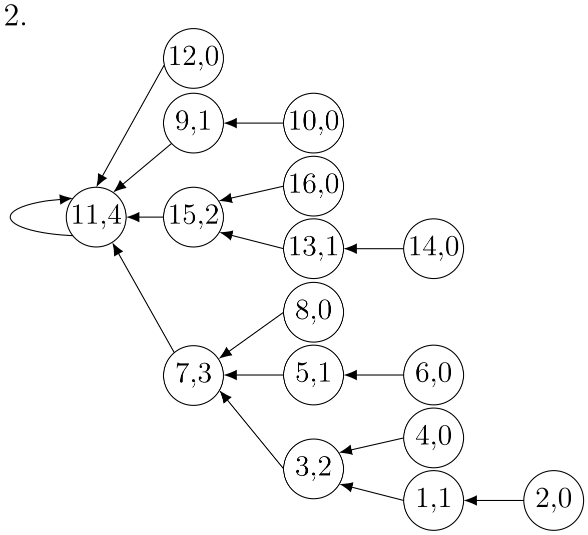
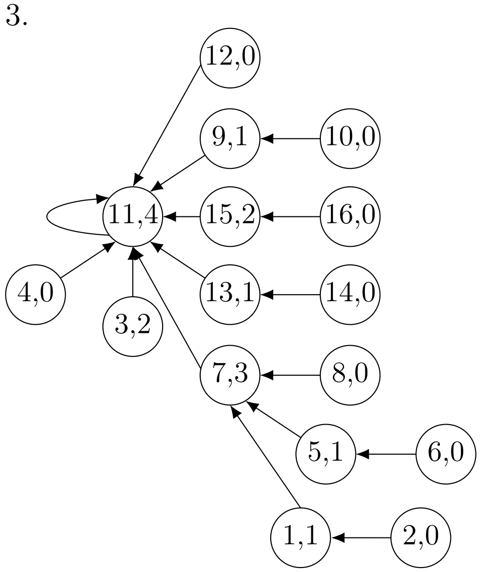
Q$_1$

1. 2 → 1 → 4 → 3 → 6 → 5 → 8 → 7 → 14 → 13 → 16 → 15 → 10 → 9 → 12 → 11

2.



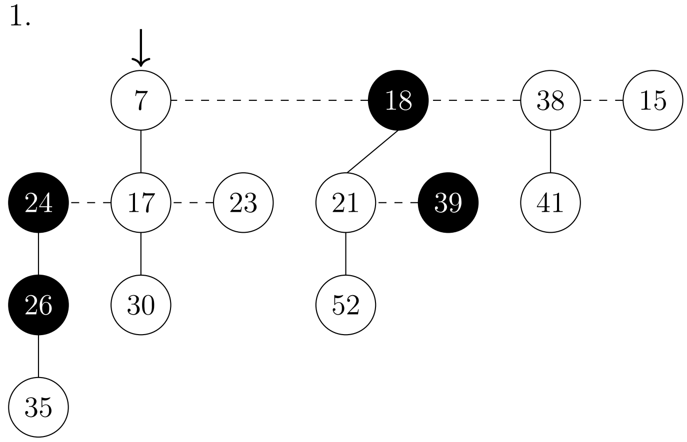
3.



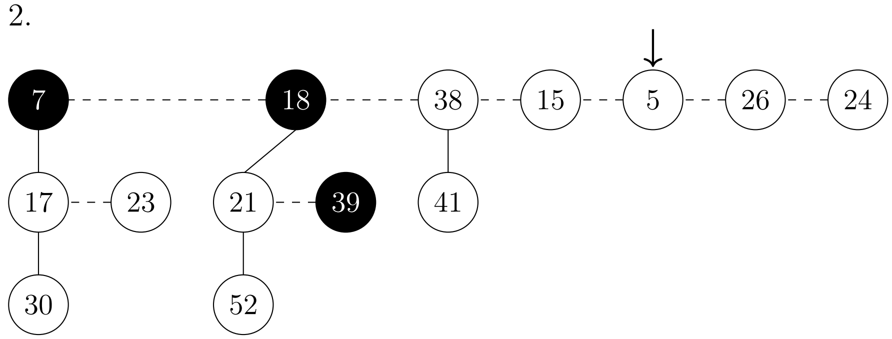
Q$_2$

a.

