Name: Wong Cheok Foong

Student id: 29801028

a) $\sum_{i=1}^{n} \mathrm{aci} = \sum_{i=1}^{n} tci + \Phi(\mathrm{Di}) - \Phi(\mathrm{Di}-1)$

$= \sum_{i=1}^{n} tci + \Phi(\mathrm{Dn}) - \Phi(\mathrm{D0})$

b) Φ(Di) − Φ(Di+1) = T(H) – T(H)+1 = 1
aci = tci + Φ(Di) − Φ(Di+1)
= O(1) + 1
= O(1)

c) The true complexity would be when the minimum is at the tree with the biggest order. The children of the tree with the biggest order O(Dmax(N)) will now be in the root list after extracting the parent node and this will increase the number of nodes in the root list and therefore the potential increases when from D_(i) to D_(i-1). The size of the root list when performing consolidate will be at most T(H) + Dmax(N)). For each while loop, the root will be linked to another and therefore the true cost will be O(T(H) + Dmax(N)). The potential is T(H) + 2m(H) before extracting the minimum node and the potential (Dmax(N) +1 + 2m(H)) at most afterwards. As no nodes will be marked during the operation and Dmax(N) + 1 roots remain, then the armotized cost will be

O(Dmax(N) + T(H)) + ((Dmax(N)+ 1) + 2m(H)) - (T(H) + 2m(H))
= O(Dmax(N)) + O(T(H)) - T(H)
= O(Dmax(N))

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

d) Dmax(N) ≤ log2 (N). This is because as we insert node we would have to perform merge until each tree is unique and therefore the number of children for the minimum of the biggest order would most of the time be less than log2(N). The times when Dmax(N) = log2(N) is when for example we insert N=3 the first 3 nodes and there will be T0 and T1 where T0 is a tree with only 1 node and T1 is the tree with two nodes and when T1 is the minimum as the biggest order, it has only 1 children for N= 3. Therefore, the example given is when Dmax(N) = log2(N).

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

e) Fibonacci sequence
F0 = 0, F1 = 1 (base case),
Fn = Fn−1 + Fn−2 for n > 1

For all integers k >=0
Expression: $1 + \sum_{i=0}^{k} Fi$

From the above expression we can derived the number of nodes that are in a maximally decreased tree of degree k.

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

f) By using induction to prove the above expression is correct. To prove the expression the below equation is formed.
F(k+2) = 1 + $\sum_{i=0}^{k} Fi$, where k >= 0

Base step: when k =0, the left hand side is equal to right hand side
1 + $\sum_{i=0}^{0} Fi$ = 1 + F0
= 1 + 0
= 1
= F2
Induction step
Now we assume that k-1 is true which is F(k+1) is true, so we assume the following is correct
F(k+1) = 1 + $\sum_{i=0}^{k-1} Fi$

Next we can prove that F(k+2) is also true by first applying the recurrence relation of Fibonacci numbers and substitute F(k+1) from the previous equation.
F(k + 2) = Fk + F(k+1)
= Fk + $(1 + \sum_{i=0}^{k-1} Fi)$
= 1 + $\sum_{i=0}^{k} Fi$

Therefore, the result is true for F(k+2). Thus the statement is true for all integers k >= 0.

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

g) Equation 1 = size(r) >= g(k)
Equation 2 = F(k+2) >= ϕ k
From the equation size(r) >= g(k) where g(k) is the number nodes in a maximally decreased tree. Since the above equation F(k+2) can determine the number of nodes that are in a maximally decreased tree of degree k. Then we can let,

g(k) == F(k+2)

Therefore we can have the overall equation is

Size(r) >= F(k+2) >= ϕ k

To prove that O(Dmax(N)) = O(log(N)) even when decrease-key is allowed, we assign x to be a node in a Fibonacci heap of N-node and we let k = degree[x]. We will then substitute N as size(r) and we will have the equation
N >= F(k+2) >= ϕ k

We then take ϕ as a base logarithm where we will have k >= logϕN. Because k is an integer, therefore k <= log ϕN. This concludes that O(Dmax(N)) = O(log(N)).

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

h) We know that for each decrease-key operation takes O(1) time and now assuming that the number of cascading cuts is more than 0 and each since each cascading cut is also O(1) time, the total operation for all cascading cuts is O(c). The total cost of decrease key operation is now O(c) with the repetition call of cascading cuts. Let T(H) be the current state for the number of nodes in the root list of Fibonacci heap H before any of the decrease key operation. For each repetition of cascading cut operation excluding the last one will cut the marked node and promote it to the root. Thus, the current state of the root list is now T(H) + c where c is the number of cascading cuts performed. Each cascading cuts unmarks a node where the last cascading cuts potentially marks a node. Let m(H) be the number of marked nodes in heap H. So the max number of marked node will be m(H) – c + 2. The change in potential is then

$$\Delta\Phi = ((t(H) + c) + 2(m(H) - c + 2)) - (t(H) + 2m(H))$$
$$= 4 - c$$

The armortised cost of decrease key is

$$O(c) + 4 - c = O(1)$$

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)

i) The equation will be
$\Phi = T(H) + \alpha M(H)$ where a = 2.

j) When calling cascading cut, we assume that one unit of potential can pay for the work where the marked node is cut and clearing off the mark of that node. Then the other unit will compensate for the it when the node becomes a root.

Citation: (Cormen, T., Leiserson, C. & Rivest, R, n.d.)
(Mayr, E. & Racke, Harald, n.d.)

References

1. "Amortized Analysis". (2012). Design and Analysis of Algorithms. *MIT OpenCourseWare.* Retrieved from https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2012/lecture-notes/MIT6_046JS12_lec11.pdf

2. Cormen, T., Leiserson, C. & Rivest, R. (n.d.). CHAPTER 21: FIBONACCI HEAPS. Retrieved from http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap21.htm
3. Mayr, E. & Racke, Harald. (n.d.). 8.3 Fibonacci Heaps. Retrieved from https://www.cl.cam.ac.uk/teaching/1415/Algorithms/fib2.pdf