潘苑之 18124004S

Problem Set 11

Problem 1: (a) 设有矩阵 $D^{(i)}_{6\times 6}$，其中 $D^{(i)}[m,n]$ 表示第 i loop 后的 $dist(x_m, x_n)$。

9

$$D^{(0)}=\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix} \quad D^{(1)}=\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix}$$

$$D^{(2)}=\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{bmatrix} \quad D^{(3)}=\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{bmatrix}$$

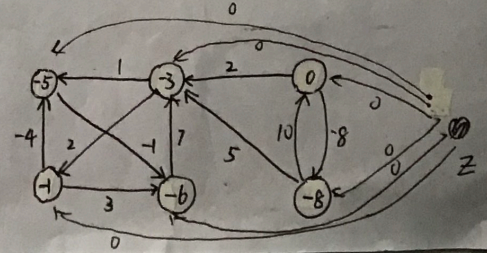
$$D^{(4)}=\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix} \quad D^{(5)}=\begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

$$D^{(6)}=\begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

(b) 依旧用上述矩阵表示法，设 Floyd-Warshall alg. 结束后的矩阵为 $D^{(n)}$，其转置为 $D^{(n)T}$。令 $D'=D^{(n)}+D^{(n)T}$，如果 D' 中存在负元素 $\Rightarrow$ 存在 negative-weight cycle。

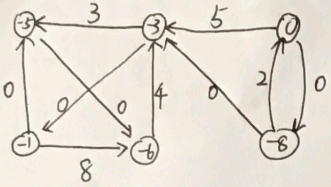
db better

Problem 2: (a)



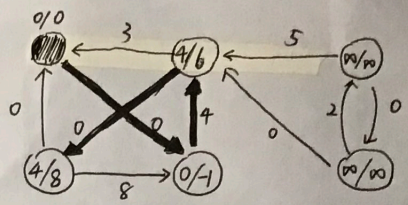
10

每个结点 V 上标记的是 h(v)

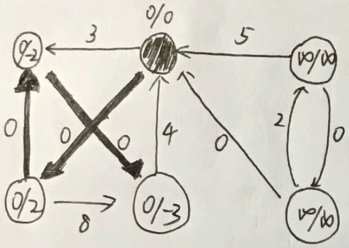


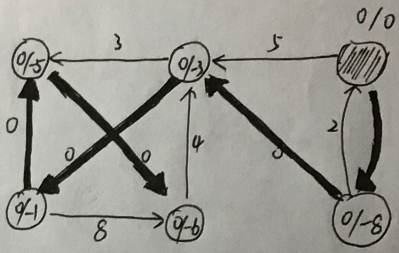
重新赋予权重后的图

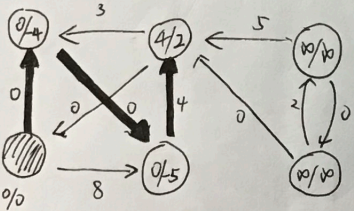
⇒

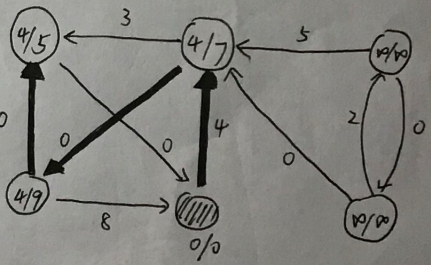


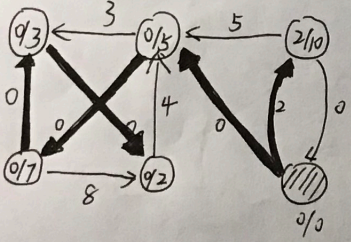
加黑节点为源节点 u，每个节点里标注的是 $\delta(u,v)$ 和 $S(u,v)$。加粗的边是 shortest path 里的边。











(b) Suppose G contains a 0-weight cycle $u_1 \to u_2 \to \cdots \to u_n \to u_1$。由题意，$w(u_1,u_2)+\cdots+w(u_n,u_1)=0$

$\Rightarrow \hat{w}(u_1,u_2)+\cdots+\hat{w}(u_n,u_1) = h(u_1)+w(u_1,u_2)-h(u_2)+\cdots+h(u_n)+w(u_n,u_1)-h(u_1) = w(u_1,u_2)+\cdots+w(u_n,u_1)=0$

