# HKOI 2015/16 TFT Q4 Solution

Prepared by: Alex Tung

# Comments

- A medium-hard (but quite standard) problem on trees
- Getting 34 points is very easy
- Expected more full-scorers (current = 1)

# Solution 1

- Calling salt() is O(N)
- For each distribution batch, simulate salt()

- Time complexity: O(NQ)
- Expected score: 34 points

# Solution 1b

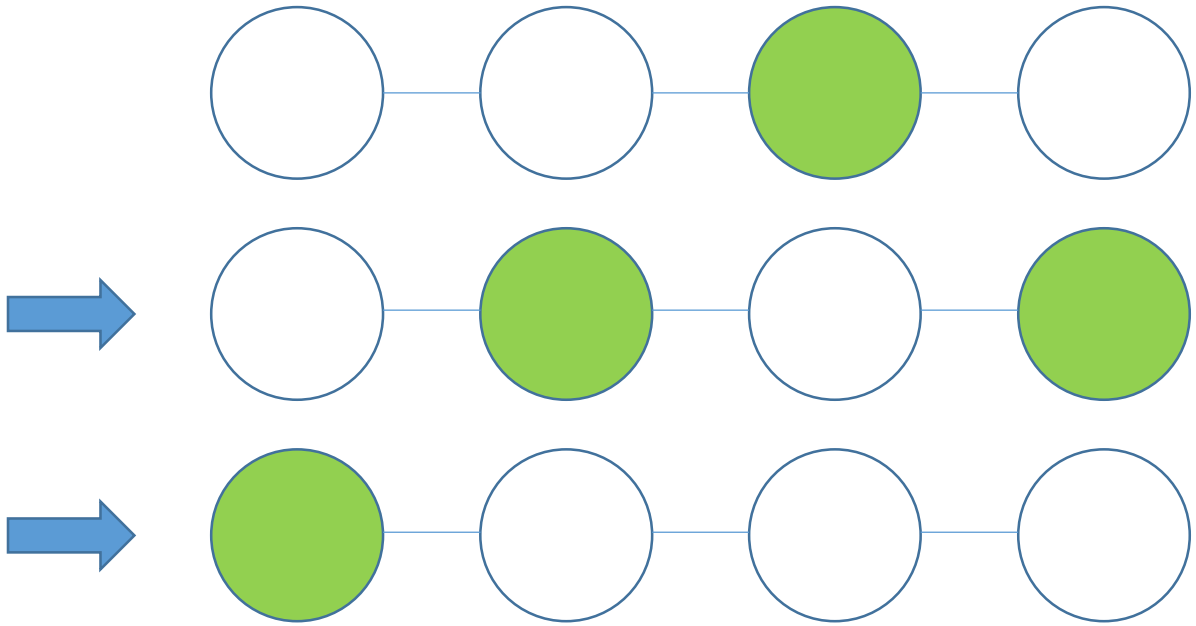
- Note that we can store the batches in an array first

```
for each batch (C, T)
  add T to sto[C]
for i from 1 to N
  run salt(i, sto[i], -1)
```

- Time complexity: $O(N^2 + Q)$
- Expected score: 34 points

# Solution 2 (chain)

- Subtasks 1 + 3: the graph is a chain
- Distribution order:



# Solution 2

- salt(P, ?,-1) --> salt(P-1, ?,P) and salt(P+1, ?,P)
  - Distribution split into ‘left’ part and ‘right’ part
- ‘Left’ part (‘right’ part is similar):
  - salt(P-1, ?,P) --> salt(P-2, ?,P-1) --> salt(P-3, ?,P-2) --> ... --> salt(1, ?,2)
- We can store the parts in two arrays and simulate each part with a single linear scan

# Solution 2

- Step 1: Processing the batches

```
//assume all out-of-bounds elements have value 0
double res[], L[], R[]
for each batch (C, T)
  add T * d_C / (d_C+d_C-1+d_C+1) to res[C]
  add T * d_C-1 / (d_C+d_C-1+d_C+1) to L[C-1]
  add T * d_C+1 / (d_C+d_C-1+d_C+1) to R[C+1]
```

# Solution 2

- Step 2: Linear scan

```
for i from 1 to N
  res[i] = res[i] + R[i] * d_i / (d_i + d_{i+1})
  R[i+1] = R[i+1] + R[i] * d_{i+1} / (d_i + d_{i+1})
for i from N downto 1
  res[i] = res[i] + L[i] * d_i / (d_i + d_{i-1})
  L[i-1] = L[i-1] + L[i] * d_{i-1} / (d_i + d_{i-1})
output res[]
```