1.

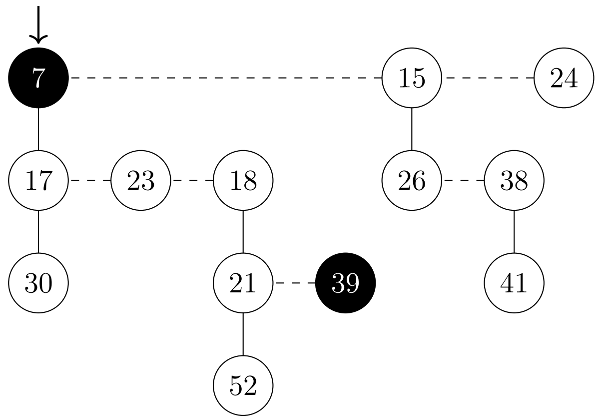


2.



3.

Final one



A[0]=24

A[1]=nil

A[2]= 15

A[3]=7

b.

decrease-priority(38,8)

decrease-priority(21,6)

extract-min()

Q_3

1.

Suppose capacity = size = m

we first use $m/2$ times of remove ()

Then after we execute the $m/2$ time remove (), size $= m/2 = \frac{1}{2}$ capacity.

$\Rightarrow$ We create a new array with capacity $= m/2$ and copy all the remaining $m/2$ elements of the old array into the new array.

$\Rightarrow$ time complexity : $\Omega(m/2)$

Now we use add() one time, as size = capacity $= m/2$

we need to double the array and copy all the $m/2$ elements

$\Rightarrow$ time complexity, $\Omega(m/2)$

At this time,

size $= m/2 + 1$

capacity $= m$

We use remove() again.

After removing the last one, size $= m/2$, capacity $= m$, so we need a new shorter array again and copy all the $m/2$ elements.

$\Rightarrow$ time complexity $= \Omega(m/2)$

Consider the combo: add() + remove(), it can maintain the size and capacity be $m/2$

So after executing remove() $m/2$ times, we can run the combo: add() + remove() for the remaining $3m - m/2 = \frac{5}{2}m$ operations:

$\Rightarrow$ We can run the combo $\frac{5}{2}m \times \frac{1}{2} = \frac{5}{4}m$ times.

Each combo has complexity $\Omega(m/2) + \Omega(m/2) = \Omega(m) \Rightarrow$ the total time complexity is $\Omega\left(m^2\right)$, as wanted.

$\therefore 3m$ operations are: ① remove() $\times m/2$ ② (add()+ remove()) $\times\frac{5}{4}$ m

2.

add(): receive \$7. spend $\begin{cases} \text{if copying: \$ capacity } +1 \text{ (use ``capacity'' before add)} \\ \text{if no copying: \$1} \end{cases}$

remove(): receive \$4, spend $\begin{cases} \text{if copying \$ size } -1 \text{ (use ``size'' before remove)} \\ \text{if no copying: \$1} \end{cases}$

Prove Invariant: amount $\geq$ capacity-size.

Initially: amount $= 0 =$ capacity-size.

remove:

Assume the size before remove is s

Assume the capacity before remove is c

Assume the amount $= a \geqslant c - s$ before remove

if copying:

$a' = a + 4 - (s-1) = a - s + 3$

$c' = \frac{1}{2}c$

$s' = s - 1$

$s - 1 \leq \frac{1}{4}c$ by remove algorithm

$c' - s' = \frac{1}{2}c - s + 1$

$\therefore a' - (c' - s') = a - s + 3 - \left(\frac{1}{2}c - s + 1\right) = a - \frac{1}{2}c + 2 \geqslant c - s - \frac{1}{2}c + 2 =$
$\frac{1}{2}c - s + 2 \geqslant \frac{1}{2}c - 2s + 2$(by $s \geq 0$ in given invariant)

