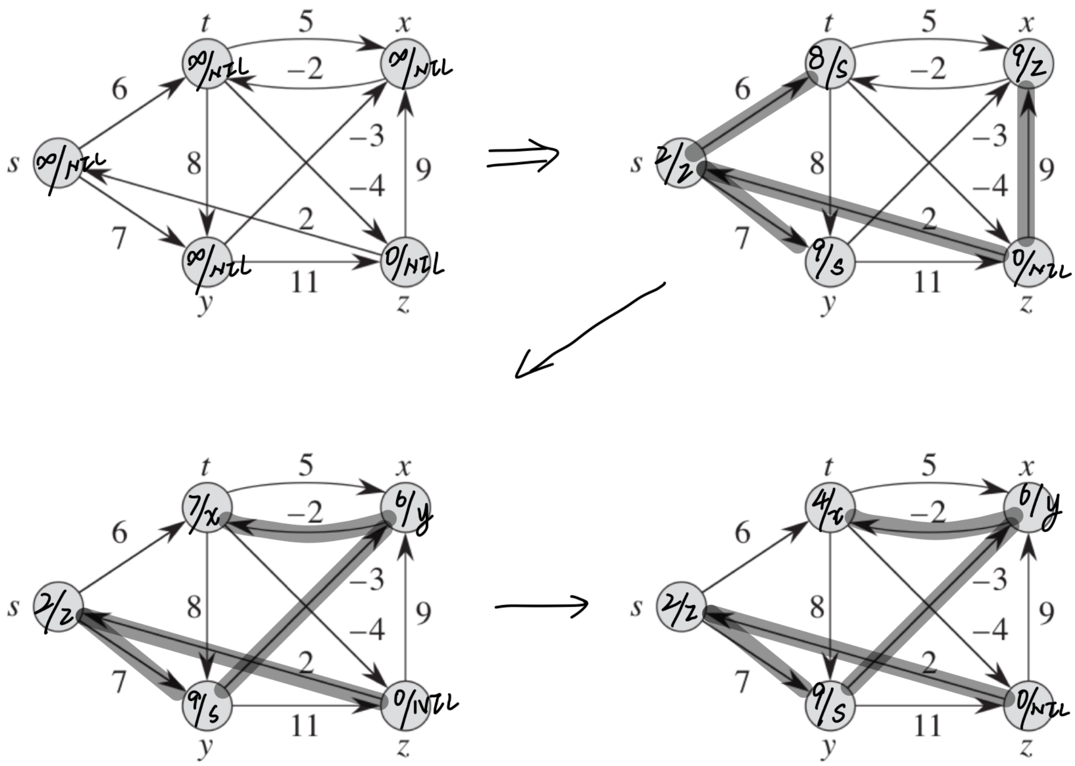
数据结构与算法第十二次作业

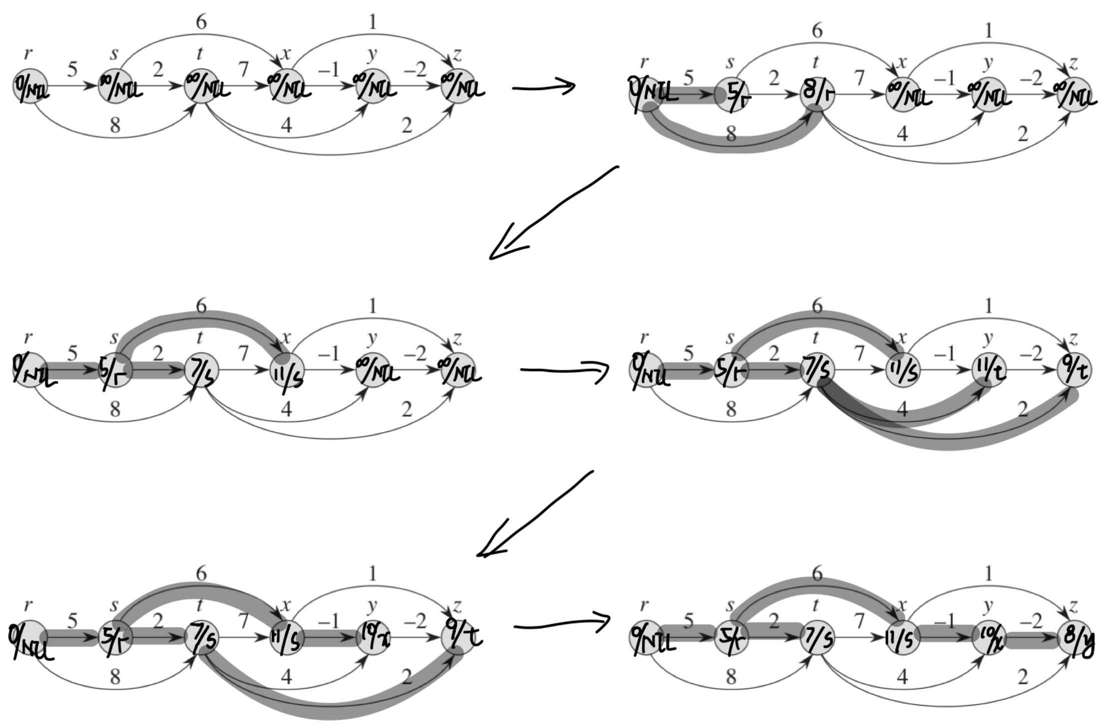
201300066 麻超

Problem 1

a



b



Problem 2

$$D^0 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -3 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 12 & \infty & \infty & 0 \end{bmatrix}, D^1 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -3 & \infty & \infty & 0 & -4 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 12 & \infty & \infty & 0 \end{bmatrix}$$

$$D^2 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & \infty & -8 \\ -3 & \infty & \infty & 0 & -4 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 12 & 7 & \infty & 0 \end{bmatrix} D^3 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & \infty & -8 \\ -3 & \infty & \infty & 0 & -4 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 12 & 7 & \infty & 0 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & -2 & \infty \\ 1 & 2 & 0 & 4 & 0 & -8 \\ -3 & \infty & \infty & 0 & -4 & \infty \\ 6 & 7 & \infty & 9 & 0 & \infty \\ 4 & 5 & 12 & 7 & 3 & 0 \end{bmatrix} D^5 = \begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -1 & 0 & \infty & 2 & -2 & \infty \\ 1 & 2 & 0 & 4 & 0 & -8 \\ -3 & 3 & \infty & 0 & -4 & \infty \\ 6 & 7 & \infty & 9 & 0 & \infty \\ 4 & 5 & 12 & 7 & 3 & 0 \end{bmatrix}$$

$$D^6 = \begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -1 & 0 & \infty & 2 & -2 & \infty \\ -4 & -3 & 0 & -1 & -5 & -8 \\ -3 & 3 & \infty & 0 & -4 & \infty \\ 6 & 7 & \infty & 9 & 0 & \infty \\ 4 & 5 & 12 & 7 & 3 & 0 \end{bmatrix}$$