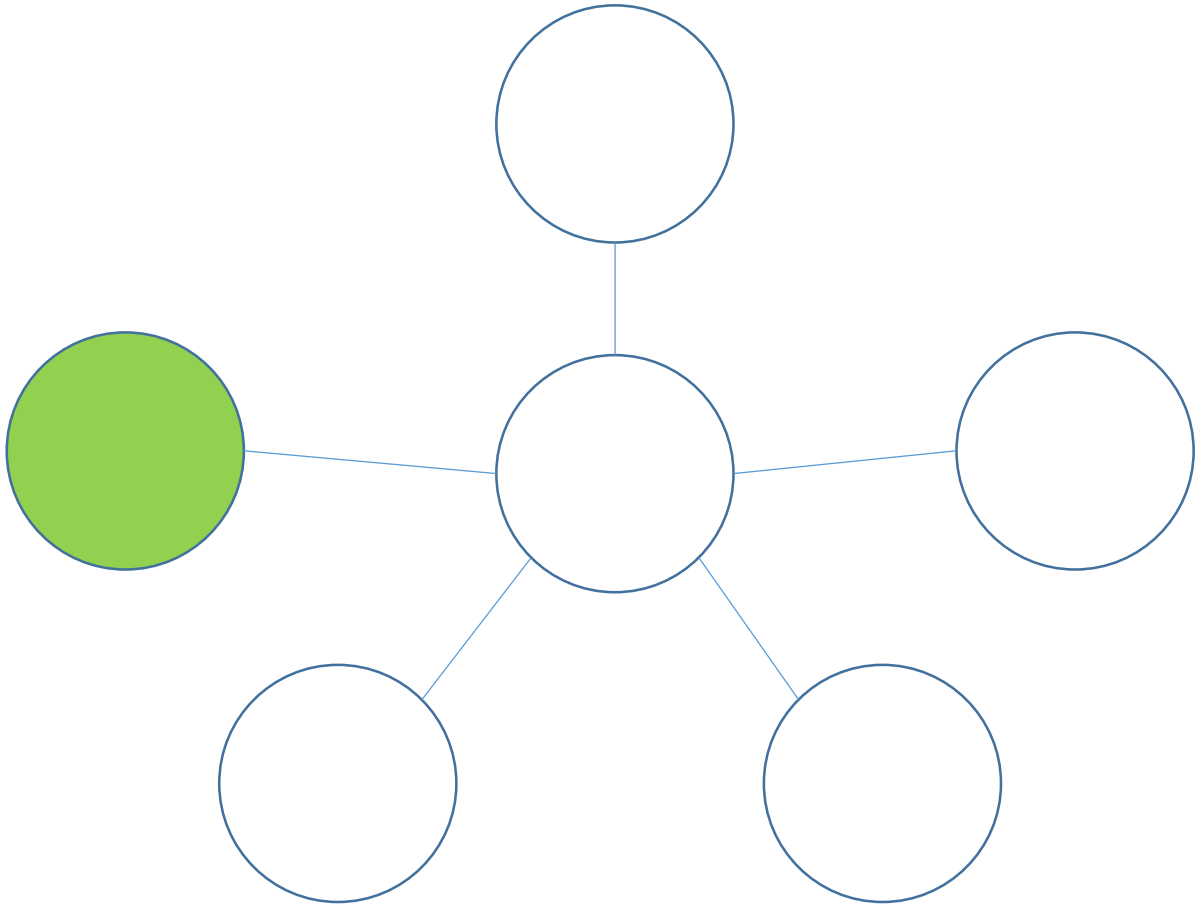
# Solution 2

- Time complexity: O(N + Q)
- Expected score: 11 + 10 = 21
- Expected score (combined with solution 1): 44

