Analyzed Structures

Van Emde Boas Tree Data Structure

Haania Siddiqui
Khubaib Sattar
Iqra Siddiqui
Shamsa Hafeez

Course: Data Structures II
Habib University
27^{th} May 2021

Splitting bit vector into clusters:

Let the bit vector i.e., our universe have u elements. This means that the range of universe would be $0, 1, 2, ..., u-1$. We split it in $\sqrt{u}$ number of clusters. Each cluster would be of size $\sqrt{u}$.
Dividend = Divisor * Quotient + Remainder
$x = i * \sqrt{u} + j$
$low(x) \ = \ x \, mod \sqrt{u} \ = \ j$
$high(x) \ = \left\lfloor \frac{x}{\sqrt{u}} \right\rfloor \ = \ i$
$V[x] = V.Cluster[i][j]$
$index(i, j) = i\sqrt{u} + j$

Example:
For example x = 12 and u = 16, Then $j = 12 \, mod \sqrt{16} = 0$
$i = \left\lfloor \frac{12}{\sqrt{16}} \right\rfloor = 3$
So, $V[12] = V.Cluster[3][0]$

INSERT(V, x)

if $V.min == None$
    $V.min = V.max = x$ $\triangleright$ $\mathcal{O}(1)$ **time**
    return
if $x < V.min$
    $swap(x \leftrightarrow V.min)$
if $x > V.max$
    $V.max = x)$
if $V.cluster[high(x) == None$
    $Insert(V.summary, high(x))$ $\triangleright$ **First Call**
$Insert(V.cluster[high(x)], low(x))$ $\triangleright$ **Second Call**

If the **first call** is executed, the **second call** only takes $\mathcal{O}(1)$ time. So   [h] saj

Insertion - Insert(V, x):

First Call: Mark cluster high(x) as non-empty i.e., Insert(V.summary, high[x]). Note that V.summary[i] indicates whether V.cluster[i] is non-empty. Also note that V.summary is of size $\sqrt{u}$.
Second Call: Set V.cluster[high(x)][low(x)] to 1 i.e., Insert(V.cluster[high(x)], low[x]). We know that V.cluster[i] is of size $\sqrt{u}$ ($\forall 0 \leq i \leq \sqrt{u}$)
O(1) time: We store minimum and maximum entry in each structure. This gives an O(1) time overhead for each insert operation.
If the first call is executed, the second call takes only O(1) time. So,
$T(u) = T(\sqrt{u}) + O(1)$
$T(u) = O(log \, log \, u)$

Similar recurrence equation is formed for **Successor** and **Delete** functions. This means that both of these functions are represented by :
$T(u) = T(\sqrt{u}) + O(1)$

Solving the recurrence relation:
Our goal is to solve :
$T(u) = T(\sqrt{u}) + O(1)$.....................(i)
According to the master theorem:
$T(n) = aT(\frac{n}{b}) + O(n^d)$.................(ii)
We will try to transform our problem from (i) to (ii).
Let u be the size of the universe with range $\{0, 1, 2, ..., u-1\}$
Consider the following:
$u = 2^k$.....................(iii)
Apply square root on both sides of equation (iii).
$\sqrt{u} = \sqrt{2^k}$
$\sqrt{u} = 2^{\frac{2}{k}}$...............(viii)
Turn the recurrence from recurrence in u to a recurrence in $k = log\, u$ ....(iv)

We define:
$S(k) = T(2^n)$................(v)

Since,
$T(2) \leq O(1)$
$T(u) \leq T(\sqrt{u}) + O(1)$..............(vii)

We have,
$S(1) \leq O(1)$
$S(k) \leq S(\frac{k}{2}) + O(1)$

This means :
$S(k) = O(log\, k)$.............(vi)

Put (viii) in (vii)
$T(u) = T(2^{log\, u})$
Using (v):
$T(u) = S(log\, u)$
Using (vi)
$T(u) = O(log\, log\, u)$

Time Complexity of Dijkstra's Algorithm:
Let n be the number of vertices. In worst case
$n = u$
We know that extract minimum is replaced by the successor function:

So the complexity of the Dijkstra's Algorithm under van emde boas tree data structure would be
O(u log log u)

References:

1. https://www.ics.uci.edu/ eppstein/261/w18-hw6-soln.html

2. http://web.stanford.edu/class/archive/cs/cs166/cs166.1146/lectures/14/Small14.pdf

3. MIT Lecture 4: 6.046J / 18.410J Design and Analysis of Algorithms: Spring 2015: $https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/lecture-notes/MIT6_046JS15_lec04.pdf$ $4.https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2012/lecture-notes/MIT6_046JS12_lec15.pdf$