CSCI 6057 Advanced Data Structures - Assignment 2

Yansong Li
B00755354
yansong.li@dal.ca

Question 1.

a) Prove by induction.

Basis case:

When k is even, let $k = 2$,

$$\sum_{j=0}^{2-1} 2^{\lfloor \frac{j}{2} \rfloor} = 2 * (2^{\frac{2}{2}} - 1)$$

$$2^{\lfloor \frac{0}{2} \rfloor} + 2^{\lfloor \frac{1}{2} \rfloor} = 2 * (2^1 - 1)$$

$$2^0 + 2^0 = 2 * 1$$

$$2 = 2 \ \text{check}\surd$$

Otherwise, let $k = 1$,

$$\sum_{j=0}^{1-1} 2^{\lfloor \frac{j}{2} \rfloor} = 3 * 2^{\frac{1-1}{2}} - 2$$

$$2^{\lfloor \frac{0}{2} \rfloor} = 3 * 2^0 - 2$$

$$2^0 = 3 - 2$$

$$1 = 1 \ \text{check}\surd$$

Induction step:

When k is even, assume $\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 2 * (2^{\frac{k}{2}} - 1)$ holds. Prove the equality $\sum_{j=0}^{(k+1)-1} 2^{\lfloor \frac{j}{2} \rfloor} = 3 * 2^{\frac{(k+1)-1}{2}} - 2$ holds for $k + 1$.

$$\sum_{j=0}^{(k+1)-1} 2^{\lfloor \frac{j}{2} \rfloor} = 3 * 2^{\frac{(k+1)-1}{2}} - 2$$

$$2 * \left(2^{\frac{k}{2}} - 1\right) + 2^{\lfloor \frac{k}{2} \rfloor} = 3 * 2^{\frac{k}{2}} - 2$$

$$2 * 2^{\frac{k}{2}} - 2 + 2^{\lfloor \frac{k}{2} \rfloor} = 2 * 2^{\frac{k}{2}} + 2^{\frac{k}{2}} - 2$$

$$2 * 2^{\frac{k}{2}} + 2^{\frac{k}{2}} - 2 = 2 * 2^{\frac{k}{2}} + 2^{\frac{k}{2}} - 2 \ \text{check}\surd$$

Otherwise, assume $\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 3 * 2^{\frac{k-1}{2}} - 2$ holds. Prove the equality

$\sum_{j=0}^{(k+1)-1} 2^{\left\lfloor\frac{j}{2}\right\rfloor} = 2 * (2^{\frac{k+1}{2}} - 1)$ holds for $k+1$.

$$\sum_{j=0}^{(k+1)-1} 2^{\left\lfloor\frac{j}{2}\right\rfloor} = 2 * (2^{\frac{k+1}{2}} - 1)$$

$$3 * 2^{\frac{k-1}{2}} - 2 + 2^{\left\lfloor\frac{k}{2}\right\rfloor} = 2 * 2^{\frac{k+1}{2}} - 2$$

$$3 * 2^{\frac{k-1}{2}} - 2 + 2^{\frac{k-1}{2}} = 2 * 2^{\frac{k+1}{2}} - 2$$

$$2^2 * 2^{\frac{k-1}{2}} - 2 = 2 * 2^{\frac{k+1}{2}} - 2$$

$$2 * 2 * 2^{\frac{k-1}{2}} - 2 = 2 * 2^{\frac{k+1}{2}} - 2$$

$$2 * 2^{\frac{k+1}{2}} - 2 = 2 * 2^{\frac{k+1}{2}} - 2 \text{ check}\surd$$

Therefore, we have proved the equality holds for any positive integer k by induction.

b) Derive the formula.