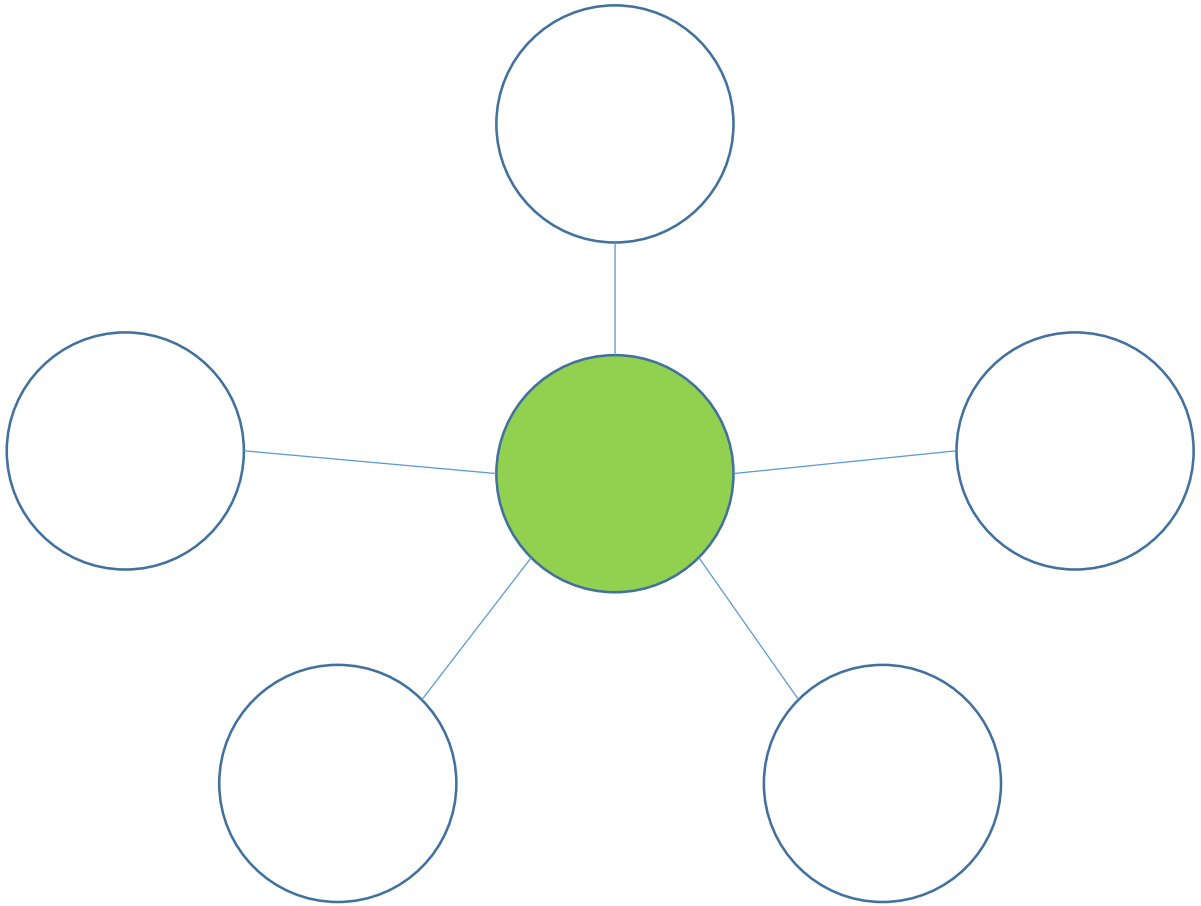
# Solution 3 (star graph)