又由于 $\hat{w}(u_1,u_2)$、…、$\hat{w}(u_n,u_1)$ 均不小于 0 $\Rightarrow$ $\hat{w}(u_1,u_2)$、…、$\hat{w}(u_n,u_1)$ 均为 0

$\Rightarrow \hat{w}(u,v)=0$ for every edge (u,v) in C

Problem 3:

(a) 假设 boolean matrix 为 T，新 insert 的边为 (m,n) (m,n为结点编号)，则显然对于∀可达 m 的节点，它们也可都达 n 可达的节点。具体到 T 中：

```
for i=1 to |V|:
    if T[i,m]==1:
        for j=1 to |V|:
            T[i,j] = T[i,j] ∨ T[n,j]
```

(b) 假设图G有2个SCC，每个都是包含 $\frac{|V|}{2}$ 的 node，新加的边 e(u,v) 连接这两个SCC。于是，由 part (a) 的 alg，原始的 bool matrix 有 $\frac{|V|}{2}$ 行需要更新，一共需要参与计算的节点数为 $\frac{|V|}{2}\cdot|V| = \frac{|V|^2}{2}$，每次计算 takes constant time ⇒ 至少需要 $\frac{|V|^2}{2}$ 时间 ⇒ $\Omega(|V|^2)$

(c) 对于 insertion sequence 的每一次 insert，只需对 part (a) 的 alg. 做出一点 modification。

For each node u, we matain two sets S_1 and S_2. The former set stores the nodes which can reach u and the latter stores the nodes which u can reach.

①

```
for i=1 to |V|:
    if T[i,u]==1:
        S1.add(i)
    if T[u,i]==1
        S2.add(i)
```

② 于是之后每一次 insert 时，不妨设 insert 的 edge 为 (m,n)，只需

```
for u in m.S1:
    u.S2.add(n)
for v in n.S2:
    v.S1.add(m)
```

③ 更新 transitive closure 操作为：

```
for u in m.S1:
    for v in n.S2:
        T[u,v]=1
```

高线算法，题目要在线

Proof: T中共有 $|V|^2$ 个元素，每个元素至多被重复 update |V| 次

⇒ total time is $O(|V|^3)$

④ 如果 T 中所有元素均为 1，则之后所有 insert 无需更新 T。

Proof: ~~Time for build S_1, S_2 for all nodes: $|V|^2$~~

~~Time for update S_1 and S_2: $x|V|$~~

~~Time for update transitive~~

Problem 4:

(a) 图的节点为 $\{V_{ij}\}$, where $1 \le i \le j \le n$, 表示求解 matrix-chain $A_i A_{i+1} \cdots A_j$ 的最小开销。

图的边为 $\{(V_{ik}, V_{ij}), (V_{k+1,j}, V_{ij})\}$, where $i \le k \le j-1$, 表示求解 $A_i A_{i+1} \cdots A_j$ 的 min_cost 需要求解 $A_i \cdots A_k$, $A_{k+1} \cdots A_j$ 的 min-cost 这两个 sub-problem。最终目标即求解 V_{1n}.

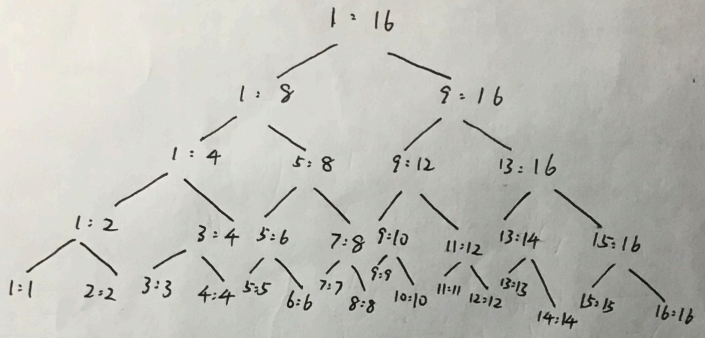
number of vertices: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

number of edges: $2\sum_{i=1}^{n}\sum_{j=i}^{n}(j-1-i+1) = 2\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i) = \sum_{i=1}^{n}(i^2-i) = \frac{(n-1)n(n+1)}{3}$

which edges:

$\{(V_{ik}, V_{ij}), (V_{k+1,j}, V_{ij})\}$, $i \le k \le j-1$

(b)



⇒ 因为没有 recursion tree 中没有重复的子问题

⇒ memoization fails to speed up

(c) Suppose optimal trade sequence 为 $i, i+1, \cdots, j-1, j$, we start with d units of i.

