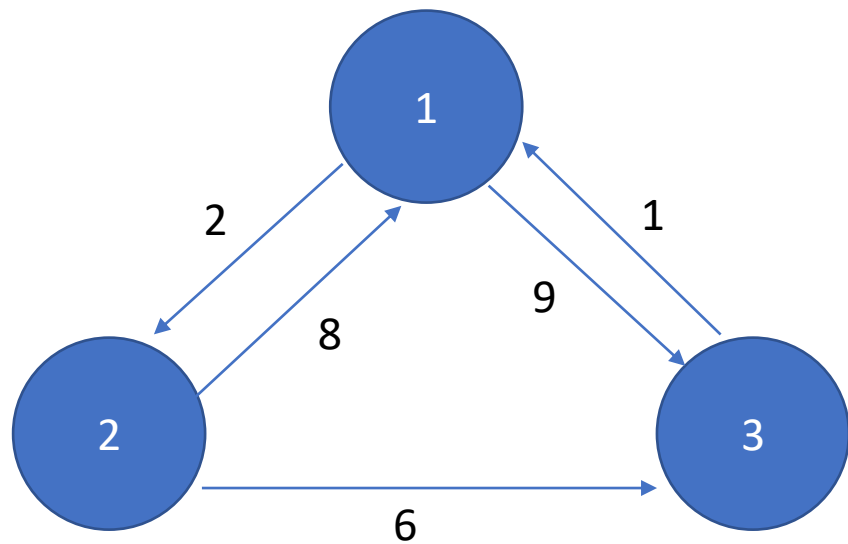


所有点对的最短路径问题

$G = (V, E)$ 是一个有向图，其中每条边 (i, j) 有非负长度 $l(i, j)$ ，如果顶点 i 到顶点 j 没有边，则 $l(i, j) = \infty$ 。假设 $V = \{1, 2, \dots, n\}$, $i \neq j$ ，定义 d_{ij}^k 是从 i 到 j ，并且不经过 $\{k + 1, k + 2, \dots, n\}$ 中任何顶点的最短路径长度，则可递归计算

$$d_{ij}^k = \begin{cases} l(i, j) & \text{if } k = 0 \\ \min\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\} & \text{if } 1 \leq k \leq n \end{cases}$$

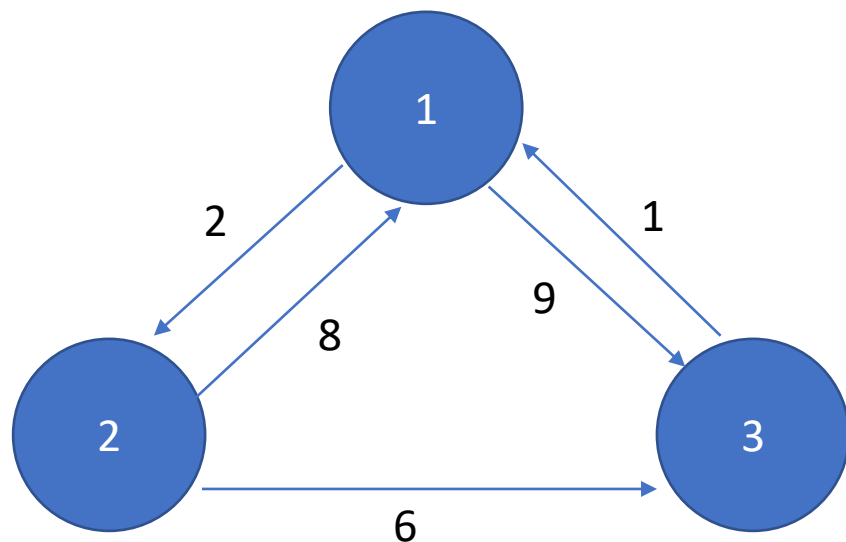


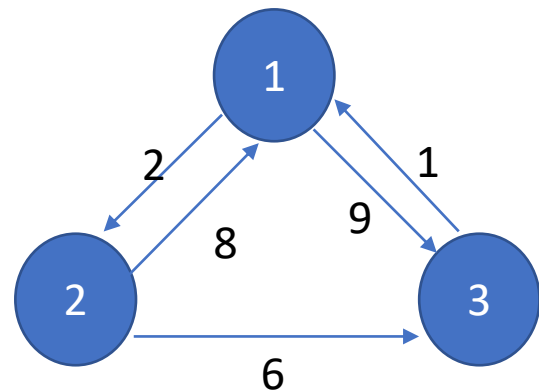
1. 根据下面的有向图，用Floyd算法求所有点对的距离。

Floyd算法

如果 $i \neq j$ 并且 (i, j) 是 G 中的边，则置 $D_0[i, j] = l(i, j)$ ；否则置 $D_0[i, j] = \infty$ 。然后执行 n 次迭代，在第 k 次迭中，

$$D_k[i, j] = \min\{D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j]\}.$$





