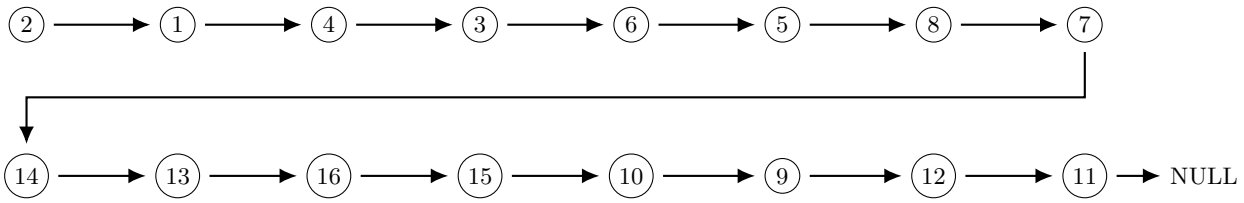
CSCB63 Assignment 4

Jieying Gong
1009388066

Question 1.

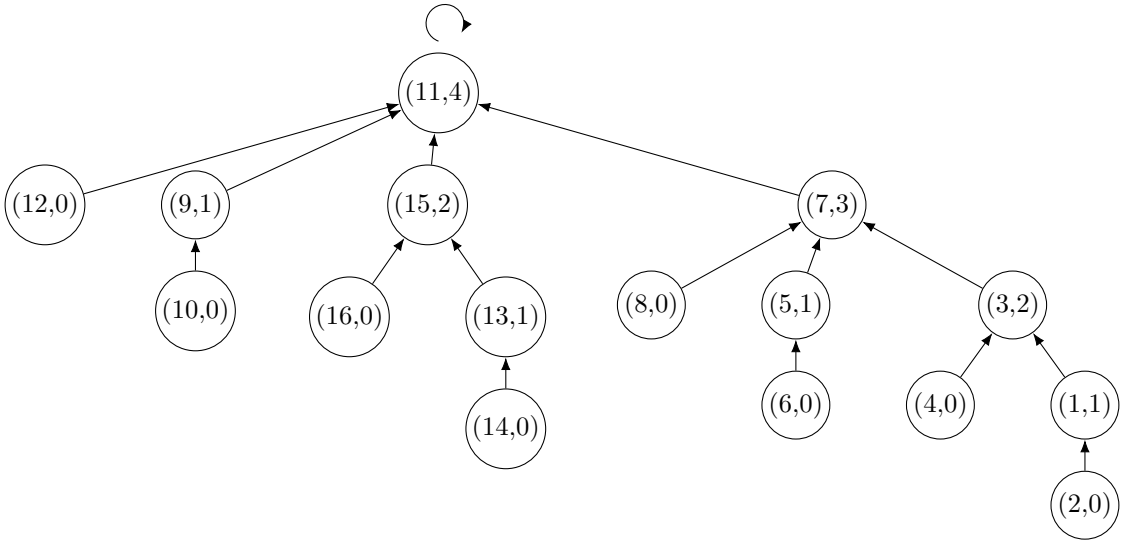
1.1

The graph shows the linked list implementation after the operations as follows.