⇒ 收益: $dr_{i,i+1} \cdots r_{j-1,j} - C_{j-i}$ ---- ①

考虑其 substructure, 不妨设 trade sequence 为 $i, i+1, \cdots, j-1$, 对该 subproblem 收益为:

$dr_{i,i+1} \cdots r_{j-2,j-1} - C_{j-i-1}$

⇒ 最终收益: $r_{j-1,j}(dr_{i,i+1} \cdots r_{j-2,j-1} - C_{j-i-1}) - (C_{j-i} - C_{j-i-1})$

$= dr_{i,i+1} \cdots r_{j-1,j} - C_{j-i} + (1 - r_{j-1,j})C_{j-i-1}$ --- ②

由于 C_{j-i-1} 为 arbitary value ⇒ ①、② 未必相等 ⇒ does not necessarily exhibit optimal substrure

Problem 5:

(a) For example, given coin values: $\{1, 3, 4, 5\}$ and the money needed for change is 7.

solution of the greedy alg. $\{5, 1, 1\}$

the optimal solution: $\{4, 3\}$

(b) [greedy-choice] 假设 target amount 是 T, b^k 是可行的 largest denomination, i.e. $b^k \le T < b^{k+1}$, 则 b^k 一定在 optimal change 中.

证：假设 b^k 不在 optimal change 中，当前的 optimal change 是 $m_0b^0+m_1b^1+\cdots+m_{k-1}b^{k-1}$，where $m_0, m_1, \cdots, m_{k-1}$ 是 non-negative 的整数且不全为 0，表示对应 coin 的个数。显然也满足 $m_0b^0+m_1b^1+\cdots+m_{k-1}b^{k-1} \le T$.

1° if $b^k \ge m_0b^0+m_1b^1+\cdots+m_{k-1}b^{k-1}$

⇒ we can substitute the optimal change $m_0b^0+\cdots+m_{k-1}b^{k-1}$ with b^k

⇒ $b^k \le T$ 且 the number of coins can't be more

2° if $b^k < m_0b^0+m_1b^1+\cdots+m_{k-1}b^{k-1}$

⇒ 必存在 $0 \le i \le k-1$, s.t. $b_k = nb^i + m_{i+1}b^{i+1}+\cdots+m_{k-1}b^{k-1}$, where $0<n\le m_i$ 且 n 为整数。

⇒ 可将 optimal change $m_0b^0+m_1b^1+\cdots+m_{k-1}b^{k-1}$ 中的 $nb^i+m_{i+1}b^{i+1}+\cdots+m_{k-1}b^{k-1}$ 替换为 b^k，

⇒ 结果为 $m_0b^0+m_1b^1+\cdots+(m_i-n)b^i+b^k = m_0b^0+m_1'b^1+\cdots+m_{k-1}b^{k-1} \le T$

且 number of coins can't be more

[optimal substructure] 设 target amount 是 T，b^k 是当前可以选的 largest denomination，则

$OPT_{T-b^k}(b^0, b^1, \cdots, b^k) \cup \{b^k\}$ is an optimal solution of the problem.

证：假设 optimal substructure does not hold

⇒ $OPT_T(b^0, b^1, \cdots, b^k)$ gives $|SOL_{T-b^k}(b^0, b^1, \cdots, b^k)| < |OPT_{T-b^k}(b^0, b^1, \cdots, b^k)|$

⇒ ~~contradict~~ contradicts the optimality of $OPT_{T-b^k}(b^0, b^1, \cdots, b^k)$.

(c) ~~Suppose the target amount is T, the smallest~~

Suppose the smallest number of coins needed to make T cents is M_T.

