Homework9

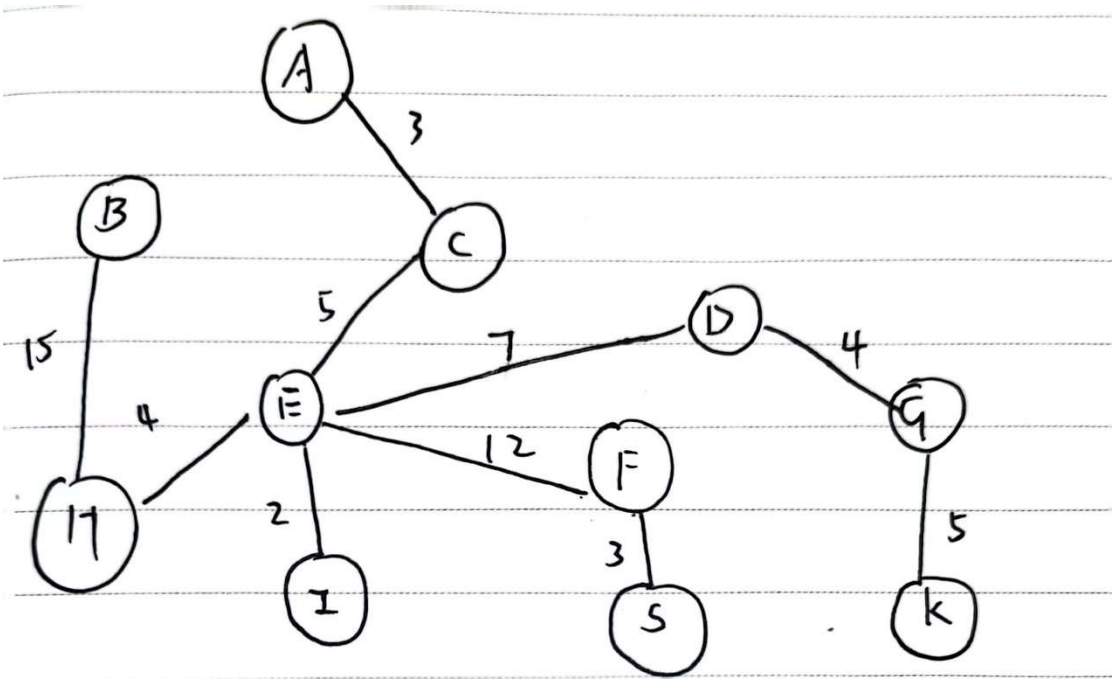
521021910522 郭俊甫

Exercise 1

最短路径树不是最小支撑树。最短路径树是到各个定点的最短路径形成的子树，而最小支撑树是能到达所有定点的路径的总权重最小形成的子树。区别在于，总权重最小不能保证到每个顶点距离最小，利用到某个节点的中间路径到一个节点可能不是到这个节点的最短路径，但可能会使总路径变少。如 A、B、C 三个顶点，A 能直接到 B、C，距离分别为 2、2，B 能到 C，距离为 1。则最短路径树为 A 直接到 B、C，但最小支撑树为 A 先到 B 再到 C。

Exercise 2

1. 记 S 为已知最短路径的节点集，Q 为其余节点集，d[u]为当前 s 点到 u 点的最短路径，w(u, v)为相邻两点 u、v 的路径长度。
   则寻找最短路径的过程如下:
   初始化 S 为空，Q 为全集，d[s]=0，其余 d[u]为无穷大。
   S <-… 表示目前距离最近的一个节点从 Q 进入到 S，后面紧跟着最近距离，若其下一为一系列的距离赋值，表示该节点的松弛操作引起的最短距离更新
   S <- s d[s] = 0
   d[k] = 2 d[f] = 3 d[i] = 20
   S <- k d[k] = 2
   d[g] = 7
   S <- f d[f] = 3
   d[d] = 12 d[e] = 15
   S <- g d[g] = 7
   d[d] = 11 d[c] = 15
   S <- d d[d] = 11
   S <- e d[e] = 15
   d[i] = 17 d[h] = 19 d[b] = 28
   S <- c d[c] = 15
   d[a] = 18
   S <- i d[i] = 17
   S <- a d[a] = 18
   d[b] = 22
   S <- h d[h] = 19
   S <- b d[b] = 22
2. 最短路径树如下:



Exercise 3

S <- s d[s] = 0

d[k] = 2 d[i] = 20

S <- k d[k] = 2

d[g] = 7

S <- g d[g] = 7

d[d] = 11 d[c] = 15 d[f] = 22

S <- d d[d] = 11

d[e] = 18 d[f] = 19

S <- c d[c] = 15

d[a] = 21

S <- e d[e] = 18

d[b] = 31

S <- f d[f] = 19

S <- I d[i] = 20

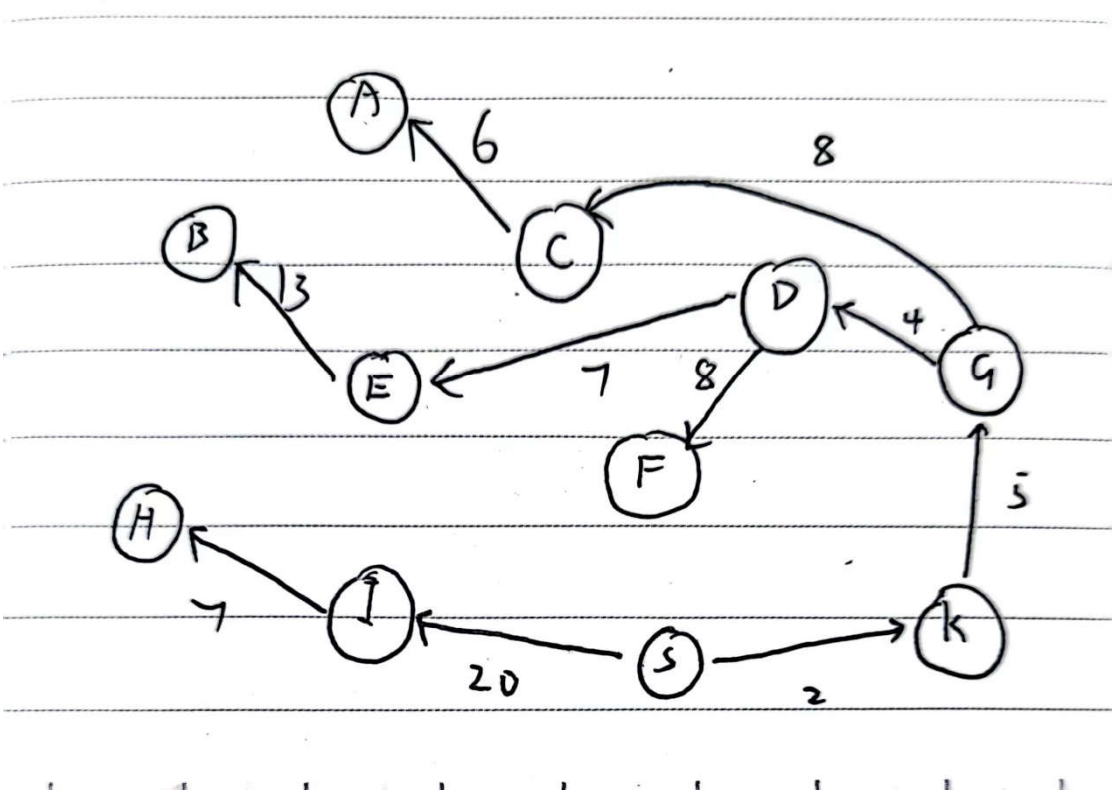
d[h] = 27

S <- a d[a] = 21

S <- h d[h] = 27

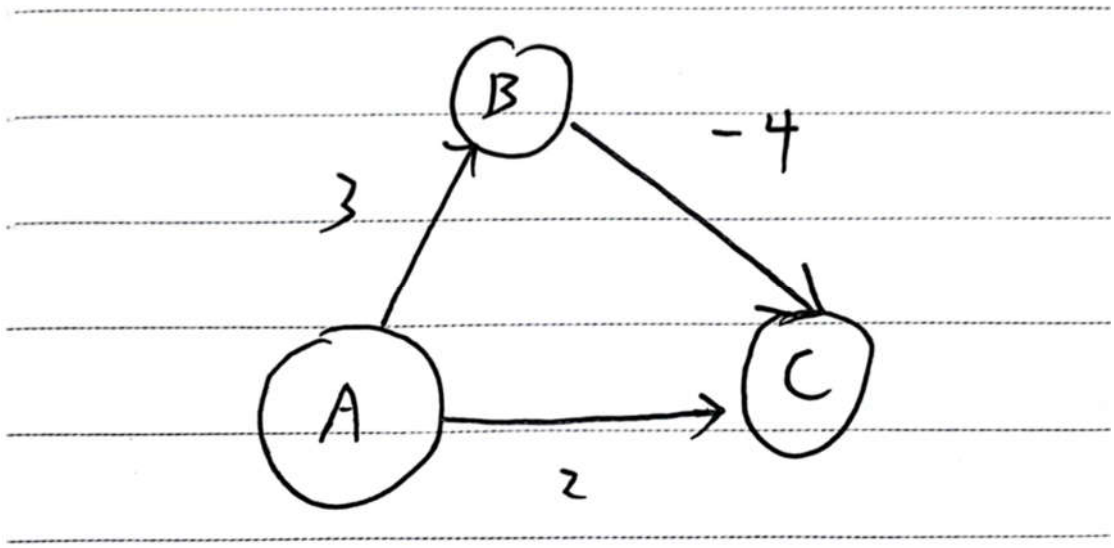
S <- b d[b] = 31

最终结果如下:



Exercise4

不能有负权，因为每次寻找距离最短点将其移入 S 的原理便是:该点距离目前最短，在之后的所有其他路径中的距离均可表示为到其他某个点的距离加上对应点到该点距离，目前距离最短能保证在其他距离加上一个正的距离后不可能比当前路径短，因此该距离为最短距离。若允许负权值存在，则不能保证加上一个负的权值后距离不比目前距离小，算法便会出现错误。举例如下:



按照 Dijkstra 算法，过程为:

S <- a d[a] = 0

d[c] = 2 d[b] = 3
S <- c d[c] = 2
S <- b d[b] = 3
得到 a 到 c 的最短路径为 a->c，距离为 2；但实际最短路径为 a->b->c，距离为-1