$\because s - 1 \leq \frac{1}{4}c \Rightarrow \frac{1}{2}c - 2s + 2 \geqslant 0 \Rightarrow a' - (c' - s') \geqslant \frac{1}{2}c - 2s + 2 \geqslant 0$

$\Rightarrow a' \geqslant c' - s'$

2° if no copying

$a' = a + 4 - 1 = a + 3$

$c' = c$

$s' = s - 1$

$a' = a + 3 \geqslant c - s + 3 > c - s + 1 = c - (s-1) = c' - s'$

add:

Assume the size before add is s

Assume the capacity before add is c

Assume the amount $= a \geqslant c - s$ before add

if copying:

$a' = a + 3 - (c + 1)$

$c' = 2c$

$s' = s + 1$

$c' - s' = 2c - s - 1$

$a' - (c' - s') = a + 2 - c - (2c - s - 1) = a + 2 - 3c + s + 1 = a - 3c + 3 + s$

Since last copying, no matter whether the last copying is from remove or add, $c/2$ cells have \$6 saved each.

$\Rightarrow$ total saved $\$c/2 \cdot 6 = 3c$

As amount after last copying is amount $\geqslant$ capacity-size $\geqslant 0$, $a \geqslant 3c$

$\Rightarrow a' - (c' - s') = a - 3c + 3 + 5 \geqslant 0$

$\Rightarrow a' \geqslant c' - s'$

if no copying:

$a' = a + 7 - 1 = a + 6$

$c' = c$

$s' = s + 1$

$a' = a + 6 \geqslant c - s + 6 > c - s - 1 = c - (s + 1) = c' - s'$

$\Rightarrow a' \geqslant c' - s'$

By 4 cases mentioned above,

$a' \geqslant c' - s' \geqslant 0$

$\therefore$ add and remove each take $O(7)$ and $O(4)$ amortized time

$\because O(7) = O(4) = O(1)$

$\therefore$ add and remove each take $O(1)$ amortized time.

Q_4

1. t-delete worst-case time: $\Theta(s\,\text{logs})$

2. PF :

t-insert (x) : $O(\lg s)(s \geqslant c/2 \Rightarrow 2s \geqslant c \Rightarrow \lg c \leq \lg(2s) \in O(\lg(s)) \Rightarrow O(\lg(c)) \subseteq O(\lg(s)))$

t-search $(x) = O(\lg s)(s \geqslant c/2 \Rightarrow 2s \geqslant c \Rightarrow \lg c \leqslant \lg(2s) \in O(\lg(s)) \Rightarrow O(\lg(c)) \subseteq O(\lg(s)))$

t-delete (x) : $O(s\lg s)(c + s\lg s \leq 2s + s\lg(s) \in O(s\lg(s)) \Rightarrow O(c + s\lg s) \subseteq O(s\lg(s)))$

Define potential function: $\Phi(T) = c + (c-s)\lg \mathrm{s}$

WTP: $\Phi\left(T_i\right) \geqslant 0 \quad \forall i \in N$

Initially: $\Phi\left(T_0\right) = 0 + (0-0)\lg 0 = 0$

Assume $\Phi\left(T_{i-1}\right) \geqslant 0$

t-insert:

If found used, no change to c and s.