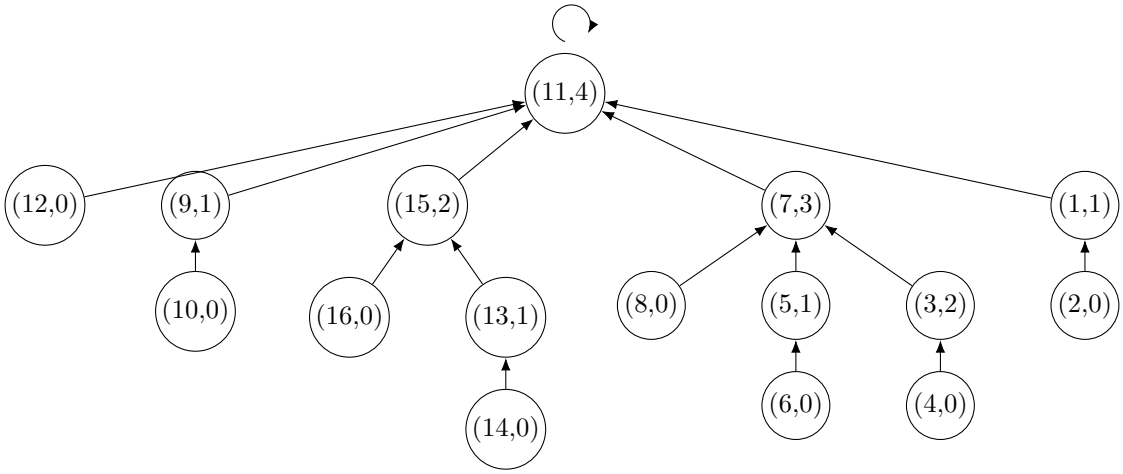
1.2

Note: the pair (x, y) in the node represents (key, rank)! The graph shows the disjoint set with union by rank without path compression implementation after the operations as follows.

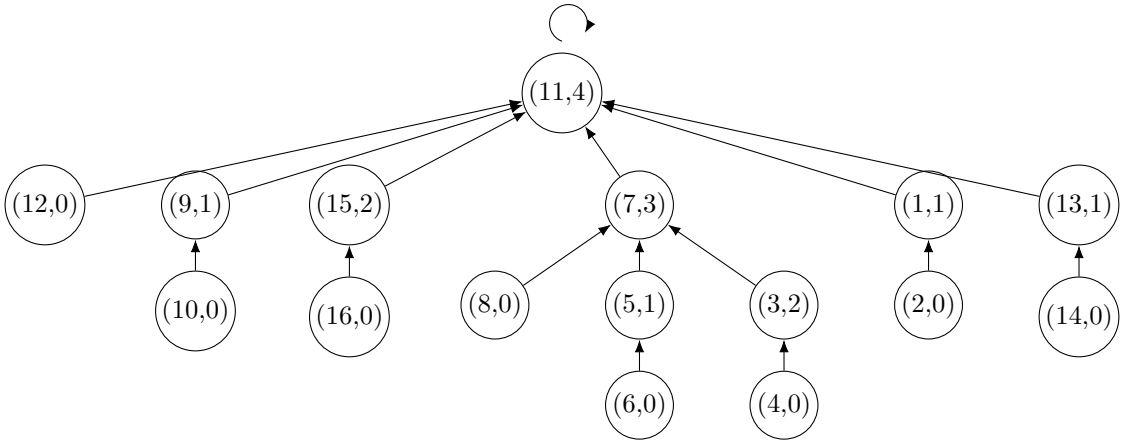


1.3

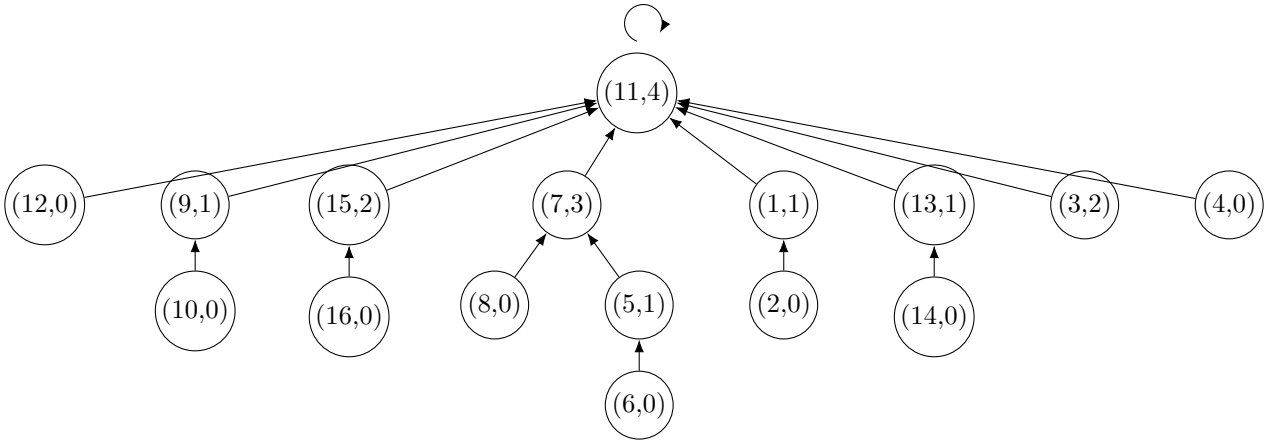
The graph shows the disjoint set with union by rank without path compression implementation after the operations as follows.
(1). after union(1,11):



