a) $a_{c_i} = t_{c_i} + \Phi(D_i) - \Phi(D_{i-1})$

$$\sum_{i=1}^{n} a_{c_i} = \sum_{i=1}^{n} \left(t_{c_i} + \Phi(D_i) - \Phi(D_{i-1}) \right)$$

$$\sum_{i=1}^{n} a_{c_i} = \sum_{i=1}^{n} t_{c_i} + \sum_{i=1}^{n} \left(\Phi(D_i) - \Phi(D_{i-1}) \right)$$

$$\boxed{\Phi(D_n)} \cancel{- \Phi(D_{n-1})} \cancel{+ \Phi(D_{n-1})} \cancel{- \Phi(D_{n-2})} \cancel{+ \Phi(D_{n-2})} - \cancel{\Phi(D_{n-3})} + \cancel{\Phi(D_{n-3})} + \cdots + \cancel{\Phi(D_2)} \cancel{- \Phi(D_1)} \cancel{+ \Phi(D_1)} \boxed{- \Phi(D_0)}$$

$$= \Phi(D_n) - \Phi(D_0)$$

$$\therefore \sum_{i=1}^{n} a_{c_i} = \sum_{i=1}^{n} t_{c_i} + \Phi(D_n) - \Phi(D_0)$$

b) $a_{c_i} = t_{c_i} + \Phi(D_i) - \Phi(D_{i-1})$

$t_{c_i} = O(1)$ for insert as it is $O(1)$ work to insert into root list and update some pointers

$a_{c_i} = O(1) + T(H) - (T(H) + 1)$
$= O(1) - 1$ → remove constants for big O complexity
$= O(1)$

c) Proving true cost $= O(D_{max}(N) + T(H))$

Extract min has two components to it:

- Extract the min value
- Consolidating the heap.

Extracting the minimum is $O(D_{max}(N))$. The minimum is removed from the root list and its children are promoted which involves changing pointers. This would be $O(D_{max}(N))$ at worst case when there the min node has $D_{max}(N)$ number of children.

Consolidating is $O(D_{max}(N) + T(H))$ work. Each tree in the root list must be accessed which is $O(T(H))$. These trees will be merged and there are max $D_{max}(N)$ merges.

$$t_{c_i} = O(D_{max}(N)) + O(D_{max}(N) + T(H))$$
$$= O(D_{max}(N) + T(H))$$

$$a_{c_i} = O(T(H) + D_{max}(N)) + m\underbrace{[\Phi(D_i) - \Phi(D_{i-1})]}_{T(H) - T(H) - 1}$$

$$= -1$$

$$a_{c_i} = O(T(H) + D_{max}(N) + m(-1))$$

Let $m = T(H)$

$$a_{c_i} = O(T(H)) + O(D_{max}(N) - T(H))$$
$$= O(D_{max}(N))$$

d) Each tree in the heap is at worst a binomial tree or a binomial tree that has lost nodes. Losing nodes will not increase the degree. Each tree has order k which means it's root will have k children at worst case. Each tree can have up to 2^k nodes.

$$N = 2^k$$

$$\log_2 N = k$$

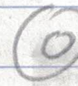
Therefore $D_{max}(N) \leq \lfloor \log_2(N) \rfloor$ as $k \leq \lfloor \log_2(N) \rfloor$

e) Number of nodes for a maximally decreased tree of size $k = F_{k+2}$

f) Statement - S: Maximally decreased tree of degree is F_{k+2} for $k \geq 0$

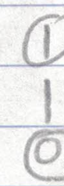
Base case:

$k = 0$

Maximally decreased tree of degree 0 : 

$F_{0+2} = F_2 = 1$ ∴ Holds for $S(0)$

$k = 1$

Maximally decreased tree of degree 1 : 

$F_{1+2} = F_3 = 2$ ∴ Holds for $S(1)$

Inductive hypothesis: Assume $S(k)$ is true for all values up to and including k for some $k \geq 1$.

Inductive step: Prove S is true for $S(k+1)$

When going from a binomial tree of degree k to $k+1$, the new child of the root will be a binomial tree of size B_k. Under the B_{k+1} tree, there are binomial trees of size B_0 to B_k. The size of each tree increases when the degree increases. For each subtree (not the tree overall), the root can only lose one child when maximally decreasing otherwise it would get cut. When the B_k tree is added, the B_{k+1} tree must lose its biggest child which is the B_k tree. This in essence turns the B_k subtree into a B_{k-1} tree. When the B_{k-1} tree is maximally decreased, it will have F_{k+1} nodes based on the inductive hypothesis. The B_k tree has F_{k+2} nodes based on the inductive hypothesis. As explained above, the B_{k+1} tree is the $B_k + B_{k-1}$ tree which is $F_{k+2} + F_{k+1}$ which equals F_{k+3}. Therefore the number of nodes in the $k+1$ tree is $F_{(k+1)+2}$ and S is true for $S(k+1)$ for all $k \geq 0$.

g) $\text{size}(r) \geq g(k)$. $g(k) = F_{k+2}$ as the number of nodes in a maximally decreased tree of degree k is F_{k+2} as was proven in f).

$$\text{size}(r) \geq F_{k+2} \geq \phi^k$$

Let $n = \text{size}(r)$

$$n \geq F_{k+2} \geq \phi^k$$

$$n \geq \phi^k$$

$$\log_\phi n \geq k$$

k is $D_{max}(N)$ as the max number of children is the degree of the tree. $D_{max}(N)$ is bounded by $\log_\phi n$

$$\therefore D_{max}(N) = O(\log(N))$$

h) $a_{c_i} = t_{c_i} + \Phi(D_i) - \Phi(D_{i-1})$

$t_{c_i} = O(1) + O(c) = O(c)$

$O(1)$ to move to root list $O(c)$ cascading cuts

$a_{c_i} = O(c) + \Phi(D_i) - \Phi(D_{i-1})$

$\Phi(D_i) = T(H) + 1 + c$ — add at least one node + how many were cascaded

$\Phi(D_{i-1}) = T(H)$

$$
\begin{aligned}
a_{c_i} &= O(c) + T(H) + 1 + c - T(H) \\
&= O(c) + 1 + c \\
&= O(c).
\end{aligned}
$$

Not expected to get $O(c)$ as decrease key should have an $O(1)$ amortized complexity.

i)

$$
\begin{aligned}
a_{c_i} &= t_{c_i} + m\,[\Phi(D_i) - \Phi(D_{i-1})] \\
&= O(c) + m\,[T(H) + c + 1 + \alpha(-c + M(H) + 1) - T(H) - \alpha M(H)] \\
&= O(c) + m\,[\cancel{T(H)} + c + 1 - \alpha c + \cancel{\alpha M(H)} + \alpha - \cancel{T(H)} - \cancel{\alpha M(H)}] \\
&= O(c) + m\,[c + 1 - \alpha c + \alpha]
\end{aligned}
$$

Let $\alpha = 2$

$$
\begin{aligned}
&= O(c) + m\,[c + 1 - 2c + 2] \\
&= O(c) + m\,[-c + 3]
\end{aligned}
$$

Scale m such that $mc = O(c)$

$$
\begin{aligned}
&= O(c) - mc + 3m \text{ — ignore constants} \\
&= O(c) - O(c) \\
&= O(1)
\end{aligned}
$$

j) $\alpha = 2$ makes sense because once a node is marked twice, it is promoted and is in the root list