# CSCI 6057
# Advanced Data Structures

# ASSIGNMENT - 2

Balaji Sukumaran (bl664064@dal.ca)

Banner ID: B0094897

# Table of contents

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:

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

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

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

**Solution:**

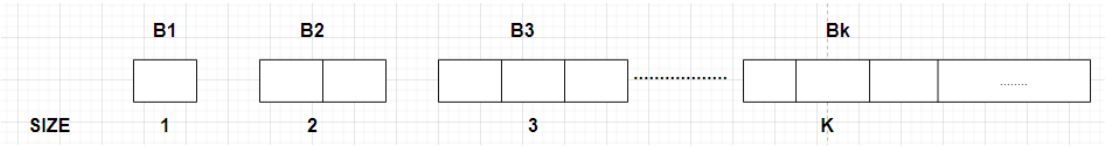
Let's try to derive the formula first:

**<u>Solution (ii)</u>:**

Let us look at two approaches that DON'T work:

*Approach 1*:

Lets, try to store elements in $\theta(\sqrt{n})$ blocks, size of the largest block $\theta(\sqrt{n})$




The total number of elements in the first k blocks is K(K+1)/2 = n

As $K^2 + K - 2n = 0$ is a quadratic equation.

the total number of blocks k required to store n elements is $\left\lceil \left( \frac{\sqrt{1+8n}-1}{2} \right) \right\rceil$ this solves to $\theta(\sqrt{n})$

if we ignore the constants.

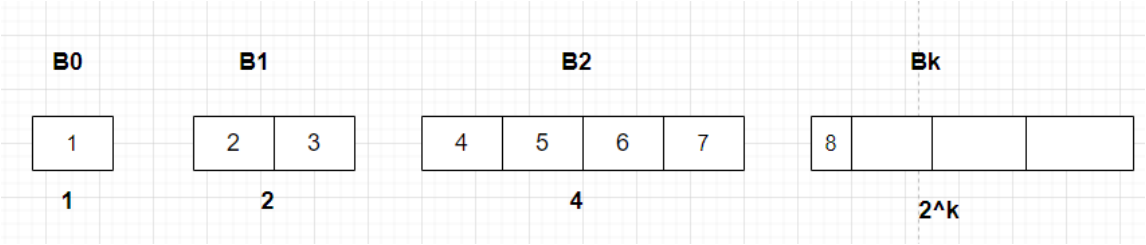
we are taking the ceil because the array supposed to have enough space under any case.

**Locate (i)**: The element **'i'** is the $\left\lceil \left( \frac{\sqrt{1+8i}-1}{2} \right) \right\rceil$ block.

**We cannot go by this approach because the RAM does not support the computation of square root in O(1) time.**

*Approach 2*:

In order to negate the root. We can try doubling the block size each time and insert the data.




**Locate (i):**

Let's see for the block number is related

| $i$ | $i+1$ | $(i+1)_2$ | Block Number |
|---|---|---|---|
| 0 | 1 | 01 | 0 |
| 1 | 2 | 10 | 1 |
| 2 | 3 | 11 | 1 |
| 3 | 4 | 100 | 2 |
| 4 | 5 | 101 | 2 |
| 5 | 6 | 110 | 2 |
| 6 | 7 | 111 | 2 |
| 7 | 8 | 1000 | 3 |

The block number is (the number of bits in binary expression of i+1) -1. Assuming that this can be calculated using the latest processors in constant time.

**The Issue here that the size of the largest block is easily $\theta(n)$**

**Correct Approach:**

Taking both the failed approaches into consideration [1][2], the following model is constructed.


*Figure 1: Generic snapshot of basic data structure [3].*

**Concept:**

- Data blocks are grouped conceptually as super block
- All the data blocks within a superblock are of same size
- When a superblock is full it has $2^{\lfloor k/2 \rfloor}$ data blocks and each block contains $2^{\lceil k/2 \rceil}$, so, a super block contains $2^{\lfloor k/2 \rfloor}$ X $2^{\lceil k/2 \rceil}$ elements

**Locate (i):**

- The element 'i' is the element 'e' of the data block 'b' and superblock 'k'
  - **K = |r| -1**
  - b is the $\lfloor k / 2 \rfloor$ bits of r immediately after the leading 1-bit