$\Rightarrow \Phi\left(T_i\right) = \Phi\left(T_{i-1}\right) \geqslant 0$

If found unused:

$c_i = c_{i-1}$

$s_i = s_{i-1} + 1$

$\because c_i \geqslant s_i$

$\therefore c_{i-1} \geqslant s_{i-1} + 1$

$$\Phi\left(T_i\right) = c_i + (c_i - s_i)\lg(s_i) = \underset{\geqslant 0}{c_{i-1}} + \left(c_{i-1} - \underset{\geqslant 0}{s_{i-1}} - 1\right)\lg\left(\underset{\geqslant 0}{s_{i-1}} + 1\right) \geqslant 0$$

If not found:

$c_i = c_{i-1} + 1$

$s_i = s_{i-1} + 1$

$\Phi\left(T_i\right) = c_i + (c_i - s_i)\lg(s_i) = \underset{\geqslant 0}{c_{i-1}} + 1 + (c_{i-1} + 1 - \underset{\geqslant 0}{s_{i-1}} - 1)\lg\left(\underset{\geqslant 0}{s_{i-1}} + 1\right) \geqslant 0$

t-search:

no change to c and s, $\Rightarrow \Phi\left(T_i\right) = \Phi\left(T_{in}\right) \geqslant 0$

t-delete:

If not found:

no change to c and $s \Rightarrow \Phi\left(T_i\right) = \Phi\left(T_{i-1}\right) \geq 0$

If found without creating new AVL:

$c_i = c_{i-1}$

$s_i = s_{i-1} - 1$

$\Phi\left(T_i\right) = c_i + \left(c_i - s_i\right)\lg\left(s_i\right) = c_{i-1} + \left(c_{i-1} - s_{i-1} + 1\right)\lg\left(s_{i-1} - 1\right)$

$\because s_i \geqslant c_i/2$ by invariant

$\Rightarrow s_{i-1} - 1 \geqslant c_{i-1}/2$

If $s_{i-1} = 0$, we have $0 - 1 \geqslant c_{i-1}/2$.

$\Rightarrow -2 \geqslant c_{i-1}$

But $c_{i-1} \geqslant 0$.

$\Rightarrow$ contradiction

$\Rightarrow s_{i-1} \neq 0$

If $s_{i-1} = 1$, we have $0 \geqslant c_{i-1}/2$.

$\because c_{i-1} \geqslant 0$

$\therefore c_{i-1} = 0 < s_{i-1} \Rightarrow$ impossible

$\therefore s_{i-1} \neq 1$

$\because s_{i-1} \geqslant 0$

$\therefore s_{i-1} \geqslant 2$

$\therefore s_{i-1} - 1 \geqslant 1 \Rightarrow \lg\left(s_{i-1} - 1\right) \geqslant 0$

$\therefore \Phi\left(T_i\right) = \underset{\geqslant 0}{c_{i-1}} + \left(c_{i-1} - \underset{\geqslant 0}{s_{i-1}} + 1\right)\lg\left(\underset{\geqslant 0}{s_{i-1}} - 1\right) \geqslant 0$

If found and creating new AVL:

$c_i = s_i$

$\Rightarrow \Phi\left(T_i\right) = c_i \geqslant 0$

By the cases mentioned above,

$\Phi\left(T_i\right) \geqslant 0, \forall i \in N$

$\because \Phi\left(T_0\right) = 0$

$\therefore \Phi\left(T_n\right) - \Phi\left(T_0\right) = \Phi\left(T_n\right) - 0 \geqslant 0, \forall n \geqslant 0$

t-insert:

If not found:

$$
\begin{aligned}
\Delta(\Phi) &= \Phi\left(T_i\right) - \Phi\left(T_{i-1}\right) \\
&= c_i + \left(c_i - s_i\right)\lg\left(s_i\right) - \left(c_{i-1} + \left(c_{i-1} - s_{i-1}\right)\lg\left(s_{i-1}\right)\right) \\
&= c_{i-1} + 1 + \left(c_{i-1} + 1 - s_{i-1} - 1\right)\lg\left(s_{i-1} + 1\right) - \left(c_{i-1} + \left(c_{i-1} - s_{i-1}\right)\lg\left(s_{i-1}\right)\right) \\
&= 1 + \left(c_{i-1} - s_{i-1}\right)\left(\lg\left(s_{i-1} + 1\right) - \lg\left(s_{i-1}\right)\right) \\
&= 1 + \left(c_{i-1} - s_{i-1}\right)\left(\lg\left(1 + \frac{1}{s_{i-1}}\right)\right)
\end{aligned}
$$