b

使用Jahnson APSP算法得到如下矩阵：

$$\begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -1 & 0 & \infty & 2 & -2 & \infty \\ -4 & -3 & 0 & -1 & -5 & -8 \\ -3 & 3 & \infty & 0 & -4 & \infty \\ 6 & 7 & \infty & 9 & 0 & \infty \\ 4 & 5 & 12 & 7 & 3 & 0 \end{bmatrix}$$

每个节点的h值如下：

$$h(1) = -4, h(2) = -3, h(3) = 0, h(4) = -1, h(5) = -5, h(6) = -8.$$

边权矩阵如下：

$$\begin{bmatrix} 0 & \infty & \infty & \infty & 0 & \infty \\ 2 & 0 & \infty & 0 & \infty & \infty \\ \infty & 5 & 0 & \infty & \infty & 0 \\ 0 & \infty & \infty & 0 & 7 & \infty \\ \infty & 5 & \infty & \infty & 0 & \infty \\ \infty & 0 & 4 & \infty & \infty & 0 \end{bmatrix}$$

Problem 3

a

要求从s到t的所有路径之和，可以考虑利用动态规划的思想解决，也就是从s到t的所有路径之和相当于所有s的后继结点到t的路径之和，满足最优子结构性质。以此递归解决问题。在每次递归时，最后总是到t点结束或者当该点没有出的路径时结束（此时为0），表示如下：

$$\text{path number}(u) = \begin{cases} 0, & \text{u have no out degree} \\ \sum_{(u \to v) \in E} (\text{path number(v)}), & \text{others} \end{cases}$$

首先对该图利用DFS算法进行拓扑排序，按该顺序逆序遍历整个图，对每个节点维护$u.path_number$的属性，利用上式的关系，递推得到结果即可。

时间复杂度上，DFS算法时间复杂度为$O(|V| + |E|)$,在后面递推过程中，实际遍历了整个图中所有的边和点，故时间复杂度为$O(|V| + |E|)$，所以总的时间复杂度为$O(|V| + |E|)$.

b

最早开始时间问题同样满足最优子结构性质，使用动态规划解决。

首先对DAG图使用DFS算法进行拓扑排序，按顺序遍历，维护每个节点的$u.starttime$的属性，根据递推关系计算，如下：

$$starttime(u) = \begin{cases} 0, & \text{u have no degree} \\ max_{(v \to u \in E)} \{v.weight + starttime(v)\}, & \text{others.} \end{cases}$$

