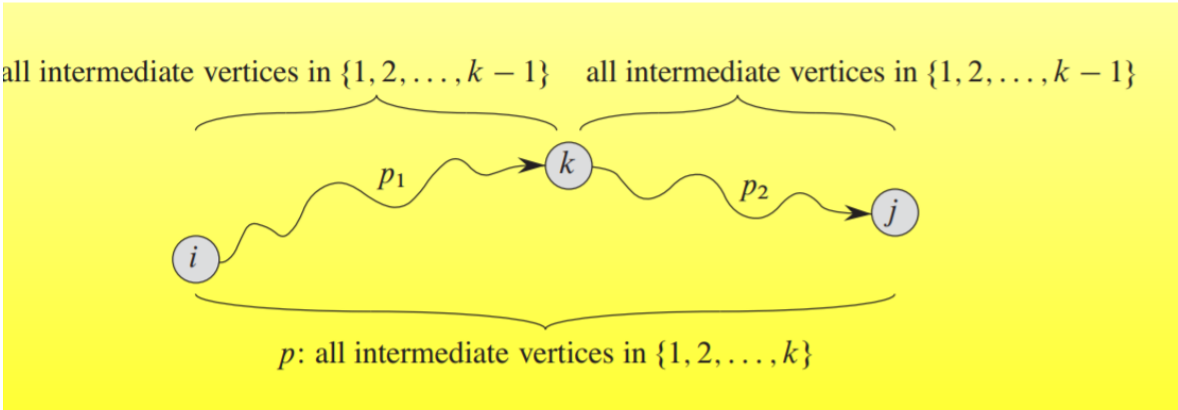
hw8

1. 给定 $G=(V,E)$ 是一带权重且没有权重为负值的环路的有向图，对于所有的结点 $v \in V$，从源结点 s 到结点 v 之间的最短路径中，包含边的条数的最大值为 m。请对算法 BELLMAN-FORD 进行简单修改，可以让其在 $m+1$ 遍松弛操作之后终止，即使 m 不是事先知道的一个数值。

根据上限理论，我们知道在 m次迭代之后，d值将不会改变。 因此，在第 (m+1) 次迭代中，d 值不会改变。 但是，我们事先并不知道确切的 m 值，我们无法使算法精确迭代 m 次然后终止。 如果我们试图让算法在每个 d 值不再改变时停止，那么它将在 m + 1 次迭代后停止。

3. Floyd-Warshall 算法的空间需求为 $\Theta(n^3)$，因为要计算 $d_{ij}^{(k)}$，其中 $i,j,k=1,2,\ldots,n$。请证明下面所列出的去掉所有上标的算法是正确的，从而将 Floyd-Warshall 算法的空间需求降低到 $\Theta(n^2)$。



Let $d_{ij}^{(k)}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set $\{1,2,3,\ldots,k\}$

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k=0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 \end{cases}$$

The final answer $d_{ij}^{(n)} = \delta(i,j)$ for all $i,j \in V$

```
FLOYD-WARSHALL(W)
1: n = W.rows
2: D^(0) = W
3: for k = 1 to n do
4:     let D^(k) = d_ij^(k) be a new n × n matrix
5:     for i = 1to n do
6:         for j = 1to n do
7:             d_ij^(k) = min( d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1) )
8: return D^n
```

Because each execution of line 7 takes $O(1)$ time, the algorithm runs in time $\Theta\left(n^3\right)$.

hw9

2. 假定在一个权重函数为 W 的有向图图 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)，$\hat{w}(u,v)=0$

Johnsons algorithm uses the technique of **reweighting**:

If all edge weights w in a graph $G=(V,E)$ are nonnegative, we can find shortest paths between all pairs of vertices by running Dijkstras algorithm once from each vertex

Note:

- 题面中“权重为0”指旧权重，直观地新权重也为0，但需要说明
- 环路的权重指环路上各边的权重和
- Given a weighted, directed graph $G=(V,E)$ with weight function $w: E \to \mathbb{R}$, let $h: V \to \mathbb{R}$ be any function mapping vertices to real numbers. For each edge $(u,v) \in E$,define

  $$\hat{w}(u,v)=w(u,v)+h(u)-h(v)$$
- 新权重满足非负性

3. (最大流的更新) 设 $G=(V,E)$ 是一个源结点为 s 汇结点为 t 的流网络，其容量全部为整数值。假定我们已经给定 G 的一个最大流。

a. 如果将单条边 $(u,v)\in E$ 的容量增加 1 个单位，请给出一个 $O(V+E)$ 时间的算法来对最大流进行更新。

b. 如果将单条边 $(u,v)\in E$ 的容量减少 1 个单位，请给出一个 $O(V+E)$ 时间的算法来对最大流进行更新。

Note:

- 回忆基本的FF算法

- FORD-FULKERSON-METHOD(G, s, t)
  1: initialize flow f to 0
  2: **while** there exists an augmenting path p in the residual network G_f
  3: augment flow f along p
  4: return f
- 最大流可能不能增加，因此残差网络中将不存在增广路径。否则
- 如果确实穿过了一个最小割，我们可以将流量增加 1，并执行一次 Ford-Fulkerson 迭代。如果存在增广路径，则会在本次迭代中找到并增加它。由于边容量是整数，因此流量值都是整数。由于流量严格增加，并且每次增加一个整数，Ford-Fulkerson 的第 3 行的 while 循环的单次迭代将使流量增加 1，我们知道这将是最大的流。其中我们使用 BFS找增广路径，时间复杂度为O(V + E)

hw10

1. 试说明如何扩展 Rabin-Karp 算法用于处理以下问题：在一个 $n\times n$ 的二维字符数组中搜索一个给定的 $m\times m$ 的模式。(该模式可以在水平方向和垂直方向移动，但是不可以旋转。)

Basic Idea of Rabin-Karp Algorithm

A string search algorithm which compares a string's hash values, rather than the strings themselves. For efficiency, the hash value of the next position in the text is easily computed from the hash value of the current position.

- If the hash values are unequal, the algorithm will calculate the hash value for next M-character sequence.
- If the hash values are equal, the algorithm will compare the pattern and the M-character sequence.
- In this way, there is only one comparison per text subsequence, and character matching is only needed when hash values match.

RK(P, T, d, q)

$n = T.length$, $m = P.length$, $h = d^{m-1} \bmod q$;
$p = 0$, $t_0 = 0$;
for $i = 1$ to m **do** //pre-processing
  $p = ((p \cdot d) + ord(P[i])) \bmod q$ // $hash(P[1..m])$
  $t_0 = ((t_0 \cdot d) + ord(T[i])) \bmod q$ // $hash(T[1..m])$
for $s = 0$ to $n-m$ **do** // matching, $(n-m+1)$ times
  if $p == t$ && $P[1..m] == T[s+1..s+m]$ **then** // $\Theta(m)$
    print "Pattern occurs with shift" s
  if $s < n-m$ **then** // compute t_{s+1} based on t_s
    $t_{s+1} = (t_s - ord(T[s+1]) \cdot h) \cdot d + ord(T[s+m+1])) \bmod q$

像一维的Rabin-Karp一样计算每一行的哈希值，然后将每一行的哈希值作为字符，再次进行哈希处理。

```
RK2(P,T)
    for i=1 to m do
        //i-th line, j-th column
        for j=1 to m do
```

```
            //hash(P[i][1..m])
            p[i]=((p*d)+ord(P[i][j]))mod q
            //hash(T[i][1..m])
            t[i][0]=((p*d)+ord(T[i][j]))mod q
    for ls=0 to n-m do
        for k=1 to m do
            //hash(p[1..m])
            p=((p*d)+ord(p[k]))mod q
            //hash(t[1..m][0])
            newt[0+ls]=((newt[0+ls]*d)+ord(t[k][0]))mod q
        for cs=0 to n-m do
            if p==newt[0+ls]&&  &&
                //all equals
                print"pattern occurs (ls,cs)"
            if cs< n - m then
                //in line shift ls, compute new hash'
                for i=1 to m
                t[i+ls][cs+1]=t[i+ls][cs]-ord(T[i+ls][cs+1]*h*d)+ord(T[i+ls]
[cs+m+1]))mod q
                for k=1 to m do
                //hash(t[1..m][0])
                newt[0+ls]=((newt[0+ls]*d)+ord(t[k][0]))mod q
        if ls< n - m then
            //
```

2. 对字母表 $\Sigma = \{a,b\}$，画出与模式 ababbabbababbababbabb 对应的字符串匹配自动机的状态转换图

0	1	0
1	1	2
2	3	0
3	1	4
4	3	5
5	6	0
6	1	7
7	3	8
8	9	0
9	1	10
10	11	0
11	1	12
12	3	13
13	14	0
14	1	15
15	16	8
16	1	17
17	3	18
18	19	0
19	1	20
20	3	21
21	9	0