CSC265 Fall 2020 Homework Assignment 7

Solutions

The list of people with whom I discussed this homework assignment: Chen Jiawei

A **black&white heap** is a doubly-linked list of binomial trees that satisfies the following properties:

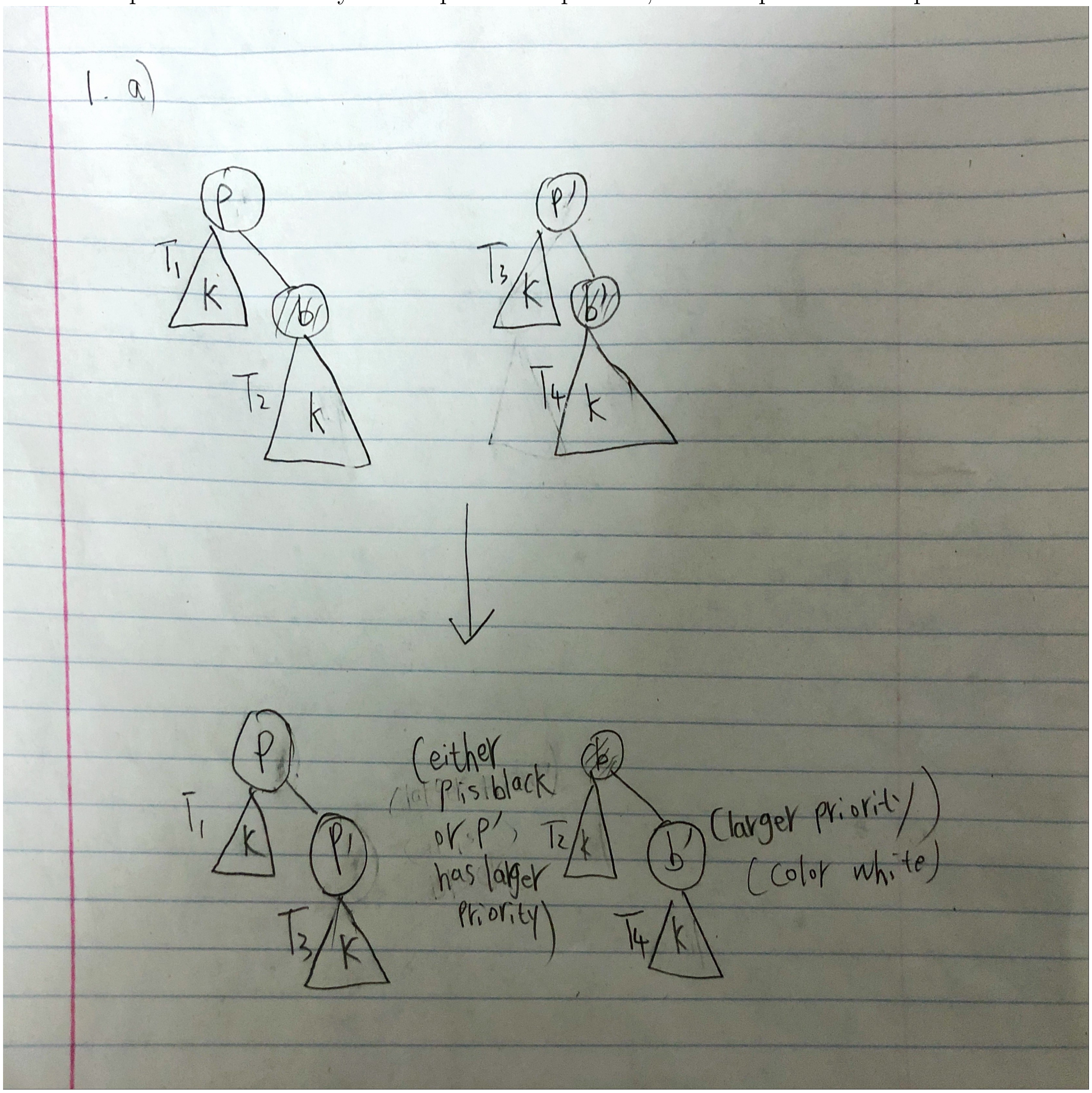
- The roots of the binomial trees in the list have strictly increasing degrees.
- Each node in the binomial trees is either white or black.
- The root of every binomial tree is white.
- If a white node is not the root of a binomial tree, its key is greater than or equal to the key of its parent.
- The black nodes all have different degrees.
- The degree of a black node is 1 less than the degree of its parent, i.e. the black node is the first child of its parent.

