Chapter 25

25.1-4

要证明算法中矩阵的乘法满足结合率，要满足：

$$(L_iL_j)L_k = L_i(L_jL_k)$$

Answer:

$$\begin{aligned}
(L_iL_j)L_k = M_1 \\
L_i(L_jL_k) = M_2 \\
M_1[a][b] &= min_{1\le m\le n}(L_iL_j)[a][m] + L_k[m][b] \\
&= min_{1\le m\le n} min_{1\le k\le n}(L_i[a][k] + L_j[k][m] + L_k[m][b]) \\
&= min_{1\le k\le n}(L_i[a][k] + min_{1\le m\le n}(L_j[k][m] + L_k[m][b])) \\
&= min_{1\le k\le n}(L_jL_k)[k][b] + L_i[a][k] \\
&= M_2[a][b] \\
M_1[a][b] = M_2[a][b]
\end{aligned}$$

所以满足结合率

25.1-5

我们很容易在 all pairs 的最终结果矩阵中，通过某一行的数据就可以构造出来单元最短路树，所以如果只求一颗单元最短路树，我们要的就是矩阵的那一行而已。所以我们只用一个 vector v 表示所有点到源的距离，首先初始化，跟矩阵的初始化一样。然后 $v_{i+1} = v_iW$ 当算到 v_{n-1} 就得到了正确的答案, 所以乘（n-2）次就行了。

与 bellman ford 算法的相似性：

相乘就相当于 bellman ford 的 relax

25.1-6

一个 n*n 的矩阵，我们需要算出没每个位置的父节点，所以我们需要一个二重循环，

每 n^2 次循环的每一次循环中，我们知道 $L_{ij} = Lik + w\ k\ j$，所以枚举 n 次 k, 我们就能找到答案。

25.1-9

25.1-10

如果有负圈，那么当当一个 circle 之后 $W[i][i] < 0$; 所以在 slow all pairs 算法中每 extend 一次我们就检查一下是否有 W[i][j]<0 如果有那么当前的循环的 i 就是圈的长度

25.2-2

矩阵初始化的时候也是有边就赋值为 1(自身也是 1), 没边赋值为 0, 算 min 的时候改为 $l[i][j]|(l[i][k]\&w(i,k))$ 最后矩阵中为 1 的就表示有边

Algorithm 1 FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

```
n=W.rows
l[1] = W
m = 1
while m<n-1 do
    let l[2m] be a new n*n marticx
    l[2m] = EXTENDED-SHORTEST-PATHS(l[m],l[m])
    m=2m;
end while
l[n] =EXTENDED-SHORTEST-PATHS(l[m],l[m])
if l[m] != l[n] then
    has negative-weight circle
end if
```

25.2-4

我们将原来的 $O(n^3)$ 和现在的 $O(n^2)$ 对比，现在的算法只用了一个矩阵，但是看关系式子

$$d_{ij} = min(d_{ij}\ d_{ik}) + d_{kj}$$

原来的算法的式

$$d_{ij}^{m} = min(d_{ik}^{m-1}, d_{ik}^{m-1} + d_{kj}^{m-1})$$

因为在更新新的矩阵 d_{ij} 之前 d_{ij} 的值其实就是 d_{ij}^{m-1}
而 $d_[ik] + d_{kj}$ 至少不必 $d_{ik}^{m-1} + d_{kj}^{m-1}$ 小，因为我们要的是更小，所以
$d_[ij]$ 得到的结果至少不必的另一个算法的 d_{ij}^{m} 小，所以最后依然能得到正确的结果。

25.2-6

只要矩阵中对角线上有小于零的值，就有副回路，应为，如果有负回路，那么一圈下来，点到自身的距离就会小于 0.

24.2-8

首先构造一个 n*n 的矩阵 M，有边为 1, 没边为 0，$|v|$ 个点每一点 v，做一个循环，然后 dfs（v,v）每一个 dfs(v,v) 是 $O(E)$, 所以总代价是 $O(EV)$

Algorithm 2 dfs(v_1,v)

```
M[v1][v]=1
for each u in G.adj[v1] do
    if !vis[u] then # let v is be a n-array,to index if vertice u is visited
        dfs(u,v)
    end if
end for
```

25.3-2

加上这个点，我们肯定能判断出来是否有负回路，但是如果我们只是从原来图上任意选择一个，如果图不是联通的，那么有可能检测不到负圈。
如果没有负回路，那么得到 $h(v) = \delta(s, v)$，能够得 b 到符合条件的 h(v) 函数

25.3-3

相等，因为 h(v)=0 对于任意一点 v
因为 $h(v) = \delta(s, v)$，但是因为所有的边都是正值，所以 $\delta(s, v) = 0$

25-2

a

最小堆和最大堆的效率是一样的，所以参考 $6-2$, $Insert\ extract - min\ decrease_key$ 的代价分别为：$O(\log_d n), O(d\log_d n), O(\log_d n)$
if $d = \theta(n^\alpha)$ 三个代价分别为 $O(\frac{1}{\alpha}), O(\frac{n^\alpha}{\alpha}), O(\frac{1}{\alpha})$
$fibonacci\_heap$ 的三个操作代价以此为 $O(1)$ $O(\log n), O(1)$

b

另 $d = n^\varepsilon$, 因为总 n 次 $ECTRACT - MIN$ 和最多 E 次 $descease - key$ 操作，所哟 $O(nd\log_d n + E(1/\varepsilon))$.
$O(\frac{n^{1+\varepsilon}+E}{\varepsilon})$
$O(n^{1+\varepsilon})$
$O(E)$

c

每一个节点都用 b 中的算法

d

转换 weight function, 增加一个节点，然后计算新的权重，和书中的方法一样。