(2). after find-set(13):



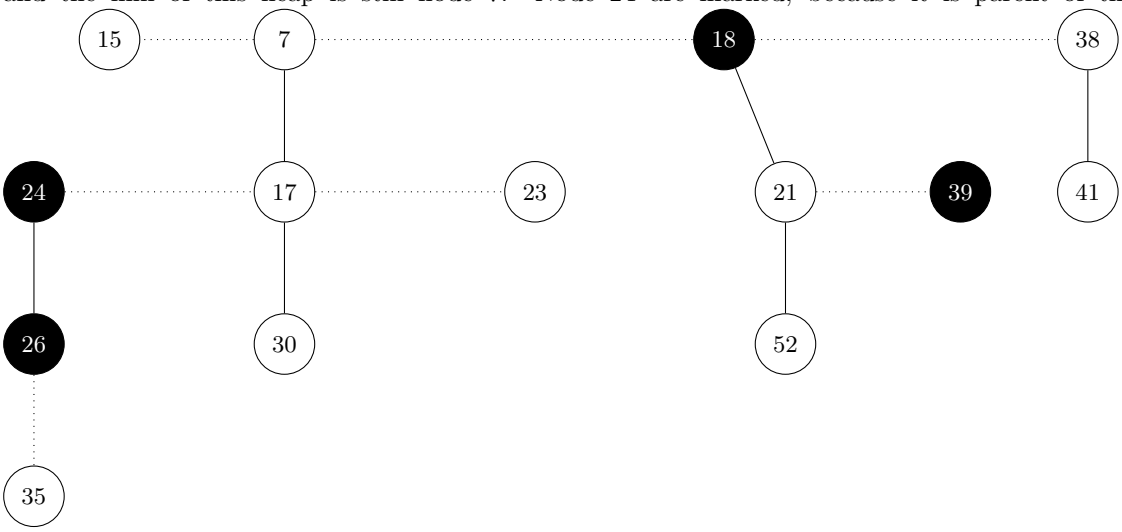
(3). after find-set(4) and also the final result:



Question 2.

