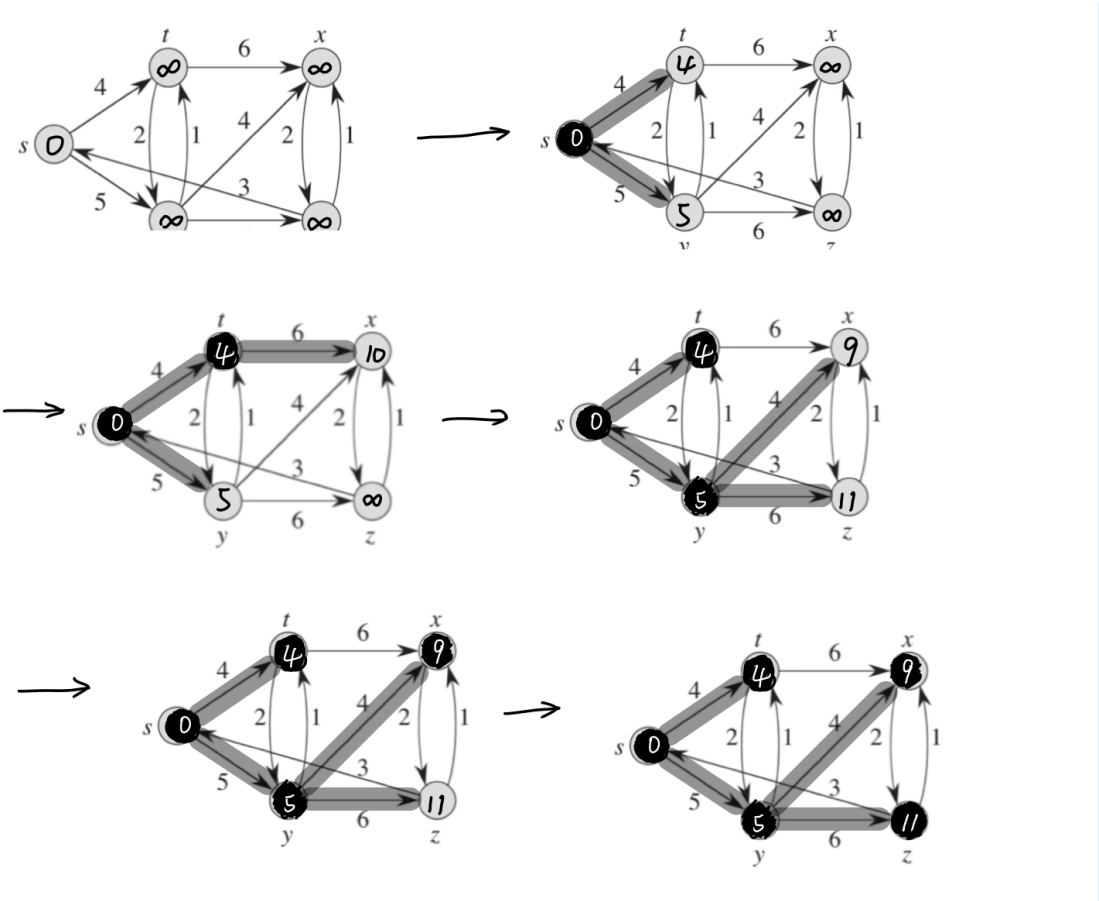
数据结构与算法第十一次作业

201300066 麻超

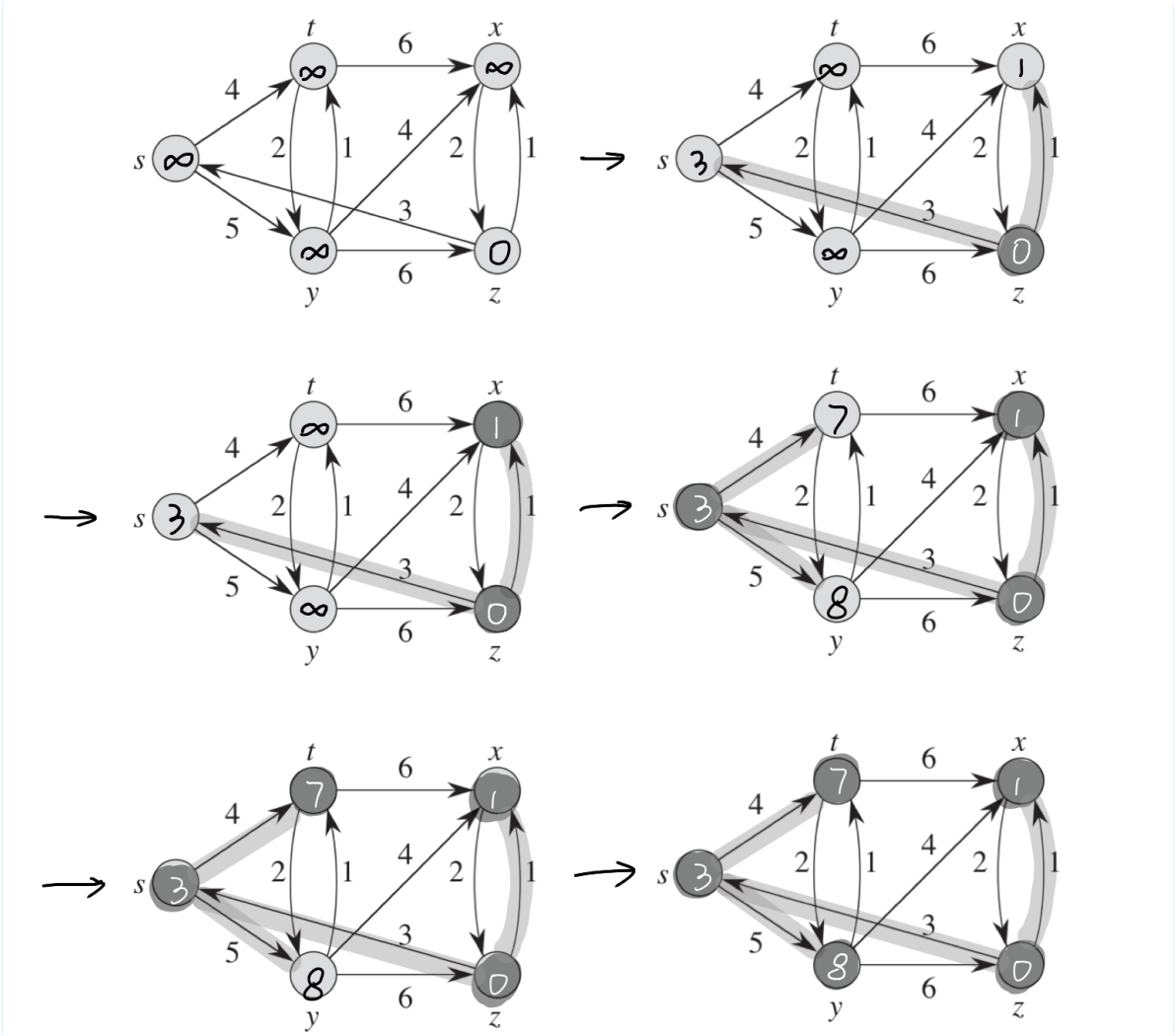
Problem 1

