CS345 : Algorithms II

Ankur Kumar(14109) Tushar Vatsal(14766)

Assignment 5

1 Dynamic Reachability

Algorithm 1: $ProcedureUpdate - R(i, j)$

```
if ($R[i]$ == $true$) and ($R[j]$ == $false$) then
    $R[j] \leftarrow true$;
    for each neighbour q of j do
        $ProcedureUpdate - R[j, q]$end
    end
```

1.1 Amortized Analysis

G refers to the graph existing at any particular moment and c refers to a constant
Potential Function:

$$\phi(G) = c \sum_{R[v]==false} degree(v)$$

where
$\phi(0) = 0$ and thus $\phi(G) \geq 0$ Thus it is a valid potential function
This potential function is nothing but the sum of degrees of all such vertices not reachable from s

Actual cost of edge (i, j) insert operation
Case 1 When $R[i] == false : c$
Case 2 When $R[i] == true\ and\ R[j] == true : c$
Case 3 When $R[i] == true\ and\ R[j] == false : c + \sum degree(v)$ such that v is reachable from j and $R[v] = false = c + P_j$, where $P_j = \sum degree(v)$ such that v is reachable from j and $R[v] = false$

$\Delta(\phi)$ for edge (i, j) insert operation
Case 1 When $R[i] == false : c$ due to an edge insertion
Case 2 When $R[i] == true\ and\ R[j] == true : 0$ due to an edge insertion
Case 3 When $R[i] == true\ and\ R[j] == false$: Sum of degree of all such vertices not reachable from s after insertion of edge (i, j) - Sum of degree of all such vertices not reachable from s before this operation = -(Sum of degree of all such vertices not reachable from s but reachable from j) = -(Sum of degree of all such vertices reachable from j and $R[v] = false$) $= -\sum degree(v)$ such that v is reachable from j and $R[v] = false = -P_j$

Amortized Analysis			
	Actual cost	$\Delta(\phi)$	Amortized cost
Case 1	c	c	$2c$
Case 2	c	0	c
Case 3	$c + P_j$	$-P_j$	c

Thus amortized cost of handling edge insertion here is indeed $O(1)$

2 Tinkering with Fibonacci heap

Yes, all the bounds will still hold on the amortized time complexity of various operations on Fibonacci heap. This time definition of marked nodes is different from what was explained in class. Just colour a node blue when it looses its first child and then colour it black when it looses its second child. When coloured black call that node as marked node. And when a marked node looses its child, unmark the node and take it to the root list of the heap.

2.1 Justification of my claim

Notations
$t(H)$ = number of trees in the root list of Fibonacci heap H
$m(H)$ = number of marked nodes in the Fibonacci heap H
H_{new} = denotes the heap after any operation
H_{old} = denotes the heap before that operation
Potential Function : $\phi(H) = ct(H) + 2cm(H)$

Amortized Analysis			
Operations	Actual cost	$\Delta(\phi)$	Amortized cost
$Decrease\_key(H, x, \Delta)$	$c + cm(H_{old}) - cm(H_{new})$	$cm(H_{new}) - cm(H_{old})$	c
$Extract - min(H)$	$ct(H_{old})$	$\leq ct(H_{new}) - ct(H_{old})$	$\leq ct(H_{new})$
$Insert(x)$	c	c	c
$Find\_min(H)$	c	0	c
$Merge(H_1, H_2)$	c	0	c

We just have to prove that maximum degree of any node in a fibonacci heap is $O(logn)$ which means for any node x of degree k $size(x)$ is $\Omega(a^k)$ for some a.
As done in class for any node x of degree k arrange the children in the order of becoming child to x. Say v_i is the i_{th} child. At the moment when v_i became x's child degree(v_i)$\geq i-1$. From that moment till now, it can loose two child so degree(v_i)$\geq i-3$
Let s_k denote the minimum size of a tree rooted at a node of degree k
Thus $s_k \geq 1+1+1+\sum_{i=3}^{k} s_{i-3}$ Here first 1 stands for the root node itself, the second node stands for first child v_1 and the third 1 stands for v_2. After that we know for any child v_i its degree is $\geq i-3$ and the summation is from third child to last child.
Removing inequality is no problem because we have to find the minimum possible value for s_k so now $s_k = 1+1+1+\sum_{i=3}^{k} s_{i-3} \Rightarrow s_k = (1+1+1+\sum_{i=3}^{k} s_{i-4}) + s_{i-3} \Rightarrow s_k = s_{k-1} + s_{k-3}$ where $s_0 = 1, s_1 = 2, s_2 = 3$
Induction Hypothesis : $s_k = ca^k$ for some $1 < a < 2$
To prove : $s_{k+1} = ca^{k+1}$
$s_{k+1} = s_k + s_{k-2} = ca^k + ca^{k-2}$ Now for $ca^k + ca^{k-2} = ca^{k+1}$, $a^2 + 1 = a^3$
Consider $f(a) = a^3 - a^2 - 1$ for $a = 1$, $f(a) = -1$ and for $a = 2$, $f(a) = 3$ thus $f(a) = 0$ for some $a \in (1, 2)$. There will always exist a base case p such that $s_p \geq a^p$ for the same a satisfying $a^3 - a^2 - 1 = 0$. Thus $s_k = a^k$ for some $a \in (1, 2)$