- Subtask 4: the graph is a star graph

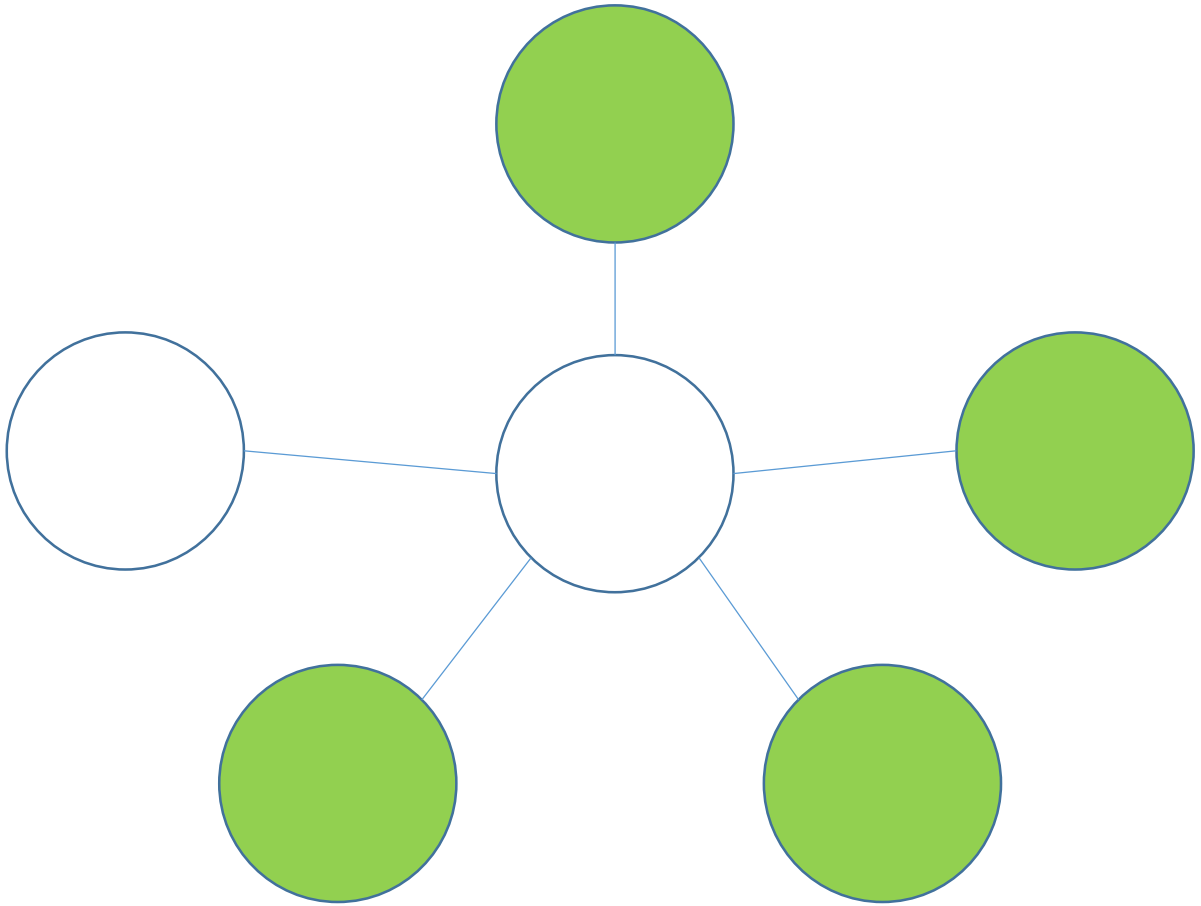
# Solution 3



# Solution 3



# Solution 3



# Solution 3

Main idea:

- salt(C, ?, -1) --> salt(1, ?, C) --> salt(X, ?, 1)
  - X: any integer in [2, N] and not equal to C
- If we can change salt(1, ?, C) to salt(1, ?, -1), we can then process all batches together

# Solution 3

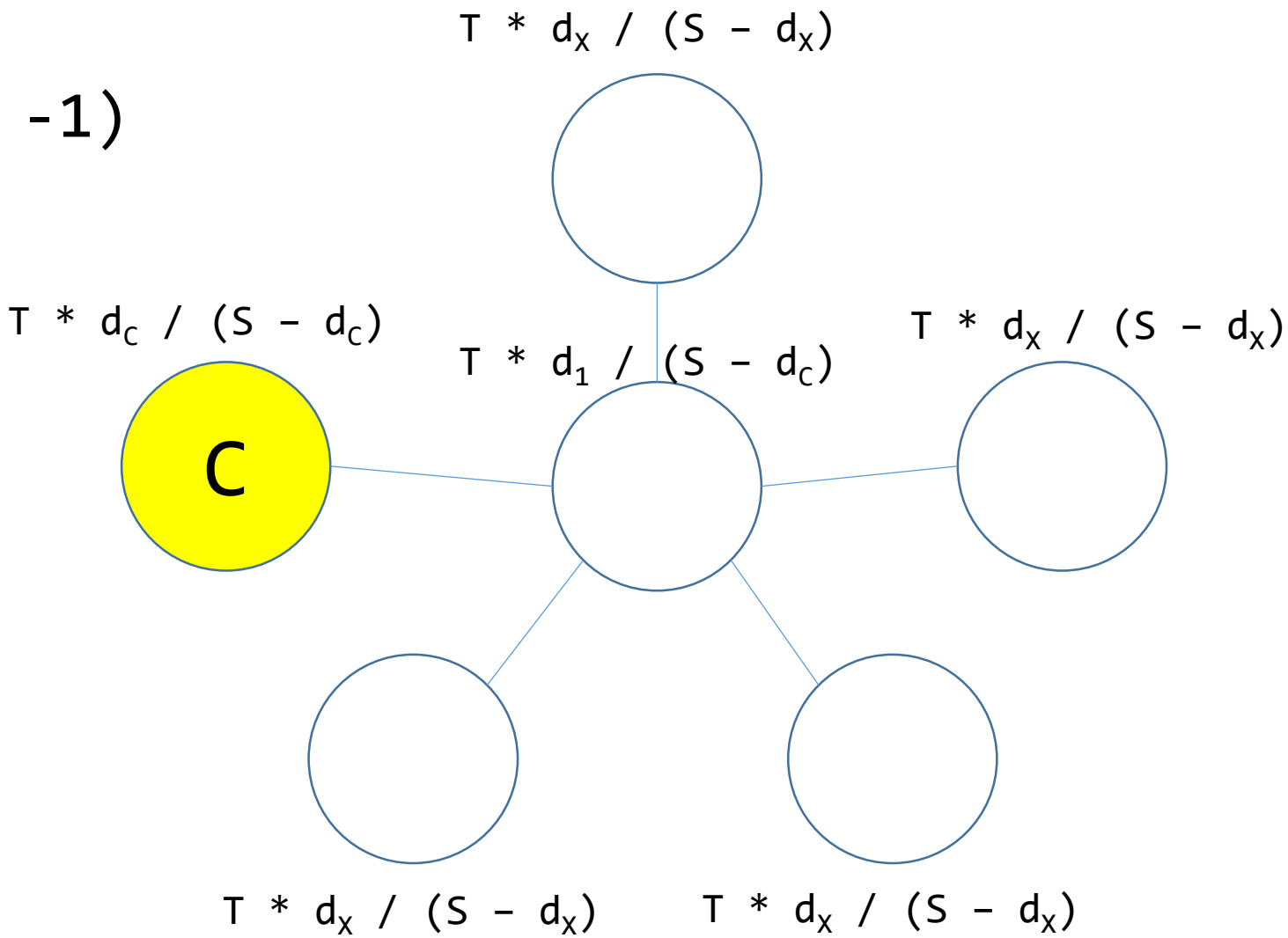
Key observation:

- Let $S = d_1 + \ldots + d_N$
- $salt(1, T, C) = salt(1, T * S/(S–d_C), -1)$
  $+ salt(C, -T * d_C/(S-d_C), 1)$

(‘+’ means accumulating the effects of the two calls)

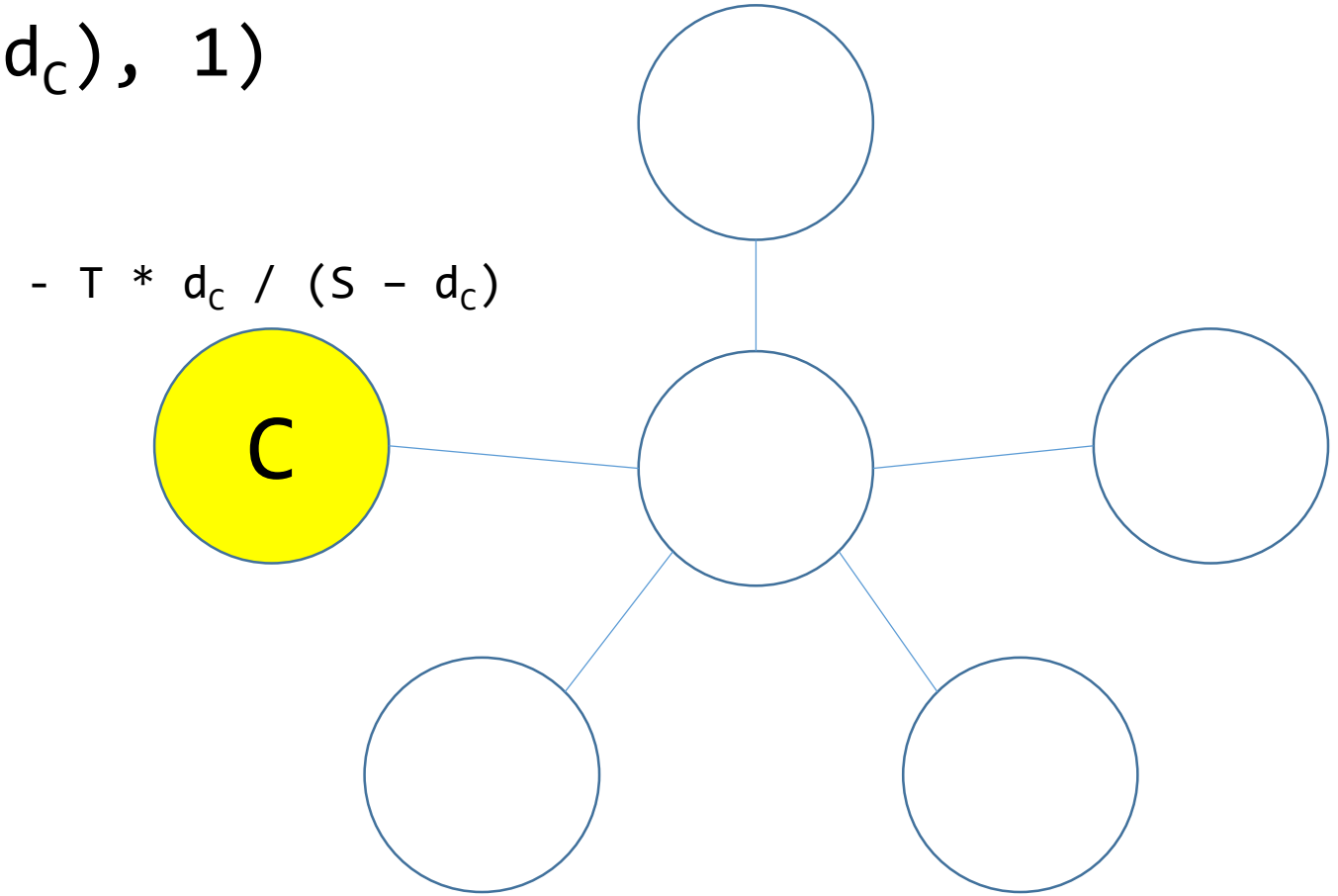
# Solution 3

salt(1, T * S/(S–$d_C$), -1)