Since the superblocks are conceptual we need to know how many data blocks are there in previous superblocks, knowing b and k does not suffice.

$$P = \sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor}$$

Let $f(k) = \sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor}$

Let us evaluate the series in terms of odd and even values seperately

Even Series:

$$f(2k) = \sum_{j=0}^{2k-1} 2^{\lfloor j/2 \rfloor}$$

∴ Any number multiplied by 2 is even

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

$$= \sum_{j=0}^{k-1} 2^{j} + 2^{\lfloor j+0.5 \rfloor}$$

∴ we can ignore the 0.5 since we only consider the floor

$$= 2 \sum_{j=0}^{k-1} 2^{j}$$

$$= 2(2^{k}-1) \quad \text{—②}$$

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

$$f(k) = 2\left(2^{k/2}-1\right) \quad \text{——①}$$

∴ $2^0 + 2^1 + 2^2 + 2^n = a\cdot\left(\frac{1-r^{n+1}}{1-r}\right)$

$$= \frac{1-2^{n+1}}{1-2}$$

$$= 2^{(n+1)} - 1$$

here $n = k-1$

$$= 2^{k-1+1} - 1$$

$$= 2^{k}$$

Odd Series:

$$f(2k+1) = f(2k) + 2^{k}$$

$$= 2(2^{k}-1) + 2^{k} \quad \text{from ②}$$

$$= 3(2^{k}) - 2$$

we have,

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

**Solution (i):**

**Prove by induction:**

$$f(n) = \sum_{j=0}^{n-1} 2^{\lfloor j/2 \rfloor} = \begin{cases} 2 \times (2^{n/2} - 1) & \text{if even} \\ 3 \times 2^{(n-1)/2} - 2 & \text{if odd} \end{cases}$$

Let us split the even and odd series and solve it seperately

For even :-

$$f(n) = \sum_{j=0}^{n-1} 2^{\lfloor j/2 \rfloor} = 2 \times (2^{n/2} - 1) \longrightarrow (1)$$

$$f(2n) = \sum_{j=0}^{2n-1} 2^{\lfloor j/2 \rfloor} \quad \therefore \text{ L.H.S of } (1)$$

$$= \sum_{j=0}^{k-1} 2^{\lfloor 2j/2 \rfloor} + \cdots$$

$$= \sum_{j=0}^{n-1} 2^{\lfloor 2j/2 \rfloor} + 2^{\lfloor (2j+1)/2 \rfloor}$$

$$= \sum_{j=0}^{n-1} 2^{j} + 2^{\lfloor j+0.5 \rfloor}$$

$$= 2 \sum_{j=0}^{n-1} 2^{j} \longrightarrow (2)$$

$$= 2 \times (2^{n} - 1) \quad \therefore f(2n) \text{ by R.H.S} = f(1)$$

$$f(2n) = 2 \sum_{j=0}^{n-1} 2^{j} = 2 \times (2^{n} - 1)$$

We have to prove this by induction

Base-case :- verify for $n = 1$

$$2(2^0) = 2(2^1 - 1)$$
$$2 = 2$$

Stands true for $n = 1$

Induction Step :- Suppose the below formula is true for some value of $k$

$$f(2k) = 2 \sum_{j=0}^{k-1} 2^j = 2 \times (2^k - 1) \quad \text{——— (3)}$$

It should be true for $k+1$

$$f(2(k+1)) = 2 \sum_{j=0}^{k} 2^j \quad \text{——— (4)}$$

expanding $\sum_{j=0}^{k} 2^j$

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

$$= 2 \times (2^{k+1} - 1) \quad \text{——— (5)}$$

From (3) & (5) & (4) hence proved by induction

For odd series

$$f(n) = \sum_{j=0}^{n-1} 2^{\lfloor j/2 \rfloor} = 3 \times 2^{(n-1)/2} - 2$$

$$f(2n+1) = 2\sum_{j=0}^{n-1} 2^j + 2^n \quad \text{——— from (2)}$$

$$f(2n+1) = 2\sum_{j=0}^{n-1} 2^j + 2^n = 3 \times 2^{(2n+1-1)/2} - 2$$

Base case :- for $n = 1$

$$= 2(2^0) + 2^1 = 3 \times 2^1 - 2$$

$$4 = 4$$

Induction Step:- Suppose the below is true for some value of $k$

$$f(2k+1) = 2\sum_{j=0}^{k-1} 2^j + 2^k = 3 \times (2^{(2k+1-1)/2} - 2$$

It should be true for $k+1$

$$f(2(k+1)+1) = 2(2^{k+1} - 1) + 2^{k+1} \quad \text{——— From (3)}$$

$$= 2(2^{k+1}) - 2 + 2^{k+1}$$

$$= 3(2^{k+1}) - 2$$

Hence proved by induction

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 67 of the CLRS book:

https://ebookcentral.proquest.com/lib/dal/detail.action?docID=6925615

Solution:

Binary Decision Tree Model [4]:

- A binary tree where each node is labeled with x<y.
- The program execution corresponds to a path from the root to a leaf, based on the decision of whether x<=y or x>y.
- The leaf contains the result of the computation.
- Decision trees correspond to algorithms that use only comparisons to gain knowledge about the input.

*Figure 2: Binary Decision tree model paths [4]*

- A shorter path from the root to a leaf corresponds to the best case.
- A longer path from the root to a leaf corresponds to the worst case.
- Any binary decision tree for membership must have at least n+1 leaves, one for each element plus "not found."
- As a binary tree with n+1 leaves must have a height of $\Omega(\lg n)$.

Let us consider a binary tree of height k then it has $2^k$ leaves. Which means that there's $2^k$ path from root to leaf. Assuming that there is n nodes in the tree then the height of the tree k corresponds to log n and the number of paths is n!.

$$2^k \geq n!$$

$$k \geq \log(n!) \quad \text{—— (1)}$$

From Stirling's Approximation from CLRS Book

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1+\Theta\left(\frac{1}{n}\right)\right) \quad \text{—— (2)}$$

From (1) and (2)

$$= \log\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)\right)$$

$$= \log(2\pi n)^{1/2} + \log\left(\frac{n}{e}\right)^n + \log\left(1+\Theta\left(\frac{1}{n}\right)\right)$$

$$\geq \log\left(\frac{n}{e}\right)^n \qquad \therefore \text{taking the upper bound}$$

$$\geq n(\log n - \log e)$$

$$\geq n\log n - n\log e$$

$$\geq \Omega(n\log n)$$

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 92-94 on "a trick of the trade" and "avoiding pitfalls" are particularly helpful.

`https://ebookcentral.proquest.com/lib/dal/detail.action?docID=6925615`

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

**Solution:**

The space of cost of van Emde Boas trees [6],

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

Where the following parts of the formula corresponds to:

1. ($\sqrt{u}$ + 1) = Sub-widgets + Summary
2. $\sqrt{u}$ = Pointers mins/max

**Solution (i):**

Let's try to prove that S(u) = O(u) by substitution method,

Guessing that the solution is S(u) = O(u) and try to show that S(u) $\leq$ cu for u > $u_0$ where c, $u_0$ > 0.

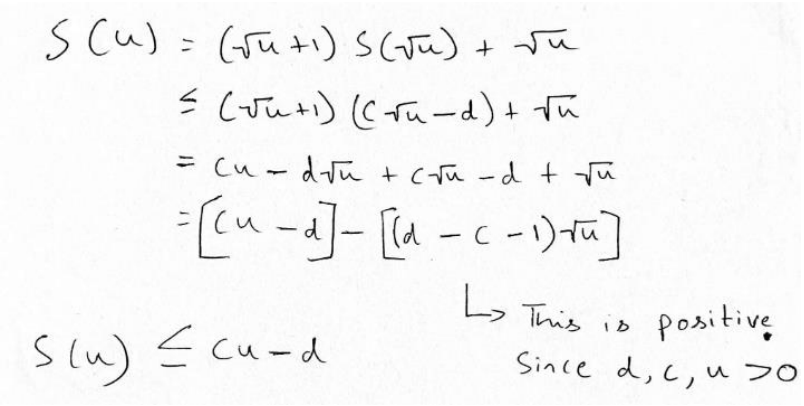
Substituting our guess into recurrence,

$$S(u) = (\sqrt{u} + 1)\, S(\sqrt{u}) + \sqrt{u}$$
$$\leq (\sqrt{u} + 1)\, c\sqrt{u} + \sqrt{u}$$
$$\leq cu + c\sqrt{u} + \sqrt{u}$$
$$\leq cu + (c + 1)\sqrt{u}$$