When k is even,

$$\sum_{j=0}^{k-1} 2^{\left\lfloor\frac{j}{2}\right\rfloor} = 2^{\left\lfloor\frac{0}{2}\right\rfloor} + 2^{\left\lfloor\frac{1}{2}\right\rfloor} + 2^{\left\lfloor\frac{2}{2}\right\rfloor} + 2^{\left\lfloor\frac{3}{2}\right\rfloor} \ldots + 2^{\left\lfloor\frac{k-2}{2}\right\rfloor} + 2^{\left\lfloor\frac{k-1}{2}\right\rfloor}$$

$$= 2^{\frac{0}{2}} + 2^{\frac{0}{2}} + 2^{\frac{2}{2}} + 2^{\frac{2}{2}} + \cdots + 2^{\frac{k-2}{2}} + 2^{\frac{k-2}{2}}$$

$$= 2(2^{\frac{0}{2}} + 2^{\frac{2}{2}} + 2^{\frac{4}{2}} \ldots + 2^{\frac{k-4}{2}} + 2^{\frac{k-2}{2}})$$

We can easily obtain that it is a geometric sequence in the parentheses; hence, we apply the formula of geometric series, which is $S_n = \frac{a(1-r^n)}{1-r}$.

In this case, $a = 2^0$, $r = 2$, and $n = \frac{(k-2)+2}{2} = \frac{k}{2}$.

$$\sum_{j=0}^{k-1} 2^{\left\lfloor\frac{j}{2}\right\rfloor} = 2\left(\frac{2^0\left(1 - 2^{\frac{k}{2}}\right)}{1-2}\right)$$

$$= 2\left(\frac{1-2^{\frac{k}{2}}}{-1}\right)$$

$$= 2\left(2^{\frac{k}{2}} - 1\right)$$

Hence, we have proved $\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 2\left(2^{\frac{k}{2}} - 1\right)$ when k is even.

Otherwise,

$$\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 2^{\lfloor \frac{0}{2} \rfloor} + 2^{\lfloor \frac{1}{2} \rfloor} + 2^{\lfloor \frac{2}{2} \rfloor} + 2^{\lfloor \frac{3}{2} \rfloor} \ldots + 2^{\lfloor \frac{k-2}{2} \rfloor} + 2^{\lfloor \frac{k-1}{2} \rfloor}$$

$$= 2^{\frac{0}{2}} + 2^{\frac{0}{2}} + 2^{\frac{2}{2}} + 2^{\frac{2}{2}} + \cdots + 2^{\frac{k-3}{2}} + 2^{\frac{k-1}{2}}$$

$$= 2(2^{\frac{0}{2}} + 2^{\frac{2}{2}} + 2^{\frac{4}{2}} \ldots + 2^{\frac{k-3}{2}}) + 2^{\frac{k-1}{2}}$$

Again, we can easily obtain that it is a geometric sequence in the parentheses; hence, we apply the formula of geometric series once more.

In this case, $a = 2^0$, $r = 2$, and $n = \frac{(k-3)+2}{2} = \frac{k-1}{2}$.

$$\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 2\left(\frac{2^0\left(1 - 2^{\frac{k-1}{2}}\right)}{1-2}\right) + 2^{\frac{k-1}{2}}$$

$$= 2\left(\frac{1-2^{\frac{k-1}{2}}}{-1}\right) + 2^{\frac{k-1}{2}}$$

$$= 2\left(2^{\frac{k-1}{2}} - 1\right) + 2^{\frac{k-1}{2}}$$

$$= 3 * 2^{\frac{k-1}{2}} - 2$$

Hence, we have proved $\sum_{j=0}^{k-1} 2^{\lfloor \frac{j}{2} \rfloor} = 3 * 2^{\frac{k-1}{2}} - 2$ otherwise.

Question 2.

First, we know that binary decision tree model has the following properties.

1) The elements are formed as a binary tree where each node is labeled $x < y$.
2) Program execution corresponds to root-to-leaf path.
3) Leaf contains the result of comparison.
4) Decision tree corresponds to algorithms that only use comparisons to get knowledge about input.

Assume the elements in the binary decision tree are distinct number from 1 to n. Since there are $n!$ different permutations from n elements, there will be $n!$ sorted output. For each of those sorted output, there will be exactly one input that corresponds with it. Therefore, there are at least $n!$ paths in the binary decision tree, which means the binary decision tree has at least $n!$ leaves.