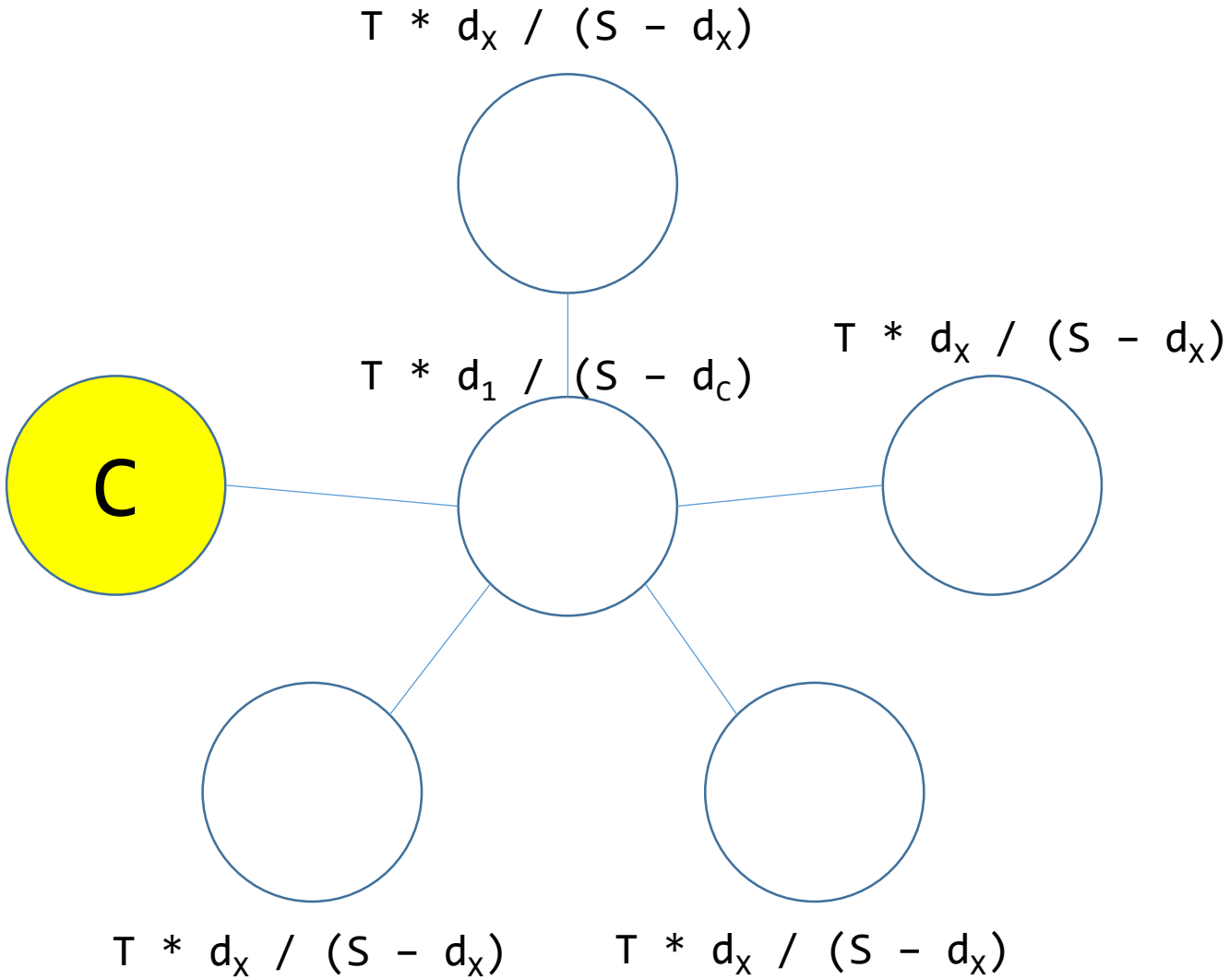
# Solution 3

+ salt(C, -T * $d_C$/(S-$d_C$), 1)



# Solution 3

= salt(1, T, C)



# Solution 3

- Algorithm

```
//sto[i] stores all distribution batches; see solution 1b
double res[], sto[]
for j from 2 to N
  res[j] += sto[j] * d_j / (d_1 + d_j)
  res[j] -= sto[j] * d_j / (d_1 + d_j) * d_j / (sum – d_j)
  sto[1] += sto[j] * d_j / (d_1 + d_j) * sum / (sum – d_j)
run salt(1, sto[1], -1)
output res[]
```

# Solution 3

- Time complexity: O(N + Q)
- Expected score: 15
- Expected score (combined with solutions 1+2): 59

