hw9

Q1

Question:

Floyd-Warshall 算法的空间需求是 $\Theta(n^3)$，因为要计算 $d_{ij}^{(k)}$，其中 i，j，$k = 1$，1，2，…，n。有同学提出可以将 Floyd-Warshall 算法修改为如下形式，从而将 Floyd-Warshall 算法的空间需求降低到 $\Theta(n^2)$。请问新的 Floyd-Warshall-New 算法是否正确，如正确，请证明，否则，请给出反例。

Algorithm 1 FLOYD-WARSHALL-NEW(W)

$n = W.rows$
$D = W$
for $k = 1$ to n **do**
  for $i = 1$ to n **do**
    for $j = 1$ to n **do**
      $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$
    end for
  end for
end for
Return D

Answer:

正确

由Floyed算法的动态规划转移方程

$dp[i][j][k] = min(dp[i][j][k-1], dp[i][k][k-1] + dp[k][j][k-1])$

$dp[i][j][0] = w_{ij}$

在伪代码中d_{ij}即为转移方程中的dp，dp[i][j][k]表示从节点i到节点j，中间结点只用前k个结点的最短路径长度

由转移方程，第k个状态可以完全由第k-1个状态计算获得，因此由压缩数组的思想，可以将三维dp空间降低为二维，dp[i][j]表示当前从i到j的最短路径长度，因此算法是正确的

C++实现如下

```cpp
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
const int N = 210, INF = 1e9;
int g[N][N];
int n, m, Q;
void floyd()
{
    for(int k = 1; k <= n; k++)
    {
```

```
        for(int i = 1; i <= n; i++)
        {
            for(int j = 1; j <= n; j++)
                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
        }
    }
}
int main(void)
{
    cin >> n >> m >> Q;
    // 初始化
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            if(i == j) g[i][j] = 0;
            else g[i][j] = INF;
        }
    }
    while(m--)
    {
        int a, b, c;
        cin >> a >> b >> c;
        g[a][b] = min(g[a][b], c);
    }
    floyed();
    for(int i = 0 ;i < Q; i++)
    {
        int a, b;
        cin >> a >> b;
        if(g[a][b] > INF / 2) puts("impossible");
        else cout << g[a][b] << endl;
    }
    return 0;
}
```

Q2

Question:

给定一个带权有向图 $G=(V,E,w)$ 和顶点 $s\in V$。图 G 具有以下性质，对于每个顶点 $v\in V$，从 s 到 v 的某个最小权重路径最多经过 k 条边，请描述一个时间复杂度为 $O(|V|+k|E|)$ 的算法来计算从 s 到每个 $v\in V$ 的最短路径权重。

Answer:

初始化一个距离数组dist，dist[i]表示从s到顶点i的最短距离，对所有除了s以外的顶点初始化为∞，dist[s] = 0

```
for i = 1 to k do
    for each edge(u,v) in G.E do
        RELAX(u, v, w)
```

执行完k次迭代后，dist包含所有从s出发到其他顶点的最短路径权重，至多包含k条边

时间复杂度为 $O(V+kE)$

Q3

Question：

假定在一个权重函数为 w 的有向图 G 上运行 Johnson 算法。证明：如果图 G 包含一条权重为 0 的环路 c，那么对于环路 c 上的每条边 (u,v)，$\hat{w}(u,v)=0$。

Answer：

由Bellman-ford算法的运行结果，对环路c上的每条边(u,v)，有$h(u)=h(v)$

故$\hat{w}(u,v)=w(u,v)+h(u)-h(v)=w(u,v)=0$

因此，如果图G包含一条权重为0的环路c，那么对于环路c上的每条边(u,v),$\hat{w}(u,v)=0$

Q4

Question：

小曹同学有一辆自行车，他想参加一年一度的自行车登山赛。他手里有一张地图，上面描绘了：

- n 个站点，每个站点 x_i 都标有其正整数海拔高度 e_i 和是否包含补给站；
- 连接它们之间的 r 条道路，每条道路 r_j 都标有正整数 t_j，表示小曹沿任一方向行驶所需的行驶时间。

站点连接多条道路，但每个站点连接的平均道路数小于 5，即 $(r \le 5n)$。小曹需要从起点 s 到终点 t，同时要满足以下条件：

- 小曹的体力值为正整数 $g < n$：他最多有 g 单位的体力（开始时体力值是满的）。他可以在沿途任何标有补给站的站点进行休息并恢复体力（任何整数单位）。他每恢复一单位体力需要休息 t_G 的时间。
- 骑车只有在上坡时才消耗体力。具体来说，如果他从海拔高度分别为 e_i 和 e_j 的站点 x_i 驾驶到交叉点 x_j，如果 $e_j > e_i$，小曹将使用 $e_j - e_i$ 单位的体力值，否则将不消耗体力。

给定自行车登山赛的地图，请描述一个 $O(n^2 \log n)$ 时间复杂度的算法，以返回一条到达终点的最快路线，并始终保持比赛过程中严格保留正量的体力值（假定一定存在这样的路线）。

Answer：

问题等价于找到路径p，使得$\forall(x_i, x_j) \in p, e_j - e_i \leq g$

初始化一个优先队列，其中元素是三元组（时间、位置、体力值）。将（0，起点，g）添加到队列中，表示从七点开始，时间为0，体力值为g

初始化一个二维数组dist，dist[i][j]表示到达位置i，体力值为j时的最短时间。初始化dist[i][j]设置为无穷大，dist[起点][g]，将其设置为0

当队列不为空时，

- 取队头元素
- 如果位置是终点，返回时间
- 对于从当前位置出发的每条边，如果体力值≥消耗的体力是，$g + e_j - e_i + t_G < dist[next\_index][g - e_j + e_i + t_G]$，进行一次松弛操作

队列优化的dijkstra算法对全源顶点的时间复杂度是$O(n^2logn)$的