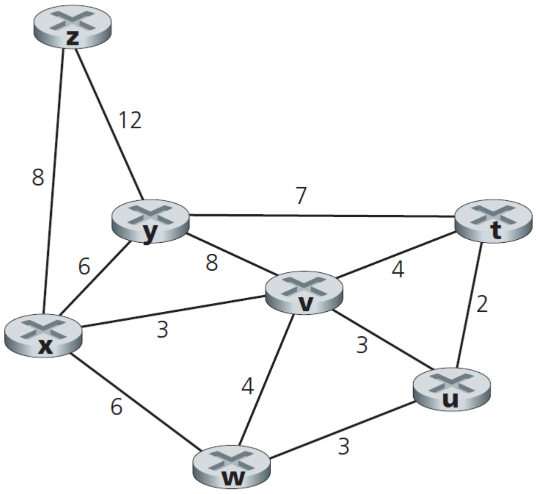
hw5

P3, P5, P11

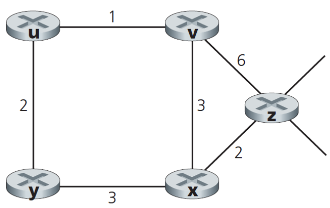
P3 Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from x to all network nodes. Show how the algorithm works by computing a table similar to Table 5.1.



Step	N'	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z)
0	x	∞	∞	3,x	6,x	6,x	8,x
1	xv	7,v	6,v		6,x	6,x	8,x
2	xvu	7,v			6,x	6,x	8,x
3	xvuw	7,v				6,x	8,x
4	xvuwy	7,v					8,x
5	xvuwyt						8,x
6	xvuwytz						

P5 Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node z.

（提示：假设采用同步方式计算，给出初始以及之后每一轮的节点 z 的距离表。第 i 轮，节点 z 从邻居收到的距离向量为到目的地最多经过 i 跳的路由，而节点 z 的距离向量为最多经过 i+1 跳的路由。）



考虑 z 的距离表：两个直接邻居 v, x 和 z 自身到所有目的地的路由

初始 v 和 x 的距离向量都为∞，自己的距离向量为距离 z 一跳的路径

目的	u	v	x	y	z
v	∞	∞	∞	∞	∞
x	∞	∞	∞	∞	∞
z	∞	6	2	∞	0

中间步骤略

最后一轮：接下来 v 和 x 传递给 z 的为其最多 2 跳可以到达的路由， z 计算得到的是 z 最多 4 跳可以到达的路由，而且可以看到上一轮和这一轮的计算结果一样，因此算法结束

目的	u	v	x	y	z
v	1	0	3	3	5
x	4	3	0	3	2
z	6	5	2	5	0

P11 Consider Figure 5.7. Suppose there is another router w, connected to router y and z. The costs of all links are given as follows: c(x,y) = 4, c(x,z) = 50, c(y,w) = 1, c(z,w) = 1, c(y,z) = 3. Suppose that poisoned reverse is used in the distance-vector routing algorithm.

a. When the distance vector routing is stabilized, router w, y, and z inform their distances to x to each other. What distance values do they tell each other? 教材翻译有误，路由器 w, y 和 z 之间互相告诉对方其到 x 的距离，请问互相告知的到 x 的距离分别为多少？

b. Now suppose that the link cost between x and y increases to 60. Will there be a count-to-infinity problem even if poisoned reverse is used? Why or why not? If there is a count-to-infinity problem, then how many iterations are needed for the distance-vector routing to reach a stable state again? Justify your answer.

c. How do you modify c(y,z) such that there is no count-to-infinity problem at all if c(y,x) changes from 4 to 60?

（可以采用 ppt 中 P22 页的表格描述算法的步骤，仅仅考虑到 x 的路由）

step	y			z			w		
cost	yx=4->60	3	1	3	zx=50	1	1	1	wx=∞
	y, next-hop	z	w	y	z, next-hop	w	y	z	w, next-hop
稳定	4, x	6	∞	4	6,w	5	4	∞	5,y
t0	9, z	6	∞	4	6,w	5	4	∞	5,y
t1	9,z	6	∞	∞	6,w	5	9	∞	10,y
t2	9,z	6	∞	∞	11,w	10	9	∞	10,y
t3	14,z	11	∞	∞	11,w	10	9	∞	10,y
t4	14,z	11	∞	∞	11,w	10	14	∞	15,y
t5	14,z	11	∞	∞	16,w	15	14	∞	15,y

t6	19,z	16	∞	∞	16,w	15	14	∞	15,y
…									
t24	49,z	46	∞	∞	46,w	45	44	∞	45,y
t25	49,z	46	∞	∞	46,w	45	49	∞	50,y
t26	49,z	46	∞	∞	50,x	50	49	∞	50,y
t27	53,z	50	∞	∞	50,x	50	49	50	50,y
t28	53,z	50	∞	∞	50,x	50	53	50	51,z
t29	52,w	50	51	∞	50,x	∞	53	50	51,z
t30	52,w	50	51	52	50,x	∞	∞	50	51,z

a. 初始 Dy(x)=4, x; Dw(x)=5, y; Dz(x)=6, w
y 发布给 w 和 z 的距离都是 5
w 发布给 y 和 z 的距离分别为∞和 5
z 发布给 y 和 w 的距离风别为 6 和∞

b. t0 时刻 c(y, x)从 4 变为 60 时，y 发现到 x 有两条路径 60 和 6 + 3 = 9，因此 y 了解到通过 z 到达 x 的路径为 9，即距离向量改为 9, z，接下来进行交换

t1: w 了解到通过 y 到 x 的路径为 9 + 1 = 10

t2: z 了解到通过 w 到 x 的路径为 10 + 1 = 11

t3: y 了解到通过 z 到 x 的路径为 11 + 3 = 14

出现了 3 个节点的路由回路： 逆时钟方向 y → z → w→y，即出现了无穷计数问题。

t26：节点 z 发现通过 w 到 x 的路径为 50+1，而 z 直接到 x 的路径为 50，因此选择 z 通过 x 到 x 的路径 50，打破了路由的回路，通知 y 和 w

t27：节点 y 了解到通过 z 到 x 的路径为 53，w 仍然采用老的路径 49，而不是通过 z 到 x 的 50+1 的路径，节点 y 通知 w

t28：现在节点 w 了解到通过 y 到 x 的路径为 53 + 1，而通过 z 到 x 的路径更短，采用通过 z 到 x 的 51 的路径， w 通知 y

t29：y 了解到通过 w 到 x 的路径 51 + 1，比通过 z 到 x 的 53 的路径更好，采用通过 w 到 x 的 52 的路径，y 通知 z

t30：z 收到 y 的通知，通过 y 到 x 的路径为 52+3=55，选择 z 通过 x 到 x 的路径。可以看到距离向量和上一轮没有变化，因此算法稳定

综上所述，从 t0 到 t30 算法稳定，总共经历了 31 轮

c) 没有布置