Floyd算法

如果 $i \neq j$ 并且 (i, j) 是 G 中的边, 则置 $D_0[i, j] = l(i, j)$; 否则置 $D_0[i, j] = \infty$ 。然后执行 n 次迭代, 在第 k 次迭中,

$$D_k[i, j] = \min\{D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j]\}.$$

D_0	1	2	3
1	0	2	9
2	8	0	6
3	1	∞	0

初始状态

D_2	1	2	3
1	0	2	8
2	8	0	6
3	1	3	0

$D[1,3] < D[1,2] + D[2,3] = 8$, 更新 $D[1,3]$

D_1	1	2	3
1	0	2	9
2	8	0	6
3	1	3	0

$D[3,2] < D[3,1] + D[1,2] = 3$, 更新 $D[3,2]$

D_3	1	2	3
1	0	2	8
2	7	0	6
3	1	3	0

$D[2,1] < D[2,3] + D[3,1] = 7$, 更新 $D[2,1]$

2. 有容量为9的背包，要装入4种体积为2,3,4和5的物品，它们的价值分别为3,4,5和7。在不超出背包容量的前提下，尽可能多地在背包内装入物品，使总价值最大。

KNAPSACK算法

输入：物品集合 $U=\{u_1, u_2, \dots, u_n\}$ ，体积分别为 s_1, s_2, \dots, s_n ，价值分别为 v_1, v_2, \dots, v_n ，容量为 C 的背包。

输出： $\sum_{u_i \in S} v_i$ 的最大总价值，且满足 $\sum_{u_i \in S} s_i \leq C$ ，其中 $S \subseteq U$

1. for $i \leftarrow 0$ to n
2. $V[i, 0] \leftarrow 0$
3. for $j \leftarrow 0$ to C
4. $V[0, j] \leftarrow 0$
5. for $i \leftarrow 1$ to n
6. for $j \leftarrow 1$ to C
7. $V[i, j] \leftarrow V[i-1, j]$
8. if $s_i \leq j$ then $V[i, j] \leftarrow \max\{V[i-1, j], V[i-1, j-s_i]+v_i\}$
9. return $V[n, C]$

$v_1=3, v_2=4, v_3=5, v_4=7$
 $s_1=2, s_2=3, s_3=4, s_4=5$
 $C = 9$

if $s_i \leq j$ then $V[i, j] \leftarrow \max\{V[i-1, j], V[i-1, j-s_i]+v_i\}$

使用动态规划，求 $V[4,9]$ 。

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0									
2	0									
3	0									
4	0									

初始状态

$v_1=3, v_2=4, v_3=5, v_4=7$
 $s_1=2, s_2=3, s_3=4, s_4=5$
 $C = 9$

if $s_i \leq j$ then $V[i, j] \leftarrow \max\{V[i-1, j], V[i-1, j-s_i]+v_i\}$

使用动态规划，求 $V[4,9]$ 。

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0									
3	0									
4	0									

只考虑第一件物品，容量大于等于2时放入。

$v_1=3, v_2=4, v_3=5, v_4=7$
 $s_1=2, s_2=3, s_3=4, s_4=5$
 $C = 9$

if $s_i \leq j$ then $V[i, j] \leftarrow \max\{V[i-1, j], V[i-1, j-s_i]+v_i\}$

使用动态规划，求 $V[4,9]$ 。

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0									
4	0									

考虑第一二件物品。容量为2时，放入第一件物品；容量为3或4时，放入第二件；容量大于4时，放入第一、二件物品。

$v_1=3, v_2=4, v_3=5, v_4=7$
 $s_1=2, s_2=3, s_3=4, s_4=5$
 $C=9$

if $s_i \leq j$ then $V[i, j] \leftarrow \max\{V[i-1, j], V[i-1, j-s_i]+v_i\}$

使用动态规划，求 $V[4,9]$ 。

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	5	7	8	9	9	12
4	0	0	3	4	5	7	8	10	11	12

最大价值为12。