq}[i,k]$=b). Then we get b $\leq$ T[i,j] $<$ b+1 ①.
When $\delta(i,j)$ is even, we have[①, $D_{sq}[i,k]$=b and T[i,j]=0] ②.
When $\delta(i,j)$ is odd, we have [①, $D_{sq}[i,k]$=b+1 and T[i,j]=1] ③.
From above ② and ③, we know that Helper_ComputeP is correct.
// this function computes (returns) P[i,j] according to given T[i,j] and D_{sq}[i,k], where T[i,j] is from the given definition , D_{sq}[i,k] is the entry in squared graph's distance matrix.

```
Helper\_ComputeP(T[i,j], $D_{sq}$[i,k]):
        if (T[i,j]'s integer part == D_{sq}[i,k])
                return P[i,j]=0
        else // (T[i,j]'s integer part) +1 == D_{sq}[i,k]
                return P[i,j]=1
```

It runs in linear time. So we can construct function ComputeP(D_{sq},T).

```
ComputeP(D_{sq},T):
        for each [i,j]
                compute P[i,j] from
                Helper_ComputeP(T[i,j], D_{sq}[i,k])
        return P.
```

The above function is correct since Helper_ComputeP(T[i,j], D_sq[i,k]) is correct. And it runs linear time since each statement in for loop runs linear time, and the whole for loop runs O(n^2) times.

Done.