1.a

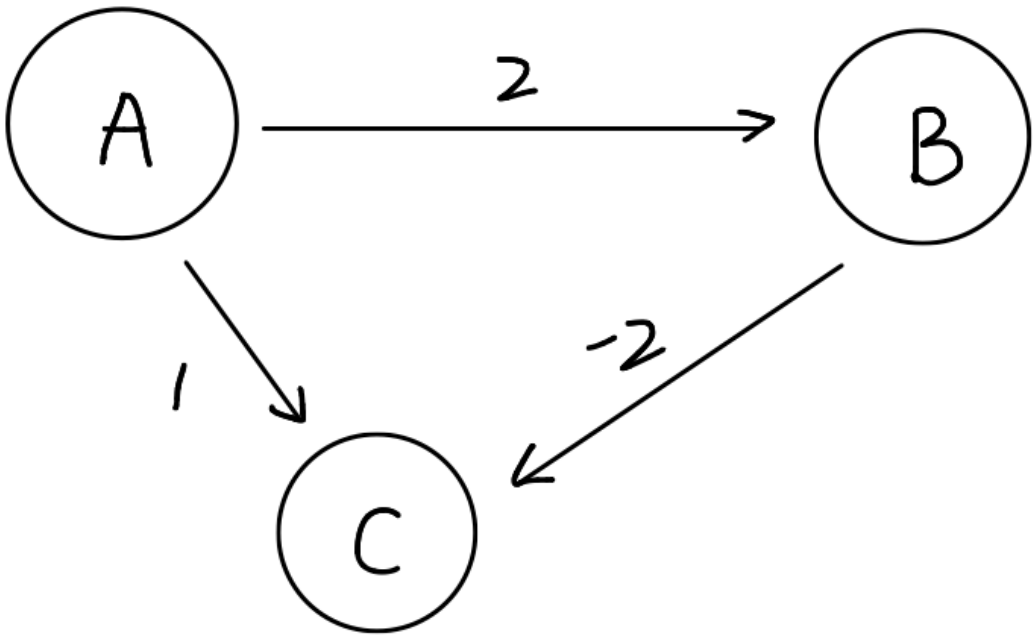
以s为起点：



以z为起点：

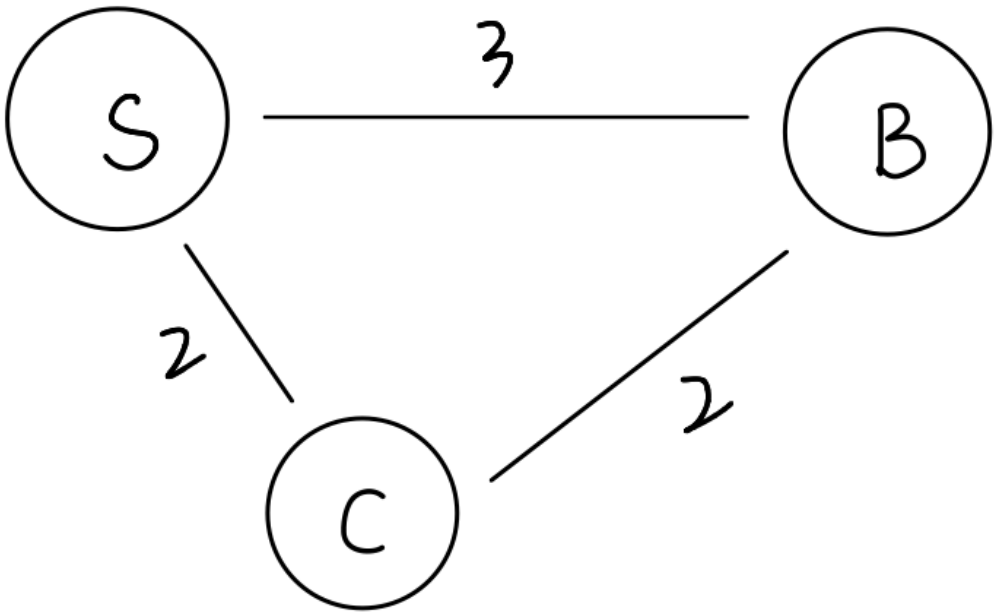


1.b



对该图而言，如果从A点开始，那么首先会标记C点，则A到C点距离为1，但是通过A-B-C得到的距离为0，与答案不同。但在A-B-C的过程中不可以更新C点，因为C是一开始就被标记了的。所以Dijkstra算法在负权重的情况下产生了错误答案。