# Solution 4 (full solution)

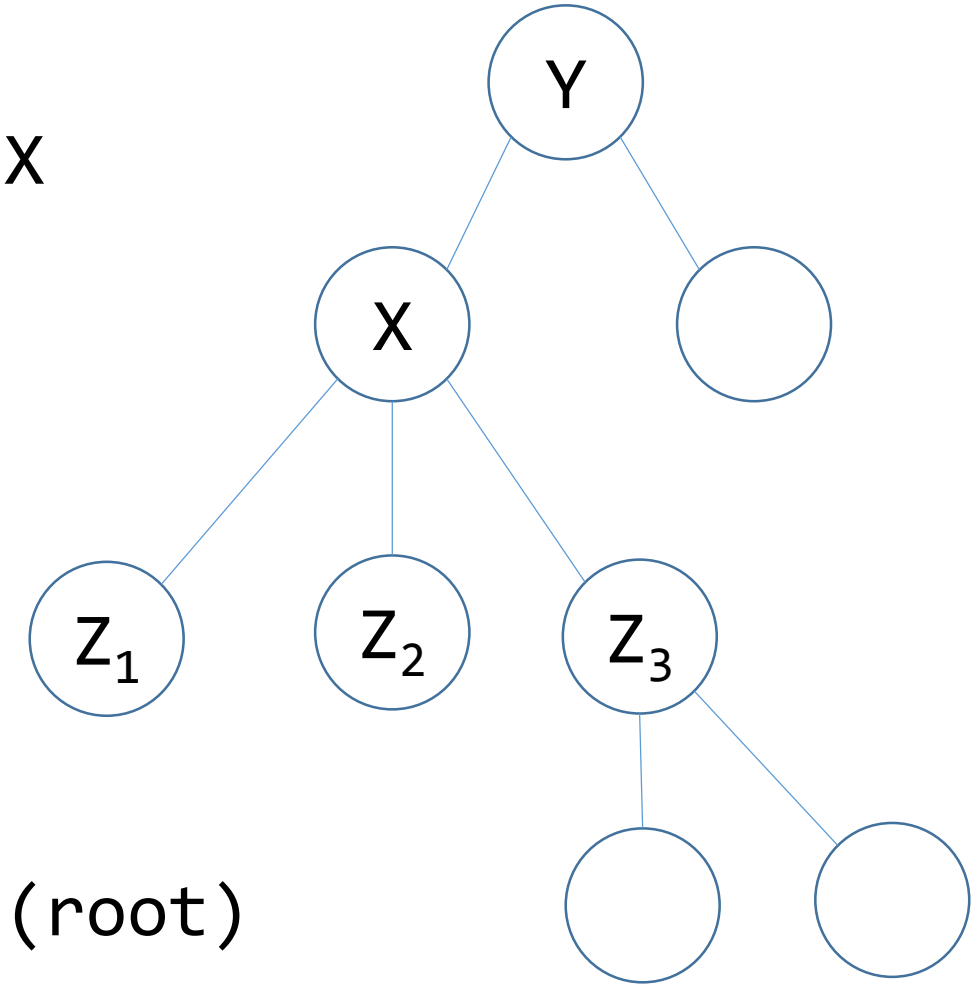
- Root tree at node 1
- If all distributions are ‘downwards’, then a single dfs suffices
- The problem is that sometimes it is ‘upwards’

- Use idea of solution 3 to modify the distributions, making them all ‘downwards’
- Use idea of solution 2 to carry out ‘downwards’ distribution efficiently

# Solution 4

Settings:

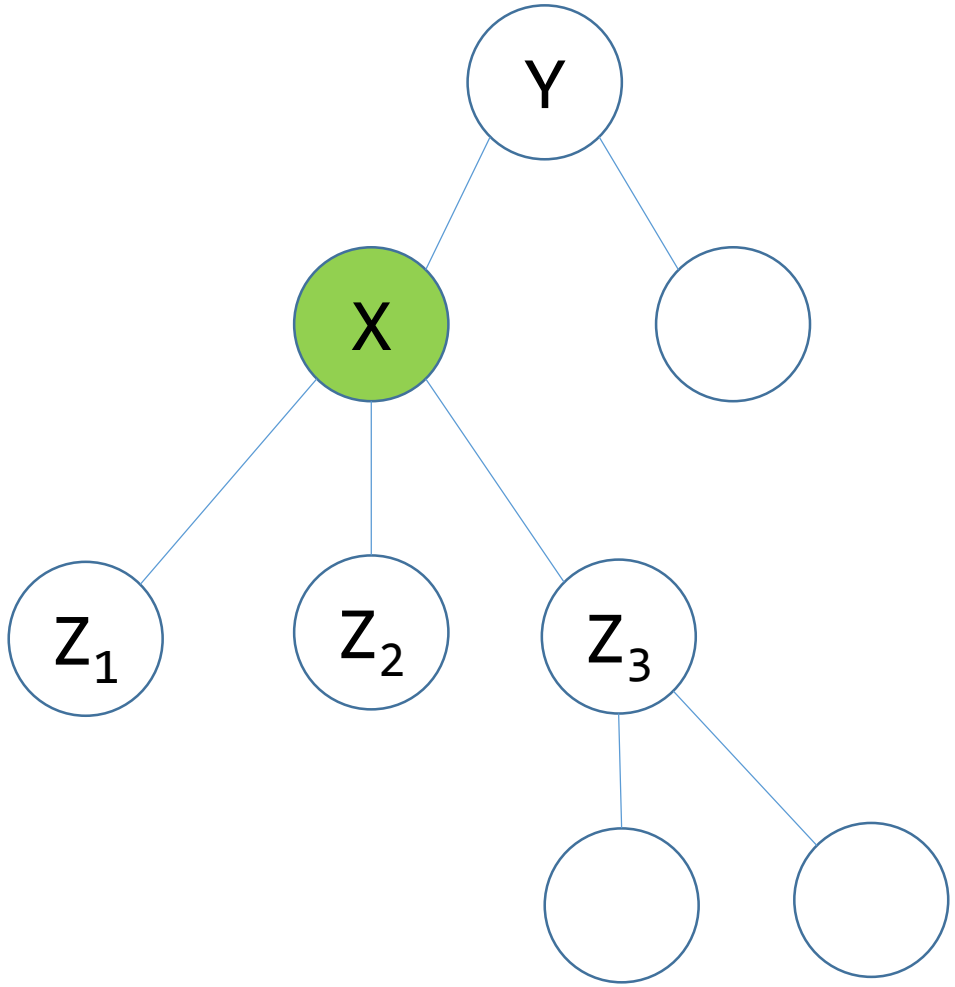
- Let Y be the direct ancestor of X
- Let $Z_i$ be the children of X
- Let $S = d_x + d_Y + sum(d_{Zi})$