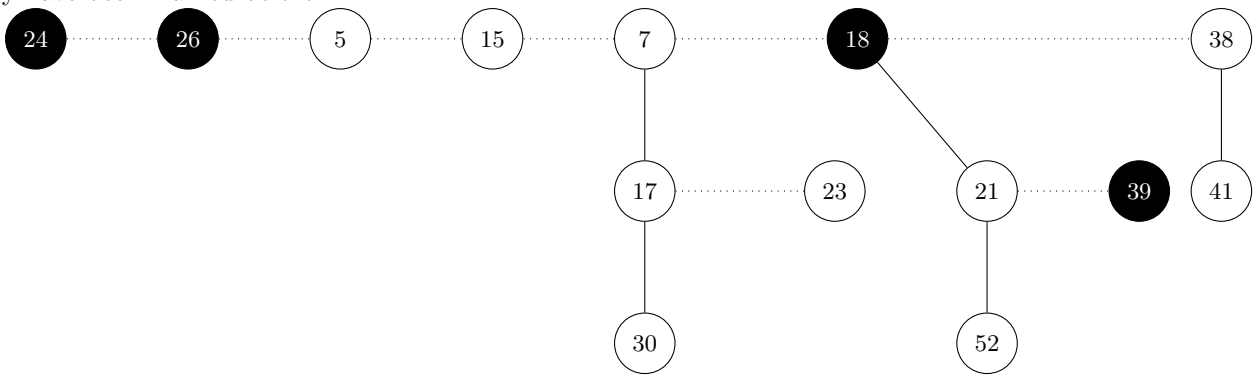
2a.1

The heap after decrease-priority(node 46, 15) as follows. The node with priority 15 is in the root node list and the min of this heap is still node 7. Node 24 are marked, because it is parent of the origin node 46.



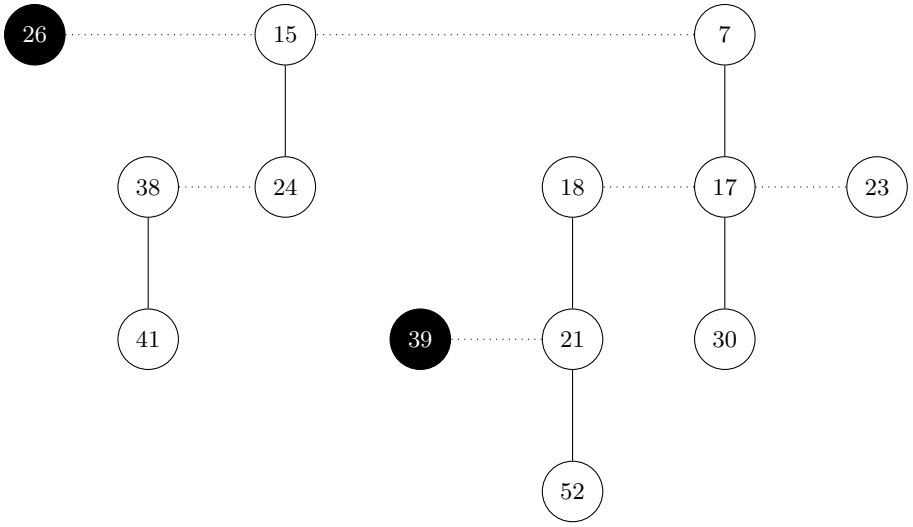
2a.2

The heap after decrease-priority(node 35, 5) as follows. The node with priority 5 is in the root node list and the min of this heap been updated is node 5. Node 24 and 26 are also been cut and get into the root node list, because they have been marked before.



2a.3

The heap been consolidate during extract-min(). The min of heap is node 5 and been extracted after the operation.



2b

The following sequence of operations can lead to required heap:

```
decrease-priority(21, 2)
decrease-priority(38, 8)
extract-min()
```

Question 3.

3.1

Assume m is a power of 2. After $3m$ operations, the smallest possible size we can obtain is m, which results from performing $2m$ additions and m removals. However, in the worst case, the current size may hover between the capacity and half the capacity repeatedly, causing continual expansions and contractions.

Concretely, suppose the size is $\frac{\text{capacity}}{2}$. If we remove an element at that point, the capacity halves to $\text{capacity}' = \frac{\text{capacity}}{2}$, and the new size becomes $\text{size}' = \text{capacity}' - 1$. If the next operation is an addition, then the size increases to $\text{capacity}'$, triggering an expansion back to $\text{capacity}'' = 2 \times \text{capacity}' = \text{capacity}$. Now the new size is $\text{size}'' = \frac{\text{capacity}''}{2} = \frac{\text{capacity}}{2}$. If we remove again, the capacity shrinks once more, and so on. In this scenario, each addition and removal can cost $\text{size} + 1$ time.