$\because (1 + 1/k)^k \leq e$

$\therefore \ln(1 + 1/k)^k \leq 1$

$\therefore \ln(1 + 1/k) \leq \frac{1}{k}$

$\therefore \lg(1 + 1/k) = \frac{\ln(1+1/k)}{\ln 2} \leq \frac{1}{(\ln 2)k}(\#)$

$\Rightarrow \Delta(\Phi) = 1 + \left(c_{i-1} - s_{i-1}\right)\left(\lg\left(1 + \frac{1}{s_{i-1}}\right)\right)$

$\leq 1 + \frac{1}{l_{n2}} \cdot \frac{c_{i-1} - s_{i-1}}{s_{i-1}}$

$\because s_{i-1} \geqslant c_{i-1}/2$ by invariant

$\therefore \frac{c_{i-1} - s_{i-1}}{s_{i-1}} = \frac{c_{i-1}}{s_{i-1}} - 1 \leq 2 - 1 = 1(\triangle)$

$\Rightarrow \Delta(\Phi) \leq 1 + \frac{1}{\ln 2}$

$\therefore a_i = t_i + \Delta(\Phi) \leq \lg\left(s_i\right) + 1 + \frac{1}{\ln 2} \in O(\lg(s_i))$

$\therefore$ In this case, t-insert operation takes amortized time $O(\lg s)$

If found unused:

$$
\begin{aligned}
\Delta(\Phi) &= \Phi\left(T_i\right) - \Phi\left(T_{i-1}\right) \\
&= c_i + \left(c_i - s_i\right) \lg\left(s_i\right) - \left(c_{i-1} + \left(c_{i-1} - s_{i-1}\right) \lg\left(s_{i-1}\right)\right) \\
&= c_{i-1} + \left(c_{i-1} - s_{i-1} - 1\right) \lg\left(s_{i-1} + 1\right) - c_{i-1} - \left(c_{i-1} - s_{i-1}\right) \lg\left(s_{i-1}\right) \\
&= \left(c_{i-1} - s_{i-1}\right)\left(\lg\left(s_{i-1} + 1\right) - \lg\left(s_{i-1}\right)\right) - \lg\left(s_{i-1} + 1\right) \\
&\leq \left(c_{i-1} - s_{i-1}\right)\left(\lg\left(1 + \frac{1}{s_{i-1}}\right)\right) \left(\because \lg\left(s_{i-1} + 1\right) \geq 0\right) \\
&\leq \left(c_{i-1} - s_{i-1}\right)\left(\frac{1}{(\ln 2) s_{i-1}}\right) \text{ by}(\#) \\
&= \frac{1}{\ln 2}\left(\frac{c_{i-1} - s_{i-1}}{s_{i-1}}\right) \\
&\leq \frac{1}{\ln 2} \text{ by}(\triangle)
\end{aligned}
$$

$\Rightarrow a_i \leqslant \lg\left(s_i\right) + \frac{1}{\ln 2} \in O\left(\lg\left(s_i\right)\right)$

$\therefore$ In this case, t-insert operation takes amortized time $O(\lg s)$

If found used:

no change to c and s

$\Rightarrow \Delta(\Phi) = 0$

$\therefore a_i = \lg\left(s_i\right) + 0 = \lg\left(s_i\right) \in O\left(\lg\left(s_i\right)\right)$

$\therefore$ In this case, t-insert operation takes amortized time $O(\lg s)$

t-search:

No change to c and s.