Each node in a black&white heap has pointers to its preceding sibling as well as its next sibling, provided they exist. If not, the pointers are NIL.

1. Suppose you have a linked-list H of binomial trees that satisfies all the properties of a black&white heap except that it has two black nodes b and b' of degree k both of which have parents of degree $k+1$. Given pointers to b and b', explain how to transform H in constant time so that it is a black&white heap with the same set of nodes except that, for some $k' > k$, it might have two black nodes of degree k' and it might have one black node of degree k' that has a parent of degree greater than $k'+1$.

   Solution:
   Note: for every solution from 1-4, the time is constant because they are all composed of a finite amount of compare, swap, and recolor operations, all of them taking constant time. The graphs represents most common cases in each question, the special cases are mentioned in the explanation without drawing separate graphs.

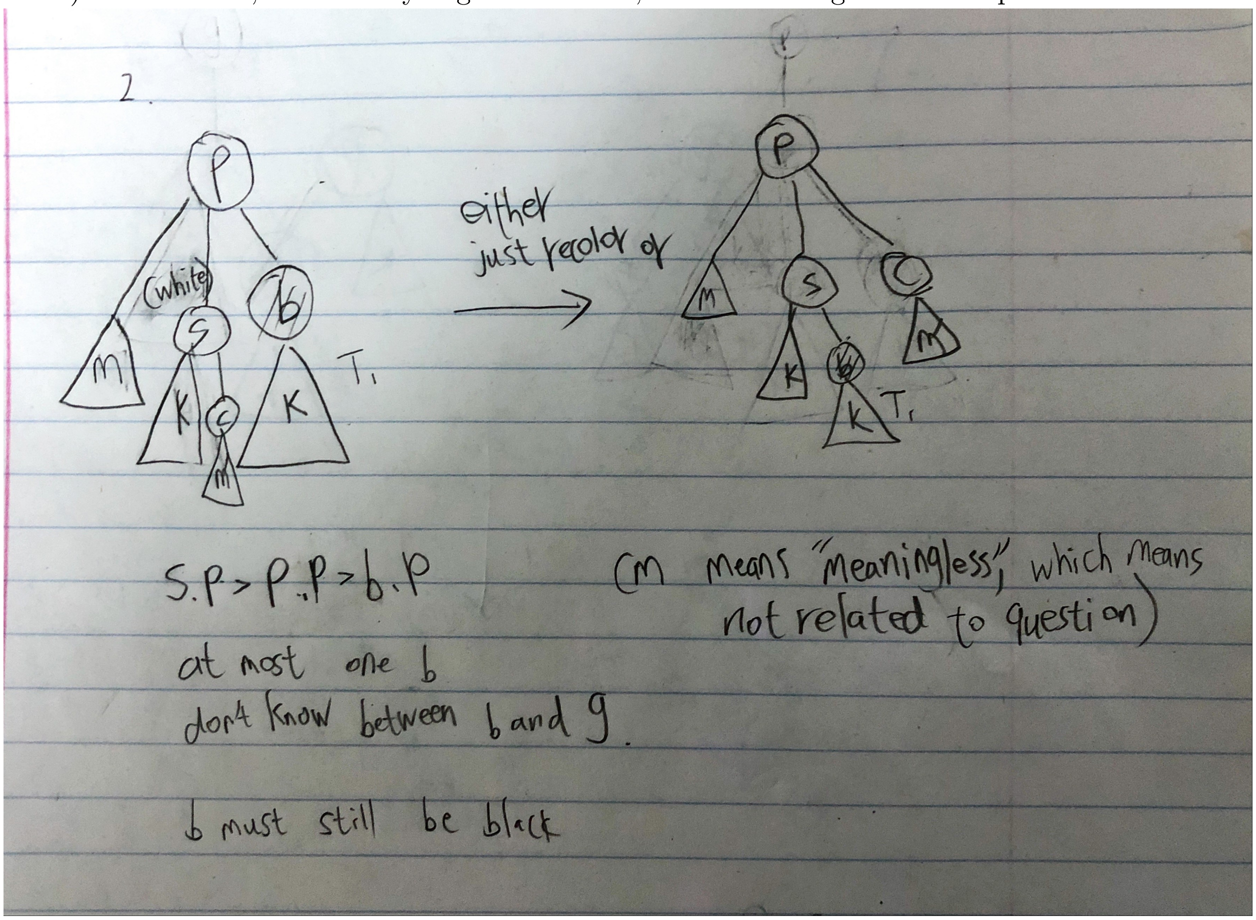
   Name the parents of b and b' as p and p'. They can't both be black, so at most 1 is black. First consider the cases where they are both white. Consider the key of their parents, pick the one with greater key. WLOG assume it is p'. Then, store their siblings in temporary variables, swap p with either b or b' which is not its child (in this case b) by swapping their siblings and parents. Consider the key of b and b'. Pick the one with less key, WLOG assume it is b, swap (swap their parents, keep their color and children) so that the one with less value is the parent of the other, and color the children white (legit since its key is larger). In this way, it might have 2 black nodes of degree k+1 since b originally has k degree plus one from b', and there might be another black k+1 node elsewhere, and the parent of b might have degree of k+2 or more. Consider p and p'. Since p' has greater key than p, and both maintained the same degree, it is legit no matter they are black or white. Then consider the case where one of them is black. WLOG assume it is p'. Swap the other one with the one among b and b' with less key, and now it has k degree and its parent has k+1 degree, and the other nodes are legit since they are in the same condition as the other case. Note that the color white part is not necessary for this particular question, but it helps us to solve question 5.