Meanwhile, we assume the height of the binary decision tree is h, based on the properties of binary tree, it can have at most 2^h leaves.

As a result, we now have an inequality $2^h \geq n!$. Hence, $\log_2 2^h \geq \log_2 n!$, which is $h \geq \log_2 n!$. Next,

$$
\begin{aligned}
h \geq \log_2 n! &= \log_2(n(n-1)(n-2) \dots (2)(1)) \\
&= \log_2 n + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2 2 + 0 \\
&= \sum_{i=2}^{n} \log_2 i \\
&= \sum_{i=2}^{\frac{n}{2}-1} \log_2 i + \sum_{i=\frac{n}{2}}^{n} \log_2 i \\
&\geq \sum_{i=\frac{n}{2}}^{n} \log_2 i \\
&\geq \frac{n}{2} * \log_2 \frac{n}{2} \\
&= \Omega(n \log_2 n)
\end{aligned}
$$

Hence, we have proved that sorting under binary decision tree model at least takes $\Omega(n \log_2 n)$.

Question 3.

a) Prove by substitution.

First, we guess $S(u) = O(u)$, which means $S(u) \leq cu$ for $\exists c > 0, \forall u \geq u_0$. Next, we assume that it is true for all $m < u$.

Therefore, we are assuming $S(m) \leq cm$ for $\forall m < u$. Then, let $\sqrt{u} = m$ and assume $S(\sqrt{u}) \leq c\sqrt{u} - d$ where $d > (c+1)$ is a constant, which holds the inequality $m < u$ (the square root of a number is always smaller than that number). Hence,

$$S(u) = (\sqrt{u} + 1)S(\sqrt{u}) + \sqrt{u}$$

$$\leq (c\sqrt{u} - d)(\sqrt{u} + 1) + \sqrt{u} \text{ substitute } S(\sqrt{u}) \leq c\sqrt{u} - d$$
$$= cu + c\sqrt{u} - d\sqrt{u} + \sqrt{u}$$
$$= cu - (d - c - 1)\sqrt{u}$$
$$\leq cu \text{ when } d - c - 1 > 0$$

Therefore, when we pick the constants c and d appropriately so that $d - c - 1 > 0$, then $S(\sqrt{u}) \leq c\sqrt{u}$, which means the space cost would be $O(u)$. Hence, it proves that $S(u) = O(u)$.

(b) Prove $S(u) = \Omega(u)$.

Similarly, we first guess $S(u) = \Omega(u)$, which means $S(u) \geq cu$ for $\exists c > 0, \forall u \geq u_0$. Next, we assume that it is true for all $m < u$.

Therefore, we are assuming $S(m) \geq cm$ for $\forall m < u$. Then, let $\sqrt{u} = m$, which holds the inequality $m < u$. As a result, we are really assuming $S(\sqrt{u}) \geq c\sqrt{u}$. Hence,

$$S(u) = (\sqrt{u} + 1)S(\sqrt{u}) + \sqrt{u}$$
$$\geq c\sqrt{u}(\sqrt{u} + 1) + \sqrt{u} \text{ substitute } S(\sqrt{u}) \geq c\sqrt{u}$$
$$= c * u + c\sqrt{u} + \sqrt{u}$$
$$= c * u + (c + 1)\sqrt{u}$$
$$\geq cu \text{ when } c > 0$$

Therefore, when $c > 0$, then $S(u) \geq c\sqrt{u}$, which means the space cost would be $\Omega(u)$. Hence, it proves that $S(u) = \Omega(u)$.

Question 4.

(a) Changes to the pseudocode.

Member:

The assignments of $high(x)$ and $low(x)$ change, which are $high(x) = \left\lfloor x/|W|^{\frac{2}{3}} \right\rfloor$ and $low(x) = x \bmod |W|^{\frac{2}{3}}$ now.

