Solution 3

1、定义witness为运行**Bellman-Ford**算法n轮时距离依然减少的点。

(a) 证明：每一个负环上至少存在一个witness

假设存在一个环没有任何witness，则对于环上所有点u有$d_u + w_{u,v} \geq d_v$，对环上所有点求和$\sum(d_u + w_{u,v}) = \sum d_v \Rightarrow \sum w_{u,v} \geq 0$，与负环矛盾。

(b) 构造一张图使得至少存在一个witness不在负环上

$w_{1,1} = -1, w_{1,2} = 0$，点2不在负环上但是witness。

2、DAG（有向无环图）中的单源最短路可以在$O(|V| + |E|)$的时间内求解。求解步骤为：1、求出图的拓扑序；2、根据拓扑序对每个节点进行松弛。

(a) 写出算法的伪代码

```
int deg[N];
vector<pii> G[N];
vector<int> topo;
// toposort
void dfs(int u) {
    deg[u] = -1;
    topo.emplace_back(u);
    for (int v: G[u]) if (!--deg[v]) dfs(v);
}
void DAG_relaxation() {
    for (int i = 1; i <= n; ++i) if (!deg[i]) dfs(i);
    vector<int> dis(n + 1);
    for (int u: topo)
        for (auto[v, k]: G[u])
            dis[v] = min(dis[v], dis[u] + k);
}
```

(b) 说明为什么按照拓扑序的松弛得到的结果是正确的

按拓扑序更新到一个点时，前驱结点已经处理完，所以距离的计算是正确的。

3、**Dijkstra**算法只适用于求边权非负的图的最短路，如果存在负权边则算法的正确性不能保证。但是，假如运行**Dijkstra**多次呢？假设一张图中有$k-1$条负权边，但不存在负环，运行k次**Dijkstra**算法，每次运行结束后，从源点s向每个节点u连一条长度为$d(s,u)$的边，证明这样正确的求出源点到每个点的距离。

Algorithm: Dijkstra-Iteration

2.1 unmark all nodes
2.2 while not all vertices marked do
2.3 $\quad u \leftarrow$ unmarked vertex with least label $D(u)$
2.4 $\quad$ mark u
2.5 $\quad$ forall its out-edges (u, v) do
2.6 $\quad\quad D(v) \leftarrow \min\{D(v), D(u) + \ell(u, v)\}$
2.7 end

数学归纳法：第i轮后，最短路上$\leq i-1$条负边的点距离确定，且最短路上恰好i条边，并且最后一条是负边的点距离确定。

4、无向图G中x到y的最小瓶颈路是这样的一类简单路径，满足这条路径上的最大的边权在所有x到y的简单路径中最小。

(a) 证明：任意一棵最小生成树上x到y的路径就是原图G中x到y的最小瓶颈路

假设有更小的瓶颈边，则违反cut-rule。

(b) 最小生成树上x到y的路径一定是原图中x到y的最小瓶颈路，但是并不是x到y的所有最小瓶颈路都在某棵最小生成树上，构造一张图使得x到y的一条最小瓶颈路不在任何一棵最小生成树上。

$w_{1,2}=3, w_{1,3}=1, w_{2,3}=2, w_{3,4}=4$，$1 \to 2 \to 3 \to 4$的路径是最小瓶颈路，但不在MST上。

5^*（**挑战问题**：可2-3位同学组队回答，但在提交的答案中需写明每位同学的**具体**贡献）、最小权重比率环

(a) 说明对于一般的图，如何用**Bellman-Ford**算法使用$O(nm)$的时间判断图中是否存在负环。（不一定所有点都从源点s可达）

建一个超级源点，向每个点连一条长度为0的边，运行Bellman-Ford。

(b) 假设一张图不存在负环，说明如何用**Bellman-Ford**算法判断一张图是否存在零环。

给每条边加一个很小的$bias$，找负环。

(c) 定义一个有向环C的权重比率为

$$\rho(C) := \frac{\sum_{a \in C} w_a}{|C|}$$

，其中w_a为环上每条边a的权重。定义最小权重比率$\rho^*(G)$为图中最小的ρ值，即

$$\rho^*(G) := \min_{\text{cycles } C} \rho(C)$$

观察到：

$$\rho^*(G) = \max\{\alpha \in Q | \text{对图}G\text{中每条边的权重都减去}\alpha\text{后图中就不存在负环}\}$$

其中Q表示有理数域，并假定图中每条边的权重$w \in [-M, M]$，且w为**整数**。给出一个算法使用$O(nm(\log M + \log n))$的时间找到这样的一个最小权重比率$\rho^*(G)$及其对应的环（注意：二分值域为K的整数的时间复杂度为$O(\log K)$，但是对于实数的二分可能不会终止。但我们是否可以在二分到达某一个特定的实数精度时停止，然后还原最小权重比率环？）。

二分答案，$\frac{\sum w_a}{|c|} \le \lambda \Rightarrow \sum(w_a - \lambda) \le 0$，每条边重新赋值$w_a - \lambda$，判断是否存在负环。因为任意环的权重比率都可以写作$\frac{p}{q}, q \le n$，任意两个环的差$|\frac{p_1}{q_1} - \frac{p_2}{q_2}| = |\frac{p_1 q_2 - p_2 q_1}{q_1 q_2}| \le \frac{1}{n^2}$，二分到$< \frac{1}{n^2}$停止。