Since m is a power of 2, when the size is minimized to m, that size is also a power of 2. Among the $3m$ operations, m of the additions each cost $O(1)$, while the remaining m additions and m removals each incur a cost of $\text{size} + 1 = m + 1$. This leads to a total time of at least $\Omega(m^2)$. Specifically, the total cost in the worst case is:

$$m + 2m \times (\text{size} + 1) \geq m + 2m \times (m + 1) = m + 2m^2 + 2m \geq 5m^2,$$

which belongs to $\Omega(m^2)$ by definition.

3.2

We use the **accounting method**.
Cost design:

- Each `add` operation is charged \$3.
- Each `remove` operation is charged \$2.

add Operation
We aim to prove the following invariant:

$$\text{amount} \geq \max(0,\ 2 \cdot \text{size} - \text{capacity}) \geq 0$$

Initially: `amount` $= 0$, `size` $= 0$, `capacity` $= 0$. The invariant holds. Assume:

$$\text{amount} \geq \max(0,\ 2 \cdot \text{size} - \text{capacity})$$

before an `add` operation.
Case 1: `size < capacity`

$$\begin{aligned} \text{amount}' &= \text{amount} + 3 - 1 = \text{amount} + 2 \\ \text{size}' &= \text{size} + 1 \\ \text{capacity}' &= \text{capacity} \end{aligned}$$

Then:

$$2 \cdot \text{size}' - \text{capacity}' = 2 \cdot \text{size} - \text{capacity} + 2$$

So:

$$\text{amount}' = \text{amount} + 2 \geq 2 \cdot \text{size} - \text{capacity} + 2 = 2 \cdot \text{size}' - \text{capacity}'$$

and clearly:

$$\text{amount}' > 0$$

Hence:

$$\text{amount}' \geq \max(0,\ 2 \cdot \text{size}' - \text{capacity}')$$

Case 2: `size = capacity`
Resizing occurs:

$$\begin{aligned} \text{amount}' &= \text{amount} + 3 - \text{size} - 1 = \text{amount} + 2 - \text{size} \\ \text{size}' &= \text{size} + 1 \\ \text{capacity}' &= 2 \cdot \text{capacity} \end{aligned}$$

Then:

$$2 \cdot \text{size}' - \text{capacity}' = 2 \cdot (\text{size} + 1) - 2 \cdot \text{size} = 2$$

And:

$$\text{amount}' = \text{amount} + 2 - \text{size} \geq 2 \cdot \text{size} - \text{capacity} + 2 - \text{size} = \text{size} - \text{capacity} + 2$$

Since `size = capacity`, we have:

$$\text{amount}' \geq 2 = 2 \cdot \text{size}' - \text{capacity}'$$

Thus:

$$\text{amount}' \geq \max(0,\ 2 \cdot \text{size}' - \text{capacity}')$$

Conclusion: Each `add` operation takes $\mathcal{O}(1)$ amortized time.
remove Operation
We want to prove the invariant:

$$\text{amount} \geq \text{capacity} - \text{size} \geq 0$$

Assume:

$$\text{amount} \geq \text{capacity} - \text{size}$$

before a `remove` operation.
Case 1: `size > capacity / 4`

$$\begin{aligned} \text{amount}' &= \text{amount} + 2 - 1 = \text{amount} + 1 \\ \text{size}' &= \text{size} - 1 \\ \text{capacity}' &= \text{capacity} \end{aligned}$$

Then:

$$\text{capacity}' - \text{size}' = \text{capacity} - \text{size} + 1$$

So:

$$\text{amount}' = \text{amount} + 1 \geq \text{capacity} - \text{size} + 1 = \text{capacity}' - \text{size}'$$

Case 2: `size = capacity / 4` Resizing occurs:

$$\begin{aligned} \text{amount}' &= \text{amount} + 2 - \text{size} - 1 = \text{amount} + 1 - \text{size} \\ \text{size}' &= \text{size} - 1 \\ \text{capacity}' &= \frac{\text{capacity}}{2} \end{aligned}$$