This does not imply that $S(u) \leq cu$

Let's try subtracting a lower order term d from our guess. d > 0

S(u) ≤ cu - d for u > $u_0$ where c, $u_0$, $d$ > 0

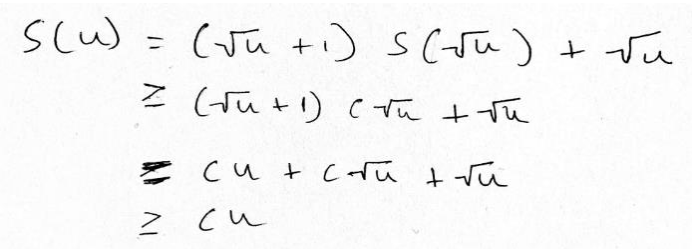



Hence proved.

**Solution (ii):**

Guessing that the solution is S(u) = $\Omega$(u) and let's show that S(u) ≥ cu for u > $u_0$ where c, $u_0$ > 0.

Substituting our guess into recurrence,




Hence proved.

4. [15 marks] Define $[a..b]$ to be an integer interval containing all the integers in $\{a, a+1, \ldots, b\}$.

   Let $P$ be a partitioning of $\{0, 1, \ldots, n-1\}$ into integer intervals. Thus, $P$ is a set of intervals of the form $[0..i_1], [i_1+1..i_2], \ldots [i_k..n-1]$.

   For example, if n = 16, then $P = \{[0..3], [4..5], [6..6], [7..15]\}$ is a possible partition.

   Our problem is to maintain a partitioning, $P$, of $\{0, 1, \ldots, n-1\}$ to support the following operations ($x$ and $y$ are both integers in $[0..n-1]$):

   `test(x, y)`, which tests whether integers $x$ and $y$ are in the same interval in $P$;

   `merge(x)`, which unites the interval containing $x$ with the immediately following interval if $x < n-1$;

   `divide(x)`, which splits the interval, $I$, containing $x$ into two intervals, $I \cap [0..x]$ and $I \cap [x+1..n-1]$, if $x < n-1$.

   In the example above, `test(8, 11)` would return true, `merge(2)` would update $P$ to $\{[0..5], [6..6], [7..15]\}$, and afterwards, performing `divide(9)` would further update $P$ to $\{[0..5], [6..6], [7..9], [10..15]\}$.

Your tasks is to design an $O(n)$-space data structure for this problem, supporting all these operations in $O(\lg \lg n)$ time.

Describe your solutions, including your data structure and the algorithms supporting these operations, and show your time and space analysis. You are not required to give pseudocode, but feel free to give pseudocode if it helps you explain your query algorithms.

Hint: Use van Emde Boas trees.

**Solution:**

This can be achieved by direct application of van Emde Boas tree and following the below rules:

1. Insert/delete the Partitions to support the requried method i.e., test, mergem divide, say the partition is of form {0, $i_1$}, {$i_1 + 1, i_2$}, {$i_2 + 1, i_3$},...{$i_k$, n-1}, we will inserting/deleting $i_1$, $i_2$, $i_2$ ... $i_k$. ----------**(1)**
2. Here the universe u is {0,1,2,3 …. n-1}.

Now, we can leverage the insert, delete, successor, predessesor methods of van Emde Boas tree to support the following methods:

- Divide(x)
- Merge(x)
- Test(x,y)

### Algorithm Explanation:

1. **Divide(x):**
   x is supposed to divide the interval I into two parts I1 which is {0….x} of I and I2 is {x+1….n-1} of I. Since we are storing the partitions **(1)**, we will be inserting the x into our van Emde Boas tree.

   *Pesuedo code for Delete(x):* here W is the widget of the van Emde Boas tree V

```
Divide (x)
    V.Ingert (w, x)
```

```
V. Insert (W, x)
    if min[w] = max[w] = -1
      min[w] ← max[w] ← x
    else if min[w] = max[w]
      (min[w], max[w]) ← (min(x, min[w]),
                          max (x, max[w])
    else
      if x < min[w]
        Swap (x, min[w])
      else if x > max[w]
        Swap (x, max[w])
      w' ← Sub[w][high(x)]
      Insert (w', low(x))
      if max[w'] = min[w']
        Insert (Summary[w] high(x))
```

**Please note that there is only 1 resursive call for the insertions ---- (2)**

2. **Merge(x):** an operation which merges the x with the next interval.
   Since x is just a partition in our data structure removing it, essentially means that the partition in which x is there be merged with the next partition.
   For example, consider partition {0….x},{x+1….n-1} removing x in our data structure means that {0….n-1}.