- Need to be careful about node 1 (root)

# Solution 4

Case 1: salt(X, T, -1)




salt(X, T, -1)

= salt(X, $T*(S-d_y)/S$, Y)

**+ salt(Y, $T*d_y/S$, X) --> Case 2**

# Solution 4

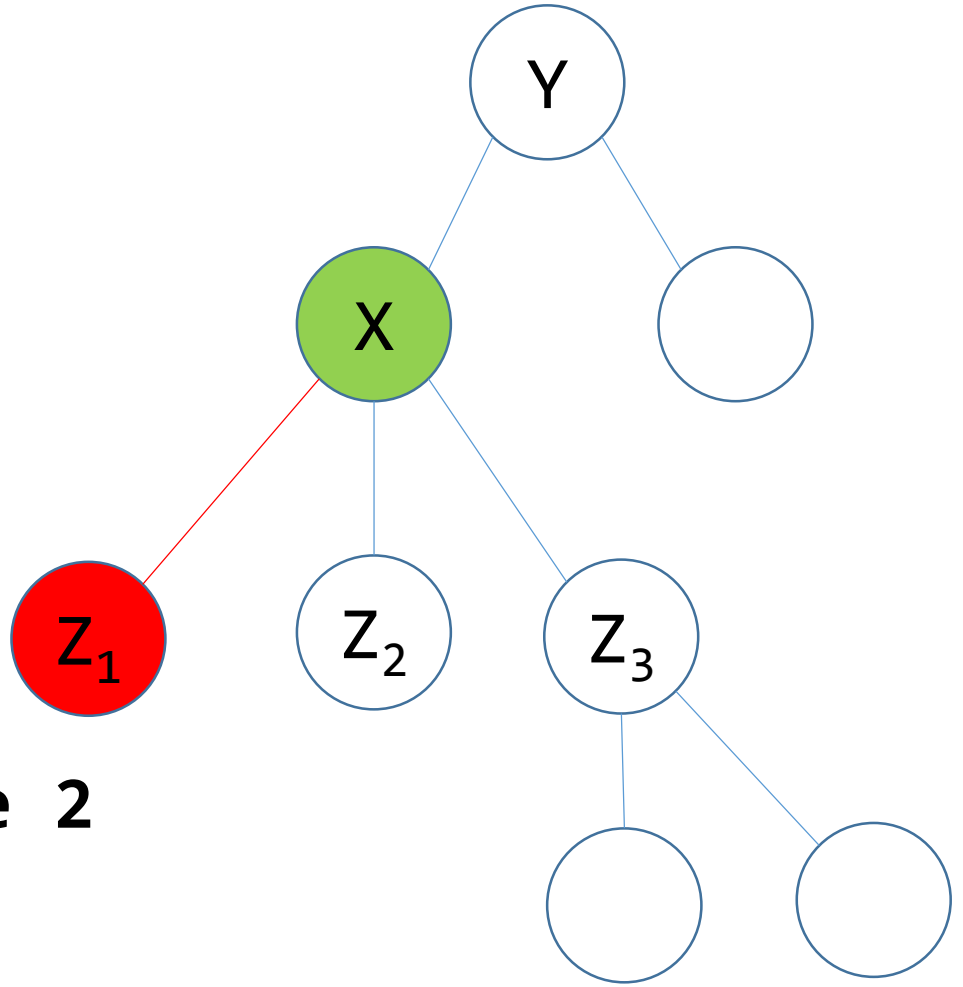
Case 2: salt(X, T, $Z_1$)

salt(X, T, $Z_1$)

= salt(X, T*(S-$d_Y$)/(S-$d_{Z1}$), Y)

+ salt($Z_1$, -T*$d_{Z1}$/(S-$d_{Z1}$), X)

+ **salt(Y, T*$d_Y$/(S-$d_{Z1}$), X) --> Case 2**



# Solution 4

- Algorithm outline

Step 1:

dfs once to transform all salt(X, sto[X], -1) into salt(X, ?, Y), where Y is the direct ancestor of X

Step 2:

dfs again to carry out salt(X, ?, Y) all at once

# Solution 4

- Time complexity: O(N + Q)
- Expected score: 100 :)