# Dalhousie University
# CSCI 6057/4117 — Advanced Data Structures
# Winter 2022
# Assignment 2 Solutions

**Please Note: These solutions are for students in the Winter 2022 version of CSCI 6057/4117 only. They may not be photocopied or distributed in any way, including electronically, to any other person without permission of the instructor.**

**Questions:**

1. [10 marks] In the `Locate` algorithm shown in class for the resizable array solution that requires a space overhead of $O(\sqrt{n})$, we saw the following formula:

$$\sum_{j=0}^{k-1} 2^{\lfloor j/2 \rfloor} = \begin{cases} 2 \times (2^{k/2} - 1) & \text{if } k \text{ is even;} \\ 3 \times 2^{(k-1)/2} - 2 & \text{otherwise.} \end{cases} \tag{1}$$

   (i) [5 marks] Prove by induction that this equality holds for any positive integer $k$.

   Solution: When $k = 1$, both sides of the equation are equal to 1. Thus the identity holds.

   Assume that this identity holds for $k = n - 1$ where $n \geq 2$. We prove that it holds for $k = n$. There are two cases.

   In the first case, $n$ is an even number. Then $n - 1$ is odd, so $\sum_{j=0}^{(n-1)-1} 2^{\lfloor j/2 \rfloor} = 3 \times 2^{(n-2)/2} - 2$. Thus

$$\begin{aligned} \sum_{j=0}^{n-1} 2^{\lfloor j/2 \rfloor} &= \sum_{j=0}^{(n-1)-1} 2^{\lfloor j/2 \rfloor} + 2^{\lfloor (n-1)/2 \rfloor} \\ &= 3 \times 2^{(n-2)/2} - 2 + 2^{n/2-1} \\ &= \frac{3}{2} \times 2^{n/2} - 2 + \frac{1}{2} \times 2^{n/2} \\ &= 2 \times (2^{n/2} - 1) \end{aligned}$$

   The identify holds for $k = n$ in this case.

   In the second case, $n$ is an odd number. Then $n - 1$ is even, so $\sum_{j=0}^{(n-1)-1} 2^{\lfloor j/2 \rfloor} = 2 \times (2^{(n-1)/2} - 1)$. Thus

$$\begin{aligned}\sum_{j=0}^{n-1} 2^{\lfloor j/2 \rfloor} &= \sum_{j=0}^{(n-1)-1} 2^{\lfloor j/2 \rfloor} + 2^{\lfloor (n-1)/2 \rfloor} \\ &= 2 \times (2^{(n-1)/2} - 1) + 2^{(n-1)/2} \\ &= 3 \times 2^{(n-1)/2} - 2\end{aligned}$$

Hence the identify also holds for $k = n$ in this case.

(ii) [5 marks] Suppose that you were not given this equality. Instead, derive this formula from scratch. Show your work.

Solution: When $k$ is even, we have

$$\begin{aligned}\sum_{j=0}^{k-1} 2^{\lfloor j/2 \rfloor} &= 2^{\lfloor 0/2 \rfloor} + 2^{\lfloor 1/2 \rfloor} + \cdots + 2^{\lfloor (k-1)/2 \rfloor} \\ &= (2^{\lfloor 0/2 \rfloor} + 2^{\lfloor 1/2 \rfloor}) + \cdots + (2^{\lfloor (k-2)/2 \rfloor} + 2^{\lfloor (k-1)/2 \rfloor}) \\ &= (2^0 + 2^0) + \cdots + (2^{(k-2)/2} + 2^{(k-2)/2}) \\ &= 2 \times (2^0 + 2^1 + \cdots + 2^{(k-2)/2}) \\ &= 2 \times (2^{k/2} - 1)\end{aligned}$$

When $k$ is odd, we have

$$\begin{aligned}\sum_{j=0}^{k-1} 2^{\lfloor j/2 \rfloor} &= 2^{\lfloor 0/2 \rfloor} + 2^{\lfloor 1/2 \rfloor} + \cdots + 2^{\lfloor (k-1)/2 \rfloor} \\ &= (2^{\lfloor 0/2 \rfloor} + 2^{\lfloor 1/2 \rfloor}) + \cdots + (2^{\lfloor (k-3)/2 \rfloor} + 2^{\lfloor (k-2)/2 \rfloor}) + 2^{\lfloor (k-1)/2 \rfloor} \\ &= (2^0 + 2^0) + \cdots + (2^{(k-3)/2} + 2^{(k-3)/2}) + 2^{(k-1)/2} \\ &= 2 \times (2^0 + 2^1 + \cdots + 2^{(k-3)/2}) + 2^{(k-1)/2} \\ &= 2 \times (2^{(k-1)/2} - 1) + 2^{(k-1)/2} \\ &= 3 \times 2^{(k-1)/2} - 2\end{aligned}$$

2. [10 marks] Prove the $\Omega(n \lg n)$ lower bound on sorting under the binary decision tree model. That is, prove that, given a sequence of $n$ elements, the worst-case running time of any comparison-based algorithm that sorts these elements is $\Omega(n \lg n)$.