2. Suppose you have a linked-list H of binomial trees that satisfies all the properties of a black&white heap except that it has exactly one black node b of degree k and its parent has degree greater than $k + 1$. Suppose that b's sibling of degree $k + 1$ is white. Given a pointer to b, explain how to transform H in constant time into black&white heap with the same set of nodes.

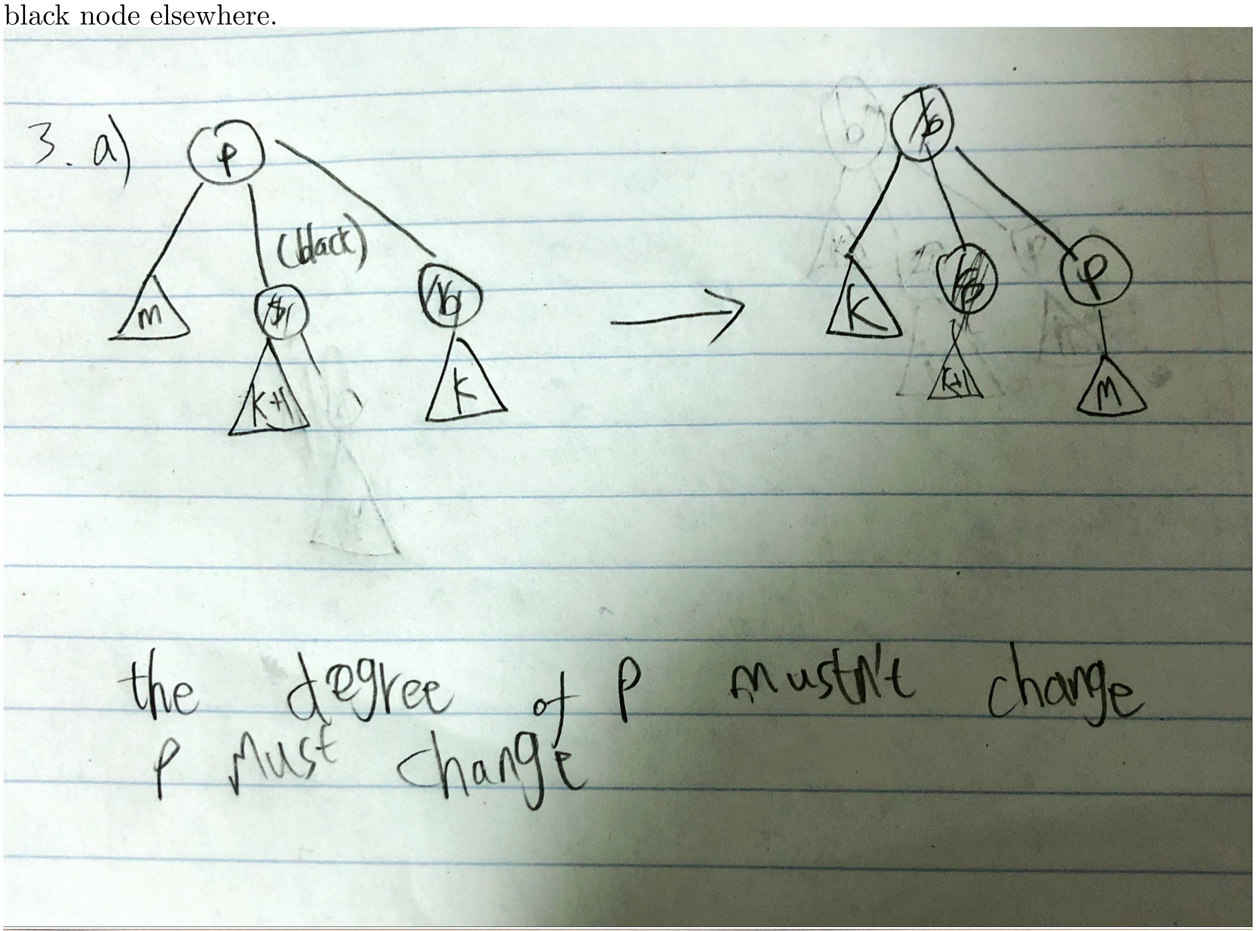
Solution: Consider the black node's parent, name it p, and its white sibling, name it s.

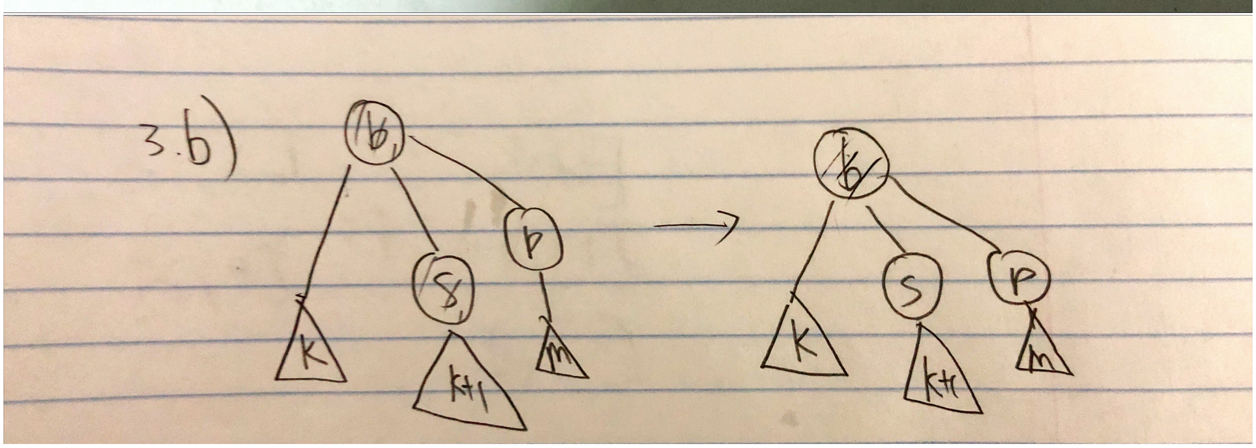
Assume that p has m degrees besides s and b. Compare the key of b and p. If b has higher or equal key than p, then color it white, and we are done. else, since s is white, $s.key > p.key > b.key$. Name the first child of s to be c. swap c and b, keeping their own children and color, swap their siblings and parent. After the swap, since s has k+1 degree(1 more than b) and both c and b have the same degree, it is now a black-white heap (if c is black). if c is white, then its key is greater than s, and therefore greater than p.