Then:

$$\text{capacity}' - \text{size}' = \frac{\text{capacity}}{2} - \text{size} + 1 = \frac{\text{capacity}}{4} + 1$$

Also:

$$\text{amount}' = \text{amount} + 1 - \text{size} \geq \text{capacity} - 2 \cdot \text{size} + 1 = \frac{\text{capacity}}{2} + 1 \geq \frac{\text{capacity}}{4} + 1 = \text{capacity}' - \text{size}'$$

Thus:

$$\text{amount}' \geq \text{capacity}' - \text{size}'$$

Conclusion: Each `remove` operation also takes $\mathcal{O}(1)$ amortized time.

Question 4.

4.1

The worst-case time of t-delete is $\Theta(slgs)$.

4.2

Let the potential function after the i-th operation be:

$$\Phi(D_i) = c + (c - s)\log s$$

where c and s are the capacity and size after the operation. This function is always non-negative since $s \leq c$ and $\log s \geq 0$ for $s \geq 1$.
We want to show that $\Phi(D_n) - \Phi(D_0) \geq 0$ for all sequence of $n \geq n_0 = 0$. We have $\Phi(D_0) = 0$ as our corner case. Then, $\Phi(D_n) - \Phi(D_0) = \Phi(D_n) \geq 0$

Proof:
Let c and s be the capacity and size before the operation, and c' and s' be the capacity and size after the operation
a. `t-insert`

Case 1: x **is found in an existing node**

- $c' = c,\ s' = s$
- $t_i = \log c + 1 \leq \log(2s) + 1 = \log s + \log 2 + 1$
- $\Phi(D_i) = \Phi(D_{i-1}) = c + (c - s)\log s$

Thus:

$$a_i = t_i = \log c + 1 \leq \log 2s + 1 \leq \log s + \log 2 + 1 \in O(\log s)$$

because $s \geq c/2 \Rightarrow c \leq 2s \Rightarrow c - s \leq s$.
Case 2: x **is not found**

- $c' = c + 1,\ s' = s + 1$
- $t_i = \log c + 1 \leq \log(2s) + 1 = \log s + \log 2 + 1$
- $\Phi(D_{i-1}) = c + (c - s)\log s$
- $\Phi(D_i) = c + 1 + (c - s)\log(s + 1)$

Therefore:

$$\begin{aligned} a_i &= \log c + 1 + (c - s)(\log(s + 1) - \log s) + 1 \\ &= \log c + (c - s)\log\left(\frac{s+1}{s}\right) + 2 \end{aligned}$$

Since $s \geq c/2 \Rightarrow c \leq 2s \Rightarrow c - s \leq s$, we get:

$$a_i \leq \log(2s) + s\log\left(1 + \frac{1}{s}\right) + 2 \leq \log(2s) + \log\left(1 + \frac{1}{s}\right)^s + 2 \leq \log s + \log 2 + \log e + 2 \in O(\log s)$$

by $(1 + 1/k)^k \leq e$.
Therefore, `t-insert` takes amortized time $O(\log s)$.

b. `t-search`

- $c' = c,\ s' = s$
- $t_i = \log c + 1$

- $\Phi(D_i) = \Phi(D_{i-1}) = c + (c - s)\log s$

Thus:

$$a_i = t_i = \log c + 1 \leq \log(2s) + 1 = \log s + \log 2 + 1 \in O(\log s)$$

Therefore, `t-search` takes amortized time $O(\log s)$.

c. `t-delete`

Case 1: x not found
Same as `t-search`, so:

$$a_i \in O(\log s)$$

Case 2: x found, but $s' \geq c/2$

- $c' = c,\ s' = s - 1$
- $t_i = \log c + 1$
- $\Phi(D_{i-1}) = c + (c - s)\log s$
- $\Phi(D_i) = c + (c - s + 1)\log(s - 1)$

$$\begin{aligned} a_i &= \log c + 1 + (c - s + 1)\log(s - 1) - (c - s)\log s \\ &= \log c + 1 + (c - s)\log\left(\frac{s-1}{s}\right) + \log(s - 1) \end{aligned}$$