有递推关系：

$$M_T = \begin{cases} 0, & T=0 \\ \min\limits_{c[i]\le T}\{M_{T-c[i]}+1\} \end{cases}$$

The alg. is:

```
M[0] = 0
for t = 1 to T:
    M[t] = ∞
    for i = 1 to n:
        if c[i] ≤ t:
            M[t] = min(M[t], M[t-c[i]] + 1)
return M[T]
```

Time complexity: there are 2 for-loop ⇒ time complexity is $O(nT)$

Problem 6:

a) ~~i为Lij~~ Suppose the length of the longest subsequence of $s_i s_{i+1} \cdots s_j$ that is also a palindrome is L_{ij}, we have the following recursion. ($i \le j$)

$$L_{ij} = \begin{cases} 1, & i = j \\ 1, & j = i+1 \text{ and } s_i \neq s_j \\ 2, & j = i+1 \text{ and } s_i = s_j \\ L_{i+1,j-1} + 2, & j > i+1 \text{ and } s_i = s_j \\ \max\{L_{i,j-1}, L_{i+1,j}\}, & j > i+1 \text{ and } s_i \neq s_j \end{cases}$$

alg. LONGESTSUB(S):

```
for i=1 to n:
    L[i,i] = 1
for l=2 to n:
    for i=1 to n-l+1
        j = i+l-1
        if j=i+1 and si≠sj:
            L[i,j] = 1
        else if j=i+1 and si=sj:
            L[i,j] = 2
        else if j>i+1 and si=sj:
            L[i,j] = L[i+1,j-1] + 2
        else:
            L[i,j] = max(L[i,j-1], L[i+1,j])
return L[1,n]
```

(b) 为使 supersequence of a given string that is also a palindrome 最短

⇒ ~~string~~ string 的最长回文子串中'对称'字符的对应关系在新的 supersequence 亦会改变

⇒ ~~对于 given string 是 $s_1 s_2 \cdots s_n$~~ 设在 given string 中不属于最长回文子串字符组成的字符串为 S

⇒ 对于 S 中的每个字符，都需要分配它的一个 copy 并插入到原 string 的正确位置。

alg. l = LONGESTSUB(S)

return 2n-l

Problem 7:

(a) T为connected acyclic undirected graph

⇒ remove 任一条边 (u,v)，将形成2个CC，记为 C_1, C_2。不妨 $u \in C_1$, $v \in C_2$.

设 the maximum total weight of a path in T is $Max\_weight(T)$, and the path is P.

Then:

$$Max\_weight(T) = \begin{cases} \max\{Max\_weight(C_1), Max\_weight(C_2)\}, & (u,v) \notin P \\ Max\_depth(C_1, u) + Max\_depth(C_2, v) + w(u,v), & (u,v) \in P \end{cases}$$

其中 $Max\_depth(C, u)$ 是连通分量C中以u为端点的路径的maximum total weight，可通过DFS求得。

具体到alg中：初始时将每个vertex视为一个CC，其需要维护两个值：

(使用bottom-up思想)

V.cur_max：当前包含V的path的maximum weight

V.cur_depth：当前以V为一端点的path的maximum

假设所求的path至少包含一条edge（即不允许单个vertex作为一条path）

alg:

```
for each v ∈ T:
    v.cur_max = -∞
    v.cur_depth = -∞
max_w = -∞
for each (u,v) in T.edges:
    temp1 = u.cur_depth
    temp2 = v.cur_depth
    u.cur_depth = max{w(u,v), temp1, w(u,v)+temp2}
    v.cur_depth = max{w(u,v), temp2, w(u,v)+temp1}
    u.cur_max = max{u.cur_max, w(u,v), w(u,v)+temp1+temp2}
    v.cur_max = max{v.cur_max, w(u,v), w(u,v)+temp1+temp2}
    max_w = max{max_w, u.cur_max, v.cur_max}
return max_w
```

(b) 思想同(a)，为每个vertex维护2个值：v.cur_max 和 v.cur_val.

```
for each v ∈ T:
    v.cur_max = w(v)
    v.cur_val = w(v)
max_w = -∞
for each (u,v) in T.edges:
    u.cur_val += w(v)
    v.cur_val += w(u)
    u.cur_max = max{u.cur_max, u.cur_val}
    v.cur_max = max{v.cur_max, v.cur_val}
    max_w = max{max_w, u.cur_max, v.cur_max}
return max_w
```