Hint: Stirling's approximation may be helpful. It is on page 57 of the CLRS book. You can also find it in the following course note that I wrote for CSCI 3110:

`https://web.cs.dal.ca/~mhe/csci3110/handouts/lecture3.htm`

Solution: The output of a sorting algorithm is a permutation of the input sequence to be sorted. Thus, the number of leaves of the binary decision tree is the number of permutations on $n$ elements, which is $n!$. By Stirling's approximation, we have

$$\lg n! = \lg(\sqrt{2\pi n}(n/e)^n) + \Theta(1) = \Theta(n \lg n)$$

Therefore, the height of the binary decision tree for sorting is $\Omega(n \lg n)$, and this is the lower bound for sorting.

3. [15 marks] In class we learned that the space cost of van Emde Boas trees satisfies

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

(Note that the last additive term on the right-hand side is slightly simpler that what is given in class; you are expected to know that this would not affect our result of analysis when presented using order notation.)

(i) [8 marks] Use the substitution method to prove that $S(u) = O(u)$.

Hint: Review Section 4.3 of the CLRS book if you are not familiar with the substitution method. Pages 85-86 on "Subtleties" are particularly helpful.
`http://site.ebrary.com/lib/dal/docDetail.action?docID=10397652`

Solution: In this question, we assume that $S(1) = c$ for some constant $c$ (it is also Okay to assume that $S(1) = 1$).
We prove by induction that $S(u) \leq cu - d$ for any positive constant $d > c - 1$.
The base case is trivially true.
Assume that the claim holds for $u' < u$. Now we prove it for $u$. We have

$$\begin{aligned} S(u) &= (\sqrt{u} + 1)S(\sqrt{u}) + \sqrt{u} \\ &\leq (\sqrt{u} + 1)(c\sqrt{u} - d) + \sqrt{u} \\ &= cu - d\sqrt{u} + c\sqrt{u} - d + \sqrt{u} \\ &= cu - (d - c + 1)\sqrt{u} - d \\ &\leq cu - d \quad \text{since d-c+1 is positive} \end{aligned}$$

(ii) [7 marks] Prove that $S(u) = \Omega(u)$.

Solution: In this question, we assume that $S(1) = c$ for some constant $c$ (it is also Okay to assume that $S(1) = 1$).

We prove by induction that $S(u) \geq cu$.

The base case is trivially true.

Assume that the claim holds for $u' < u$. Now we prove it for $u$. We have

$$\begin{aligned} S(u) &= (\sqrt{u}+1)S(\sqrt{u}) + \sqrt{u} \\ &\geq (\sqrt{u}+1)(c\sqrt{u}) + \sqrt{u} \\ &= cu + c\sqrt{u} + \sqrt{u} \\ &\geq cu \end{aligned}$$

4. [15 marks] Suppose that we have the following variant of van Emde Boas trees: Let $W$ be a widget whose universe size is $u$ (i.e., $W$ stores $u$ bits). Then, instead of using $\sqrt{u}$ subwidgets each of size $\sqrt{u}$ to represent $W$ as done in class, we use $u^{1/3}$ subwidgets, and the universe size of each subwidget is $u^{2/3}$. The size of the summary widget of $W$ is then set to $u^{1/3}$ so that it still has exactly one bit for each subwidget.

   The above is done for every widget in the van Emde Boas tree when building it recursively. For simplicity, assume that $u^{1/3}$ and $u^{2/3}$ are both integers.

   (i) [6 marks] To perform `Member`, `Successor`, `Insert` and `Delete` operations over this variant, what changes need to make to the pseudocode shown in class?

   The only change is the computation of `high(x)` and `low(x)`. Now `high(x)` is $\lfloor x/u^{2/3} \rfloor$ and `low(x)` is $x \mod u^{2/3}$.

   (ii) [9 marks] Analyze the running time of each operation over this variant.

   For each operation, the running time satisfies the following recurrence $T(u) = T(u^{2/3}) + O(1)$, since in the worst-case, we perform one recursive call on a subwidget, which is of size $u^{2/3}$.

   To solve it, let $m = \lg u$, so that $u = 2^m$. Then $T(2^m) = T(2^{m/(3/2)}) + O(1)$.

   Rename $S(m) = T(2^m)$. Then we have $S(m) = S(m/(3/2)) + O(1)$. By master's theorem, we have $S(m) = O(\log_{3/2} m) = O(\lg m)$.

   Changing back from $S(m)$ to $T(u)$, we have

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