   Pesuedo code for Merge(x):

```
Merge (x)
    V. Delete (w, x)

V. Delete ( w, x)
    if min[w] = -1
        return
    if min [w] = max[w]
        if min[w] = x
            min[w] ← max[w] ← -1
        return
    if min [Summary[w]] = -1
        if min[w] = x
            min[w] ← max[w]
        else if max[w] = x
                max[w] ← min[w]
        return

    if x = min[w]
        j ← min [Summary[w]]
        min[w] ← min [Sub[w][j]] + j√|w|
        x ← min[w]
    else if x = max[w]
            j ← max[Summary [w]]
            max[w] ← max[Sub[w][j]] + j√|w|
            x ← max[w]
    w' ← Sub [w] [high(x)]
    Ddete (w', low(x))
    if min[w'] = max[w'] = -1
        Delete (Summary[w], high(x))
```

**Please note that there is only 1 resursive call for the Delete ---------- (3)**

3. **Test(x,y):** check whether x, y are in same interval.
   We know that, if x and y are in same interval they will be having the same successor and predessesor since we are storing the interval partitions in a van Emde Boas tree this can easily computed by predessesor and successor methods.

   Pesuedo code for Test(x,y):

```
Test (x,y)
    Successor_x = V. Successor(W,x)
    Successor_y = V. Successor(W,y)
    Set
    predessesor_x = V. predessesor(W,x)
    Predessesor_y = V. predessesor(W,y)
Return (Successor_x == Successor_y)
    && (predessesor_x == predessesor_y)
```

   Successor Psuedo-code [6]:

```
V. Successor (W, x)
    if x < min[W]
        return min[W]
    else if x ≥ max[W]
            return ∞
     W' ← Sub[W][high(x)]
     if low(x)' < max[W']
        return high(x) √|W| + Successor(W', low(x))
     else
        i ← Successor(summary[W], high(x))
        return i √|W| + min[Sub(W)[i]]
```

   Note: Only one recursive call in Successor ----------**(4)**

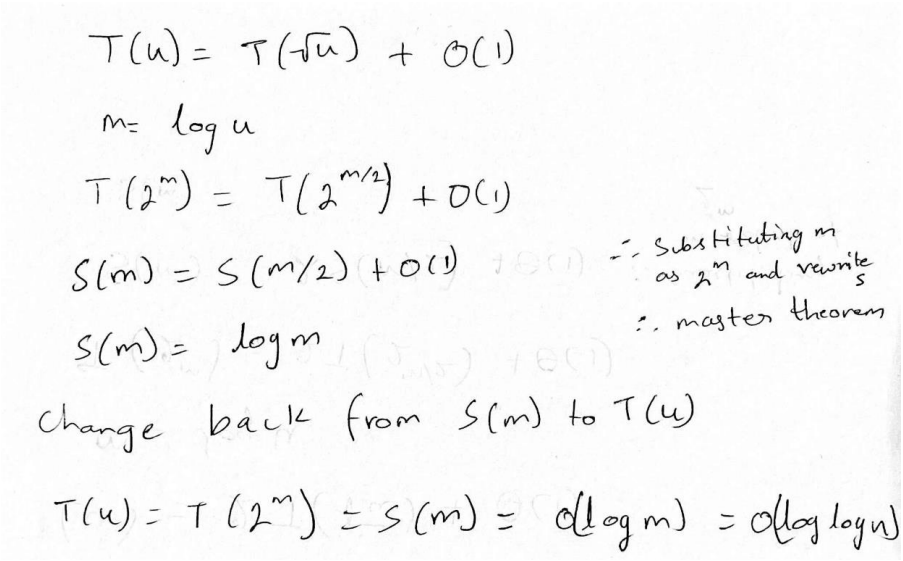
   Predessessor Psuedo-code [7]:

   1. We first search for the predecessor in the cluster in which the key is present.
   2. If we find any predecessor in the cluster then generate its index and return it.