Successor:

The assignments of $high(x)$ and $low(x)$ change to $\left\lfloor x/|W|^{\frac{2}{3}} \right\rfloor$ and $x \bmod |W|^{\frac{2}{3}}$ as well, and $\sqrt{|W|}$ change to $|W|^{\frac{2}{3}}$.

Insert:

The assignments of $high(x)$ and $low(x)$ change to $\left\lfloor x/|W|^{\frac{2}{3}} \right\rfloor$ and $x \bmod |W|^{\frac{2}{3}}$ as well.

Delete:

The assignments of $high(x)$ and $low(x)$ change to $\left\lfloor x/|W|^{\frac{2}{3}} \right\rfloor$ and $x \bmod |W|^{\frac{2}{3}}$ as well, and $\sqrt{|W|}$ change to $|W|^{\frac{2}{3}}$.

(b) Running time of each operation.

Member:

The number of recurrences is still 1, but the size of recurrence changes to $u^{\frac{2}{3}}$ (searching a sub widget, which has size $u^{\frac{2}{3}}$). Hence, $T(u) = T\left(u^{\frac{2}{3}}\right) + O(1)$. Let $m = \log_2 u$, so $u = 2^m$. Hence,

$$T(2^m) = T\left(2^{\frac{2m}{3}}\right) + O(1)$$

$$\text{Rename } T(2^m) = S(m)$$

$$S(m) = S\left(\frac{2m}{3}\right) + O(1)$$

$$S(m) = O(\log_2 m) \text{ by Master Theorem}$$

$$\text{Change back from } S(m) \text{ to } T(u)$$

$$T(u) = T(2^m) = S(m) = O(\log_2 m) = O(\log_2 \log_2 u).$$

Hence, the running time of Member is still $O(lglgu)$.

Successor:

The number of recurrences is still 1, but the maximum possible size of recurrence changes to $u^{\frac{2}{3}} + u^{\frac{1}{3}}$ (searching the summary then searching a sub widget). Hence, $T(u) = T\left(u^{\frac{2}{3}}\right) + T\left(u^{\frac{1}{3}}\right) + O(1)$. Next, we apply the same trick that we used for Member to obtain $S(m) = S\left(\frac{2m}{3}\right) + S\left(\frac{m}{3}\right) + O(1)$. Since $\frac{2}{3} > \frac{1}{3}$, the first term dominates; hence, we have $S(m) = S\left(\frac{2m}{3}\right) + O(1)$ again. Therefore, we get $S(m) = O(\log_2 m)$ by Master Theorem, then we change back to $T(u) = O(\log_2 \log_2 u)$. Hence, the running time of Successor is still $O(lglgu)$.

Insert:

The number of recurrences is still 2 as the it is, and the condition that if the second recurrence is true then the first one take $O(1)$ time still holds. However, the possible maximum size of the recurrence changes to $u^{\frac{2}{3}} + u^{\frac{1}{3}}$ too. Hence, we have $T(u) = T\left(u^{\frac{2}{3}}\right) + T\left(u^{\frac{1}{3}}\right) + O(1)$ again. Once more, we apply the same trick and Master Theorem to obtain $T(u) = O(\log_2 \log_2 u)$. Hence, the running time of Insert is still $O(lglgu)$.

Delete:

The number of recurrences is still 2 as same as the Insert operation, and the condition that if the second recurrence is true then the first one take $O(1)$ time still holds as well. For the possible maximum size of the recurrence, it is the same with the Successor operation, which is $u^{\frac{2}{3}} + u^{\frac{1}{3}}$, means it needs to search the summary then search a sub widget. Hence, it gives the same running time as the Successor operation $T(u) = T\left(u^{\frac{2}{3}}\right) + T\left(u^{\frac{1}{3}}\right) + O(1)$. Therefore, after we apply the same trick and Master Theorem, we will obtain the same running time complexity, which is $T(u) = O(\log_2 \log_2 u)$. Hence, the running time of Delete is still $O(lglgu)$.

Overall, we can conclude that the running time of operations are the same as the running time in the original case when we have $\sqrt{u}$ as the size of summary and sub widgets.