$\Rightarrow \Delta(\Phi) = 0$

$\therefore a_i = \lg\left(s_i\right) + 0 = \lg\left(s_i\right) \in O\left(\lg\left(s_i\right)\right)$

$\therefore$ In this case, t-search operation takes amortized time $O(\lg s)$

t-delete:

If found and creating a new AVL:

$$\Delta(\Phi) = c_i + (c_i - s_i)\lg(s_i) - c_{i-1} - (c_{i-1} - s_{i-1})\lg(s_{i-1})$$

$$= (s_{i-1} - 1) + 0 - c_{i-1} - (c_{i-1} - s_{i-1})\lg(s_{i-1})$$

$\because s_{i-1} - 1 < c_{i-1}/2$ (the condition to creat a new AVL)

$\therefore 2s_{i-1} - 2 < c_{i-1}$

$\therefore \Delta(\Phi) < (s_{i-1} - 1) - (2s_{i-1} - 2) - (2s_{i-1} - 2 - s_{i-1})\lg(s_{i-1})$

$$= -(s_{i-1} - 1) - (s_{i-1} - 2)\lg(s_{i-1})$$

$$\begin{aligned}
a_i &= t_i + \Delta(\Phi) \\
&< s_i \lg(s_i) - (s_{i-1} - 1) - (s_{i-1} - 2)\lg(s_{i-1}) \\
&= (s_{i-1} - 1)\lg(s_{i-1} - 1) - (s_{i-1} - 1) - (s_{i-1} - 1)\lg(s_{i-1}) + \lg(s_{i-1}) \\
&= (s_{i-1} - 1)(\lg(s_{i-1} - 1) - \lg(s_{i-1}) - 1) + \lg(s_{i-1}) \\
&= (s_{i-1} - 1)\left(\lg\left(\frac{s_{i-1} - 1}{s_{i-1}}\right) - 1\right) + \lg(s_{i-1}) \quad (s_{i-1} \geq 1 \text{ as need at least 1 node to be deleted}) \\
&< (s_{i-1} - 1)\left(\lg\left(\frac{s_{i-1} + 1}{s_{i-1}}\right) - 1\right) + \lg(s_{i-1}) \quad (\text{ by } y = \lg(x) \text{is monotonically increasing}) \\
&< (s_{i-1} - 1)\left(\frac{1}{(\ln 2)s_{i-1}} - 1\right) + \lg(s_{i-1}) \, (\text{by } (\#)) \\
&= \frac{1}{\ln 2} - \frac{1}{(\ln 2)s_{i-1})} - s_{i-1} + 1 + \lg(s_{i-1}) \\
&< 1 + \frac{1}{\ln 2} + \lg(s_i + 1) \in O(\lg(s_i))
\end{aligned}$$

$\therefore$ In this case, t-delete operation takes amortized time $O(\lg s)$.

If found but no copying:

$$
\begin{aligned}
\Delta(\Phi) &= c_i + (c_i - s_i)\lg(s_i) - c_{i-1} - (c_{i-1} - s_{i-1})\lg(s_{i-1}) \\
&= c_{i-1} + (c_{i-1} - s_{i-1} + 1)\lg(s_{i-1} - 1) - c_{i-1} - (c_{i-1} - s_{i-1})\lg(s_{i-1}) \\
&= (c_{i-1} - s_{i-1})(\lg(s_{i-1} - 1) - \lg(s_{i-1})) + \lg(s_{i-1} - 1) \\
&= (c_{i-1} - s_{i-1})\left(\lg\left(\frac{s_{i-1} - 1}{s_{i-1}}\right)\right) + \lg(s_i) \\
&< (c_{i-1} - s_{i-1})\left(\lg\left(\frac{s_{i-1} + 1}{s_{i-1}}\right)\right) + \lg(s_i) \\
&< (c_{i-1} - s_{i-1}) \cdot \left(\frac{1}{(\ln 2)s_{i-1}}\right) + \lg(s_i)\ (\text{by } (\#)) \\
&\leqslant \frac{1}{\ln 2} + \lg(s_i)\ (\text{by } (\triangle))
\end{aligned}
$$

$$
\begin{aligned}
\therefore a_i &= t_i + \Delta(\Phi) \\
&= \lg(s_i) + \Delta(\Phi) \\
&\leqslant \frac{1}{\ln 2} + \lg(s_i) + \lg(s_i) \in O(\lg(s_i))
\end{aligned}
$$

$\therefore$ In this case, t-delete operation takes amortized time $O(\lg s)$

If not found:

No change to c and s.