3. Otherwise, search for the previous cluster, with at least one key present, in summary. If any cluster is present then return the index of the maximum of that cluster.
4. If no cluster with that property is present then see if due to lazy propagation, the minimum of the tree(In which the cluster is present) is less than the key, if yes then return minimum otherwise return null.

### Time Complexity:

From (2), (3), (4) we can see that Insert, delete, predessor and successor has 1 recursive call. So the recurrence relation of total time taken T(u) for each of these method can be written as follows:




### Space Complexity:

Since our data structure is a direct application of van Emde Baos tree it has the same space complexity.

**Please note that the universe u of the van Emde Boas tree is actually our n list of elements so if the space complexity is O(u) then it is equal to O(n)**

The space of cost of van Emde Boas trees [6],

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

Where the following parts of the formula corresponds to:

1. $(\sqrt{u} + 1)$ = Sub-widgets + Summary
2. $\sqrt{u}$ = Pointers mins/max

Let's try to prove that S(u) = O(u) by substitution method,

Guessing that the solution is S(u) = O(u) and try to show that S(u) $\leq$ cu for u > $u_0$ where c, $u_0$ > 0.

Substituting our guess into recurrence,

$$S(u) = (\sqrt{u}+1)\, S(\sqrt{u}) + \sqrt{u}$$
$$\leq (\sqrt{u}+1)\, c\sqrt{u} + \sqrt{u}$$
$$\leq cu + c\sqrt{u} + \sqrt{u}$$
$$\leq cu + (c+1)\sqrt{u}$$

This does not imply that $S(u) \leq cu$

Let's try subtracting a lower order term d from our guess. d > 0

S(u) ≤ cu - d for u > $u_0$ where c, $u_0$, $d$ > 0

$$S(u) = (\sqrt{u}+1)\, S(\sqrt{u}) + \sqrt{u}$$
$$\leq (\sqrt{u}+1)(c\sqrt{u} - d) + \sqrt{u}$$
$$= cu - d\sqrt{u} + c\sqrt{u} - d + \sqrt{u}$$
$$= [cu - d] - [(d - c - 1)\sqrt{u}]$$

↳ This is positive since $d, c, u > 0$

$$S(u) \leq cu - d$$

Hence proved. Space complexity is O(u). In terms of our data structure O(n)

## References:


[1] Lecture 6 Video from winter 2022. Accessed: Feb. 13, 2024. [Online Video]. Availability: Lecture 6 video from winter 2022 - CSCI6057 CSCI4117 - Advanced Data Structures - Sec: 01 - 2023/2024 Winter (brightspace.com)

[2] Lecture 6 Video from winter 2022. Accessed: Feb. 13, 2024. [Online Video]. Availability: Lecture 4 video from winter 2022 - CSCI6057 CSCI4117 - Advanced Data Structures - Sec: 01 - 2023/2024 Winter (brightspace.com)

[3] Brodnik, A., Carlsson, S., Demaine, E.D., Ian Ian Munro, J., Sedgewick, R. (1999). Resizable Arrays in Optimal Time and Space. In: Dehne, F., Sack, JR., Gupta, A., Tamassia, R. (eds) Algorithms and Data Structures. WADS 1999. Lecture Notes in Computer Science, vol 1663. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48447-7_4

[4] Lecture 7 Video from winter 2022. Accessed: Feb. 13, 2024. [Online Video]. Availability: Lecture 7 video from winter 2022 - CSCI6057 CSCI4117 - Advanced Data Structures - Sec: 01 - 2023/2024 Winter (brightspace.com)

[5] Introduction to Algorithms Fourth Edition Cormen, Thomas H., et al. Introduction to Algorithms, MIT Press, 2022. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/dal/detail.action?docID=6925615.
Created from dal on 2024-02-11 12:27:40.

[6] Lecture 8 Video from winter 2022. Accessed: Feb. 13, 2024. [Online Video]. Availability: Lecture 8 Video from Winter 2022 - CSCI6057 CSCI4117 - Advanced Data Structures - Sec: 01 - 2023/2024 Winter (brightspace.com)

[7] P. Aakash, “Van Emde Boas Tree”, *GeeksForGeeks*. Accessed: Feb. 13, 2024. [Online]. Availability: Van Emde Boas Tree - Set 3 | Successor and Predecessor - GeeksforGeeks