3. Suppose you have a linked-list H of binomial trees that satisfies all the properties of a black&white heap except that it has exactly one black node b of degree k and its parent has degree greater than $k+1$. Suppose that b's sibling of degree $k+1$ of b is black. Given a pointer to b, show how to transform H in constant time so that it is a black&white heap with the same set of nodes except that, for some $k' > k$, it might have two black nodes of degree k' and it might have one black node of degree k' that has a parent of degree greater than $k'+1$.

   Solution: Consider the black node's parent, name it p, and its white sibling, name it s. Assume that p has m degrees besides s and b. Compare the key of b and p. If b has higher or equal key than p, then color it white, and we are done. else, consider p. There are Two cases: p is black or p is white. If p is white, since it is greater than b.key, swap p and b, maintain their original children and swap their parents and siblings. except that now s is a child of b. Now, b has k+2 degree, s has k+1 degree and is its black children, p has greater key than b. This is picture 3a), and we are not done yet. Find the node between b and s with the smaller key, swap them so that the node with smaller key is the child, and color it white, as indicated in picture 3b). Now consider the case that p is black. Do the same swapping as the other case, except that color p white. s and the original children of b will maintain their black-white heap properties, and since p has greater key than b and have its original children, it is also maintained. In both cases, b will have k+2 degree which might collide with another black node elsewhere.