1.c



如该图所示，其最小生成树应为以C为顶点，由C-S和C-B两条边组成。而以S为根的最小路径树是由S-C和S-B组成的。

Problem 2

一条线路的可靠性由所有分线路可靠性的积决定，所以只需要修改Dijkstra算法使其依据权重之乘积判断而不是和即可，其与Dijkstra算法具有相同的时间复杂度。

Algorithm 1 RE-Dijkstra(G,r,x,y)

```
INITIALIZE-SINGLE-SOURCE(G,x)
S = ∅
while Q ≠ ∅ do
    u =EXTRACT(Q)
    S = S ∪ {u}
    for each vertex v if v ∈ G.Adj[u] do
        if v.d < u.d × r(u,v) then
            v.d = u.d × r(u,v)
            v.π = u
        end if
    end for
end while
while y ≠ x do
    Output y
    y = y.π
end while
Output x
```

时间复杂度与Dijkstra算法相同，均为$O(V \lg V + E)$

Problem 3

Problem 4

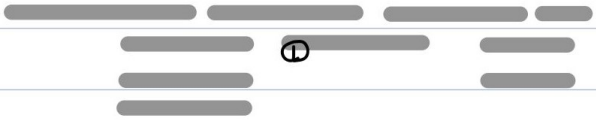
4. a) 不正确. 如图：会选择上面一行，但应选择下面两个.

b) 同上，应选下一行两个，而不是上一行.

c) 正确.

d) 不正确. 应当选择下一行三个活动，但该算法只能选择上一行两个.

e) 不正确. 该算法会因为选择①导致最后只能有3个活动，但第1行有4个.



f) 不正确. 该算法会删去下面一行而保留上面一行，但下一行为最优解.

g) 不正确. ①和②冲突数最为应删去，但最优解是上面一行.

h) 正确.

Problem 5

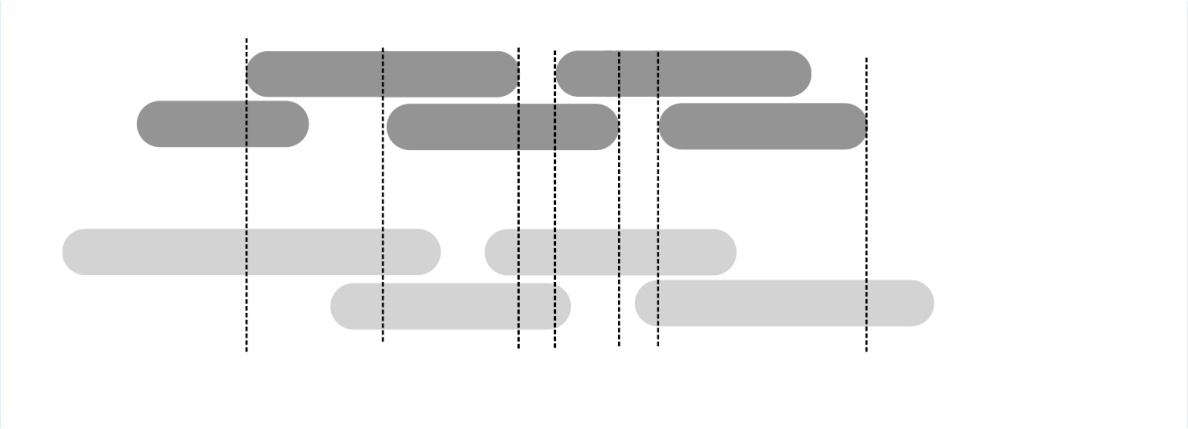
对于该问题，可以利用贪心算法解决。首先将所有给出的区间按照左端点的值进行排序，然后从左往右选，每一次选的时候选择左端点在已经覆盖到的区域之内，右端点尽可能大的区间。定义表示每一个区间的结构体为$rec()$,其中含有两个要素$rec.l$(表示左端点)和$rec.g$表示右端点。