DFS算法时间复杂度为$O(|V| + |E|)$,该算法需要遍历所有的点和边，故总的时间复杂度同为$O(|V| + |E|)$.

c

不影响工期的前提下最晚开工时间同样满足最优子结构性质，使用动态规划解决。

Problem 4

首先将所有除s之外的点的dist值都设置为∞,再根据后续情况来判断。

因为该图中可能有环的存在，所以在利用Bellman-Ford算法时可能会遇到跳在一个循环里，导致死循环的情况，考虑加一个限制条件，我的想法是考虑一个已访问的列表，在运行算法时如果碰到了之前已访问的节点则说明该图中存在环（这个已访问节点必然是第一个），删除这一条边（若访问$a \to b$时发现b是已访问节点就删除$a \to b$这条边，并从a开始再次访问后续的节点，将所有后续的节点的dist值都设置为$-\infty$.）

Bellman-Ford算法时间复杂度为$O(VE)$,后面的分析过程时间复杂度为$O(V+E)$，所以总的时间复杂度为$O(VE)$.

Problem 5

a

当有一个负权环时，根据$w_{ij} = r \cdot c_{ij} - p_j$，求和得到$\sum_{(i,j)\in C} w_{ij} = r \cdot \sum_{(i,j)\in C} c_{ij} - \sum_{(i,j)\in C} p_j$,易得$\sum_{(i,j)\in C} w_{ij} < 0$，所以$r < \frac{\sum_{(i,j)\in C} p_j}{\sum_{(i,j)\in C} c_{ij}}$.

令$\frac{\sum_{(i,j)\in C} p_j}{\sum_{(i,j)\in C} c_{ij}} = a$,由于存在一个负权重的环，所以$a$是一个可行的$r$值，又因为$a < r^*$,即$r < r^*$.

b

类似于上一问，可以得到$\sum_{(i,j)\in C} w_{ij} > 0$，所以$r > \frac{\sum_{(i,j)\in C} p_j}{\sum_{(i,j)\in C} c_{ij}}$.由于所有的环都是正权重的，所以对于所有的环路，都有$r > a$成立，也就是说$r > max a = r^*$.结论成立。

c

通过a和b可以确定二分查找的策略，如果找到一个权重为负的环，说明估计值比r^*小，否则说明估计值比r^*大。

Problem 6

a

由于需要最少的到达车站次数，所以要让每次更换电池都能走尽可能多的路程。从旧金山开始，寻找每100km以内最远的一个车站，到达上面找到的车站之后，利用相同的原理再次找到下一个车站，以此类推，直到到达目的地。

该算法是线性的，故时间复杂度为$O(n)$.

b

假设有如下两个数组：

$$D[1] = 0, D[2] = 20, D[3] = 90$$
$$C[1] = 10, C[2] = 10, C[3] = 1000$$

总的路程是120英里。利用a中的贪心算法，需要在车站1和3更换电池，总成本为1010，但实际上在车站1和2更换电池也可以到达目的地，且成本为20，更低。

c

该问题满足最小子结构性质，可以利用动态规划的思想解决问题。

首先构建一个有向无环图$G=(V,E)$,其中v表示车站,e表示某两个车站之间有道路可以到达。

令$cost(i)$表示从第i个车站到纽约的最小花费，则可以得到

$$cost[i]=C[i]+min\{cost[j]+C[j]\}$$

根据该式得到递推关系，当到达纽约时停止。

时间复杂度为$O(|V|+|E|)$.

Problem 7

统计Partition的数量具有最优子结构的性质，可以使用动态规划的思想解决。

设$num(i)$表示从第i个字符开始，后面的字符串一共有多少个Partition的数目，对该字符串从后往前遍历，对于遍历过的每个字符都考虑其与之后的字符是否可以组成单词，记录每个$num(i)$的值。

时间复杂度为$O(n^2)$.

$$num(i)=\begin{cases}1,\text{if IsWord(A[n]=true)} & i=n\\ \sum_{j=i}^{n}1(\text{if IsWord(A[i,...,j]=true)} * num(j+1)) & ,i<n\end{cases}$$

伪代码:

```
Partition(A[],n):
    re[1,...,n]=0
    if isWord[A[n]]:
        re[n]=1
    for i=n-1 downto 1:
        sum=0
        for j=1 to n:
            sum=sum+IsWord(A[i,...j])*Partition[j+1]
        re[i]=sum
    return re[1]
```

时间复杂度由调用$IsWord$函数的次数而决定，若$IsWord$的时间复杂度为$O(1)$，则总时间复杂度为$O(n^2)$.