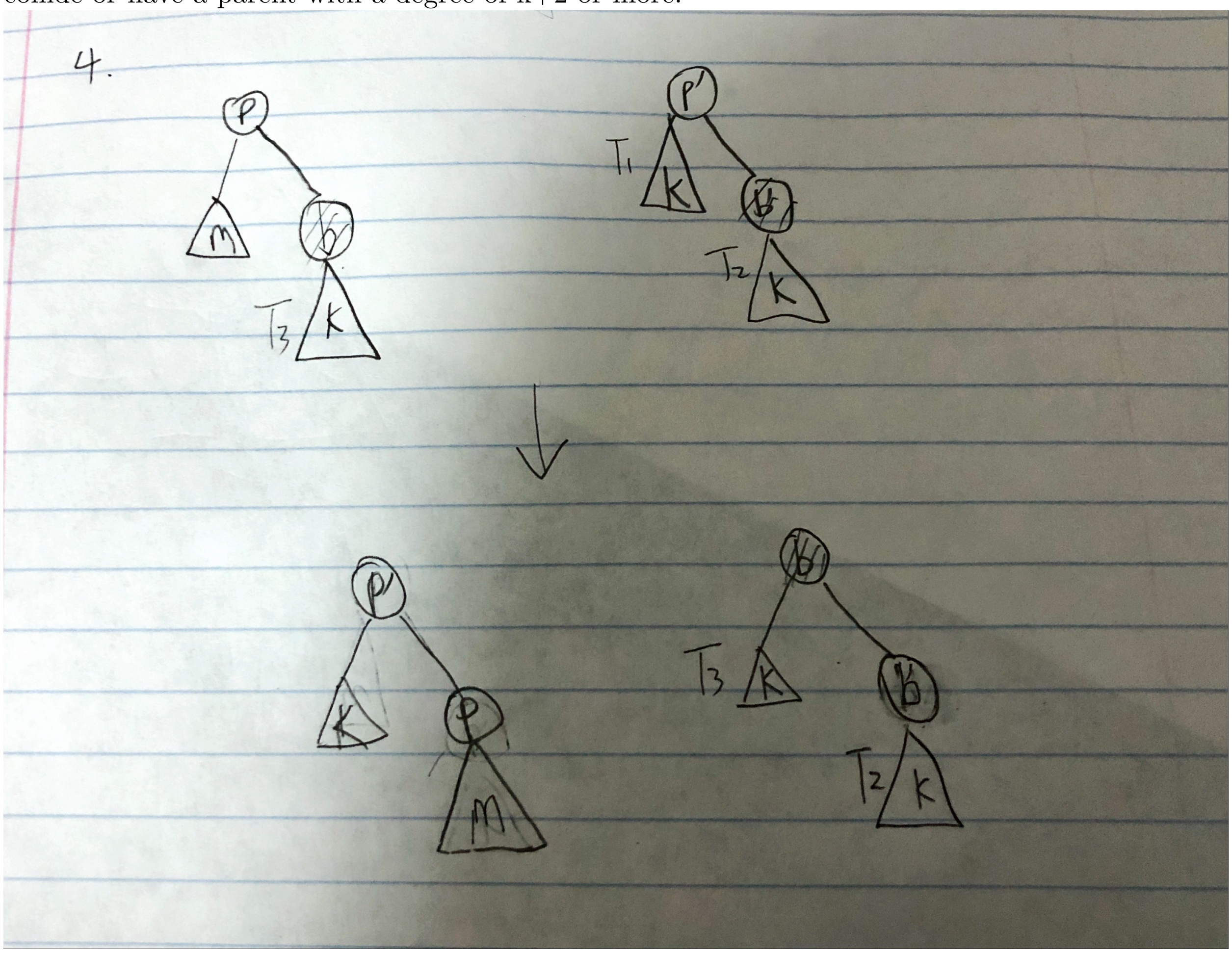




4. Suppose you have a linked-list H of binomial trees that satisfies all the properties of a

black&white heap except that it has one black node b of degree k whose parent p has degree greater than $k+1$ and another black node b' of degree k whose parent has degree $k+1$. Given pointers to b and b', explain how to transform H in constant time so that it is a black&white heap with the same set of nodes except that, for some $k' > k$, it might have two black nodes of degree k' and it might have one black node of degree k' that has a parent of degree greater than $k' + 1$.

Solution: Similar to question 1, If any black node has greater key than parent then just color it white and it's done. Else, first swap one black node with the parent of the other one, and then rearrange both the two parents and the two black nodes. For the two parents, the one parent with degree larger than k'+1 might be black, if this is the case then compare it with the other parent, if it is smaller then keep it as is, else swap the parents and color the child one white. If both are white then swap them so that the larger key is the child. For the black nodes, swap them (and their children and change siblings) so that the one with the larger key is the child, color it white. Now the remaining black node has its own children, which is k degrees, plus one from the other black node, so it's degree is now k+1, which might collide or have a parent with a degree of k+2 or more.



5. The operation Decrease-Key(H, x, v) takes a node x in a black&white heap H and a value v and decreases the priority of x by v. Explain how to perform Decrease-Key in

a black&white heap so that its amortized cost is $O(1)$ with respect to a potential function that is some constant times the number of black nodes in the black&white heap.

Solution: Assume that swap,recolor and compare each consumes 1 potential. Then, look at questions 1-4, we find that for the operations of each question, they consume at most 7 potential, and at the end of each question, the number of black nodes decreased by 1. Define the potential function to be 7 times the number of black nodes. Decrease x's priority by v, if it still has greater key than parent then we are done, else color it black. Now, assume that x has a degree of k. This x might violate the constraints since there might be another k degree black node and that its parent might have degrees greater than k+1. Now, if there are two k degree black nodes and x's parent is k+1, then it is question 1, if x's parent has more degrees it is question 4, if there is no repeated black node but x's parent is more than k+1 degrees it is case 3. In all cases, after constant time, the exception of the heap is shifted upwards by at least 1 level with degree of the black node causing exception increased by at least 1. Also, more importantly, the total number of black nodes decreased by 1. Since the operation took at most 7 step, the amortized time is 0 (paid by the potential of the deleted black node). This process repeats at most n times, where n is the number of the black nodes, since before x is colored black, each black node has a unique degree, and this collision of degrees can happen at most n times. When there is no collision, it is case 2, and it takes only constant time to make the black white heap normal.