$\Rightarrow \Delta(\Phi) = 0$

$\therefore a_i = 0 + \lg(s_i) \in O(\lg(s_i))$

$\therefore$ In this case, t-delete operation takes amortized time $O(\lg s)$

$\therefore$ Based on the cases mentioned above, each operation takes amortized time in $O(\lg s)$

QED.

Q_5

1. $A_i = \sum\limits_{1 \le j \le n, j \ne i} B_{i,j}$

2. $E\left(B_{i,j}\right) = Pr\left(z_j \text{ be the ancestor of } z_i\right)$

Consider the key array $\left[z_{\min(i,j)}, z_{\max(i,j)}\right]$:

$z_{\min(i,j)} < z_{\min(i,j)+1} \ldots < x < \cdots < z_{\max(i,j)}$

1° If x is the first one to be picked from this array, then $z_{\min(i,j)} < x < z_{\max(i,j)}$.

$\therefore$ z_i and z_j will be separated two sides and z_j can never be the ancestor of z_i.

2° If z_i is the first one to be picked from this array, then we have either $z_j > z_i$ or $z_i > z_j$, but z_j can't be the ancestor of z_i

3° If z_j is the first one to be picked from this array:

If $z_i < z_j$, then z_i will in z_j's left sub-tree.

If $z_j < z_i$, then z_i will in z_j's right sub-tree.

In both cases, z_j can be the ancestor of z_i.

$$
\begin{aligned}
\therefore E\left(B_{i,j}\right) &= Pr\left(z_j \text{ be the ancestor of } z_i\right) \\
&= Pr\left(z_j \text{ is the first one to be picked in } \left[z_{\min(i,j)}, z_{\max(i,j)}\right]\right) \\
&= \frac{1}{|i-j|+1}
\end{aligned}
$$

3.

PF:

If $i \ne 1$ or $n \quad (i \in (1, n))$:

$$
\begin{aligned}
E\left(A_i\right) &= E\left(\sum_{j=1}^{i-1} B_{i,j} + \sum_{j=i+1}^{n} B_{i,j}\right) \\
&= \sum_{j=1}^{i-1} E\left(B_{i,j}\right) + \sum_{j=i+1}^{n} E\left(B_{i,j}\right) \\
&= \sum_{j=1}^{i-1} \frac{1}{|i-j|+1} + \sum_{j=i+1}^{n} \frac{1}{|i-j|+1} \\
&= \sum_{k=2}^{i} \frac{1}{k} + \sum_{k=2}^{n-i+1} \frac{1}{k} < 2\sum_{k=1}^{n} \frac{1}{k} < 2 \cdot \ln(n) \in O(\ln(n))
\end{aligned}
$$

If $i = 1$:

$$
\begin{aligned}
E(A_1) &= E\left(\sum_{j=2}^{n} B_{1,j}\right) \\
&= \sum_{j=2}^{n} E(B_{1,j}) \\
&= \sum_{j=2}^{n} \frac{1}{j} \\
&< \sum_{j=1}^{n} \frac{1}{j} < \ln(n) \in O(\ln(n))
\end{aligned}
$$

If $i = n$:

$$
\begin{aligned}
E\left(A_n\right) &= E\left(\sum_{j=1}^{n-1} B_{n,j}\right) \\
&= \sum_{j=1}^{n-1} E\left(B_{n,j}\right) \\
&= \sum_{j=1}^{n-1} \frac{1}{n-j+1} \\
&= \sum_{k=2}^{n} \frac{1}{k} < \sum_{k=1}^{n} \frac{1}{k} < \ln(n) \in O(\ln(n))
\end{aligned}
$$

By 3 cases mentioned above, $E\left(A_i\right) \in O(\ln(n))$.

QED.