Since $\frac{s-1}{s} \leq 1 \Rightarrow \log((s-1)/s) < 0$,

$$a_i \leq \log c + 1 + \log(s - 1) \leq \log(2s) + 1 + \log s = 2\log s + \log 2 + 1 \in O(\log s)$$

Case 3: x found, and $s' < c/2 \Rightarrow s = s' + 1 = c/2$

- After rebuilding: $c' = s' = s - 1$
- $t_i = \log c + 1 + s\log s + s$
- $\Phi(D_{i-1}) = c + (c - s)\log s$
- $\Phi(D_i) = c' + (s - 1 + s - 1)\log(s - 1) = s - 1$

Then:

$$\begin{aligned} a_i &= \log c + 1 + s\log s + s + \Phi(D_i) - \Phi(D_{i-1}) \\ &= \log c + 1 + s\log s + s + s - 1 - c - (c - s)\log s \\ &= \log c + s\log s + 2s - c - c\log s + s\log s \\ &= \log c + 2s - c + (2s - c)\log s \end{aligned}$$

Since $s = c/2 \Rightarrow 2s - c = 0$ in this case,

$$a_i = \log c \leq \log(2s) = \log s + \log 2 \in O(\log s)$$

Therefore, `t-delete` takes amortized time $O(\log s)$.
Conclusion
In all cases, `t-insert`, `t-search`, and `t-delete` take amortized time $O(\log s)$ under the given potential function.

Question 5.

5.1

$$A_i = \sum_{j=1, j\neq i}^{n} B_{i,j}$$

5.2

Proof.

$$E(B_{i,j}) = \frac{1}{|i-j|+1}$$

$E(B_{i,j})$ is the probability that z_j is an ancestor of z_i.
WLOG, suppose $i < j$, and thus we have $z_i < z_{i+1} < \cdots < z_{j-1} < z_j$.
WTP : z_j is an ancestor of z_i only if z_j is the first element inserted into the tree within the interval $[z_i, z_j]$.

We will prove this by contradiction. Assume z_j is an ancestor of z_i, but z_k, for some $i < k < j$, is the first element inserted into the tree.
Since z_k already exists in the tree and $i < k < j$, when z_j is inserted into the tree, it will go to the left subtree of z_k, and when z_i is inserted, it will go to the right subtree of z_k. Thus, in this case, z_j cannot be an ancestor of z_i, contradicting the assumption.

Therefore, z_j is an ancestor of z_i only if it is the first element inserted into the tree within the interval $[z_i, z_j]$. Thus, the probability that z_j is an ancestor of z_i equals the probability that z_j is the first element inserted in the interval $[z_i, z_j]$, i.e.,

$$E(B_{i,j}) = \frac{1}{|i-j|+1}$$

□

5.3

$$\begin{aligned}
E(A_i) &= E(\sum_{j=1, j\neq i}^{n} B_{i,j}) \\
&= \sum_{j=1, j\neq i}^{n} E(B_{i,j}) \\
&= \sum_{j=1, j\neq i}^{n} \frac{1}{|i-j|+1} \\
&= \sum_{j=1}^{i-1} \frac{1}{i-j+1} + \sum_{j=i+1}^{n} \frac{1}{j-i+1} && \text{Partition the sum into } j < i \text{and } j > i \\
&= \sum_{k=1}^{i-1} \frac{1}{k+1} + \sum_{k=1}^{n-i} \frac{1}{k+1} \\
&\leq \sum_{k=1}^{n} \frac{1}{k+1} + \sum_{k=1}^{n} \frac{1}{k+1} \\
&= \ln n + \ln n \\
&= 2 \cdot \ln n \in O(\ln n) \text{ as wanted}
\end{aligned}$$