Problem 1

(a)

```
Element 0 has label a, parent b, and representative d.
Element 1 has label b, parent d, and representative d.
Element 2 has label c, parent d, and representative d.
Element 3 has label d, it is root, and also representative d
Element 4 has label e, parent f, and representative d.
Element 5 has label f, parent d, and representative d.
Element 6 has label g, parent d, and representative d.
Element 7 has label h, it is root, and also representative h
[Finished in 0.1s]
```

(b)

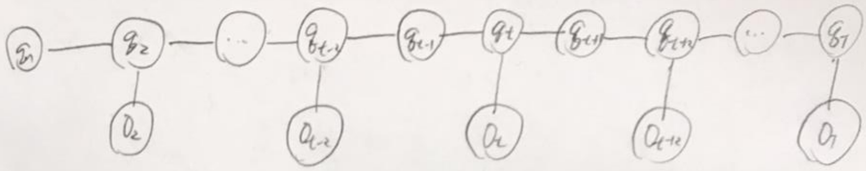
I use an array to implement the data structure. First for the initialization, I can use the Node class to create objects and I append it into nodes array. For initialize the node, I create repr for representative and parent each. This can my code to find out representative and parent separately. And my find_parent function can just return the node.parent since they are linked together. After this, I read the value from edge_list to check link the nodes. This is where I feel like my implementation is not the best runtime solution, since I have to check every element in the nodes which will take O(n) time. It could be a problem and I think the dictionary implementation is a better way since it only take O(1) compare the label from edge_list with nodes. And the rest is follow the pseudo code from the textbook.

Though it’s not the best runtime solution, it still gave me a correct answer. Definitely can be improved if I have more time, and I will try to improve it if HW5 data is too many.

Problem 2.

(a)

problem 2 HMM



a. Known $\alpha_t(i) = P[O_{1:2:t}, q_t = i]$ for even t.

Find $\alpha_2(i)$, $\alpha_t(i)$

$\Rightarrow \alpha_2(i) = P[O_{1:2:2}, q_2 = i]$ since we can't find O_1.

$\Rightarrow \underline{\alpha_2(i) = P[O_2, q_2 = i]}$ #

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j)\, a_{ji}\, b_i(O_t)$$

since we can not find odd O_t, we find previous two steps and transform to $q_{t-1}(i)$, transform again and multiply by the likelihood, we will get $\alpha_t(i)$

$$\Rightarrow \underline{\alpha_t(i) = \sum_{j=1}^{N} \sum_{k=1}^{N} (\alpha_{t-2}(k)\, a_{kj})\, a_{ji}\, b_i(O_t)}$$ #

(b)

b. $\beta_t(i) = P[O_{t+1:2:T} \mid q_t = i]$ for even t.

Find $\beta_T(i)$, $\beta_t(i)$ as func $(\beta_{t+2}(j))$

the backward pass $\beta_T(i)$ is always 1. ∴ $\underline{\beta_T(i) = 1}$ #

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$$

the same reason as the part (a), we can not find odd O.. Therefore, we find even O. and transform to odd O. and transform to the desired state.

$$\beta_t(i) = \sum_{j=1}^{N}\sum_{k=1}^{N} \left(a_{kj}\, b_j(O_{t+2})\right) a_{ij}\, \beta_{t+2}(j)$$
#

(c)

(c) prove or disprove: $P[O_{1:2:T}] = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$ for even t.

$$\alpha_t(i)\cdot\beta_t(i) = \sum_{i=1}^{N} P[O_{1:2:t}, q_t = i]\, P[O_{t+1:2:T} \mid q_t = i]$$

$$= \sum_{i=1}^{N} P[O_{1:2:t} \mid q_t = i]\, P[q_t = i]\, P[O_{t+1:2:T} \mid q_t = i]$$

$$= \sum_{i=1}^{N} P[O_{1:2:T} \mid q_t = i]\, P[q_t = i]$$

$$= \sum_{t=1}^{N} P[O_{1:2:T}]$$

proved

(d)

d. $\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$

$$= \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

(e)

e. $\xi_t(i,j) = P[q_t = i, q_{t+1} = j \mid O_{1:2:T}]$ for any $t = 1 : T-1$

$\xi_t(i,j) = P[q_t = i, q_{t+1} = j \mid O, \lambda]$

$= \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$

$$\Rightarrow \xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_t(k)\, a_{kl}\, b_l(O_{t+1})\, \beta_{t+1}(l)}$$

#