伪代码如下：

```
SMALLEST-COVER(L[1...n],R[1...n]):
    struct rec{
        int l;
        int r;
    };
    Rec[n]=new array
    for i in [1...n]:
        Rec[i]=new rec;
        rec.l=L[i]
        rec.r=R[i]
    End For
    for every rec in Rec[n] do:
        sort by rec.l
    End For
    R=[] #储存结果
    s=Rec[0] #第一个元素
    R.add(s)
    Rec.delete(s)
    now=1
    While Rec[]≠null:
        If Rec[now].l>s.r:
            Return false #无解
        else max=now
            for max in Rec[now...n] #新变量max表示符合条件的区间最大索引值
                if Rec[max].l>s.r:
                    break
                End If
            End For
            for every rec in Rec[now...max]:
                If a have the largest r value:
                    now=a.index #表示其在Rec数组中所处的位置
                    delete all rec if rec.index<now
                    R.add(a)
                End If
            End For
        End If
    End While
```

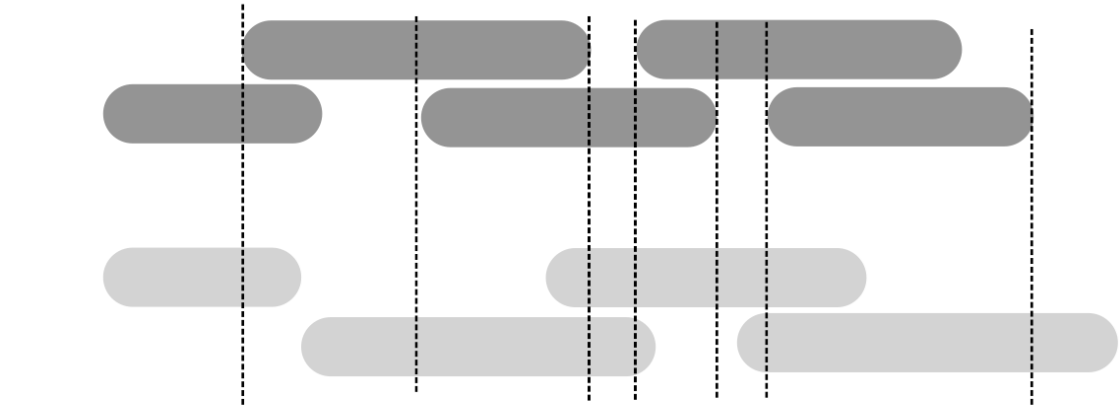
排序所用时间复杂度为$O(n\lg n)$,建立结构体rec的集合用时为$O(n)$,对后面的循环，每个元素会至多被遍历两次，所以总的时间复杂度为$O(n\lg n)$.

对于该问题，假设用贪心算法得到的结论不是最佳结论，则表示必然在某个区间$[left,right]$中，必然有比贪心算法更少的区间数就可以完成(注意此处的$left,right$必须是在$L[1..n]$和$R[1..n]$之中。假设如图所示，灰色的部分比黑色的部分所用的区间数更少。



在这种情况下，根据贪心算法的原理，则在选择时会选择第一个灰色方块，而不是第一个黑色方块，总之，这是不会出现的情况。

如果是下图的情况，那么其在选择时贪心算法同样应考虑第二个灰色的区间块而不是第二个黑色块。所以贪心算法得到的覆盖必定是最小覆盖。



Problem 6

6.a

假设有一组硬币集为{1,10,15}(cents)，现在需要给顾客找零20美分，根据贪心算法，应当先减去15，剩下的5用5个1来代替，共6个硬币。但是实际上最简单的办法是找两个10美分的硬币。

6.b

假设通过a中给出的贪心算法一共用m个硬币就可以解决问题，一个解集为$(x_0, x_1, \ldots, x_m)$，每一个x_i都表示面值为c^i的硬币的数量。那么必然有所有的x_i都小于c，如果x_i大于等于c，那么可以用c个x_i的硬币换取一个面值为x_{i+1}的硬币，这样总硬币的数量就减少了$c-1$个，且总值不变。

现在需要说明利用贪心算法可以得到与其结果相同的一组解。由于贪心算法的性质，其总是选择当前最大的c^i,直到总价值小于c^i时才会考虑c^{i-1}，那么由于$c \cdot c^{i-1} = c^i$,则c^{i-1}的数量不会超过c，即x_{i-1}的数量不会超过c。这就表明贪心算法就是当前的最优解。