Part A (Klevis Tefa)

Question A.1

|  |  |  |  |
| --- | --- | --- | --- |
| Target | **Linear Search** | **Improved Linear Search** | **Binary Search** |
| Algorithm | 1 | 1 | 4 |
| Computer | 3 | 3 | 4 |
| heap | 8 | 8 | 1 |
| int | 9 | 9 | 4 |
| open | 15 | 15 | 4 |
| bit | 15 | 2 | 4 |
| stack | 15 | 15 | 4 |
| queue | 15 | 15 | 4 |
| n = 100 | 100 | 100 | 7 |
| n = 1000 | 1000 | 1000 | 10 |
| n = 106 | 106 | 106 | 20 |
| n = 109 | 109 | 109 | 30 |

Question A.2

1. 6 different almost level trees.
2. Let L be the number of leaves, N the number of internal nodes, and T the total number of nodes in a full binary tree. Since each node is either an internal node or a leaf in full binary tree than it’s trivial that T = N + L. From observing the structure of different full binary trees I came to the relation that L = N + 1. We can prove this by induction.  
   **Proof:** Let S = N (set of all integers ≥ 0) such that if a tree is a full binary tree with I internal nodes than L = N + 1.  
   (Base case): If N = 0, then the tree has only the root with no children since the tree is full. Hence there is only one leaf (i.e L = N + 1 = 0 + 1 = 1).

Now suppose for some integer K ≥ 0, every N from 0 through K is in S. (i.e: if a nonempty binary tree has N internal nodes (0 ≤ N ≤ K), then that tree has N + 1 leaves.  
Let’s have a tree with K + 1 internal nodes. Then the root of that tree will have two subtrees L and R, and suppose L has NL internal nodes and R has NR internal nodes (neither L nor R can be empty). So every internal node in L and R is an internal node in our original tree plus the root of the tree itself. Hence K + 1 = NL + NR + 1.   
Now by induction hypothesis, L must have NL + 1 leaves, and R must have NR + 1 leaves. Since every leaf in our original tree is either in L or in R we have a total of NL + NR + leaves.  
Therefore we must have K + 2 leaf nodes so K + 1 is in S. Hence by mathematical induction S = [0, ∞).

Since we proved that L = N + 1 (or N = L – 1) and we know that T = L + N (or T = L + L – 1 = 2L – 1).

Therefore 99 = 2L – 1 which means that L = 50. So we have 50 leaf nodes.

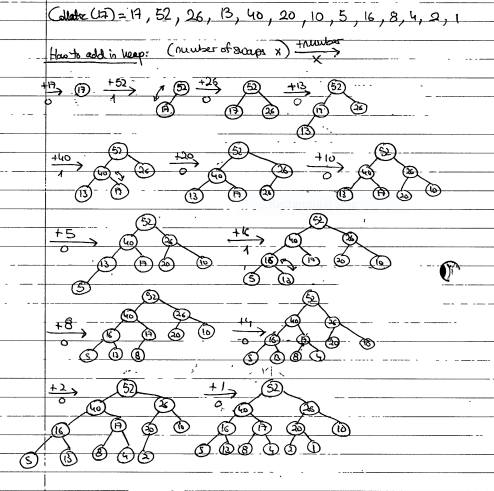
1. Formula for a geometric series of the form:

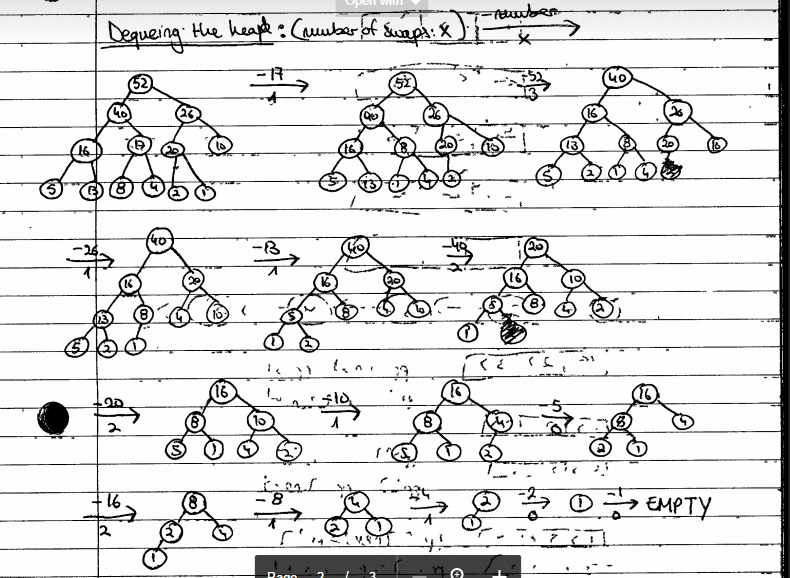
In our case , , and . Therefore our required sum is

=

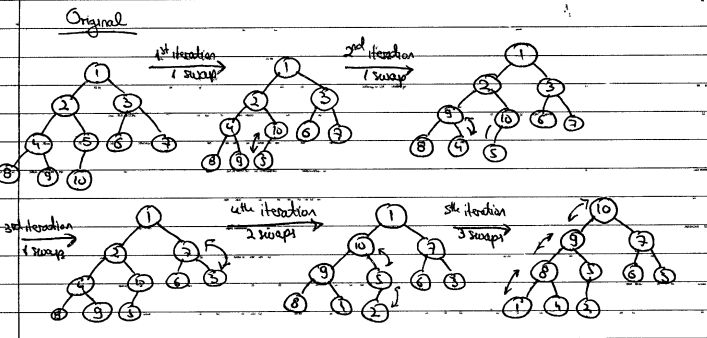
1. According to Lecture 11, at level i, we have 2i nodes. In the last level i = height – 1. In order to find height of a complete binary tree of 1000 nodes we can use the formula from lecture 11, where Hence, . So at the last level i = 8, therefore there are 28 = 256 nodes.

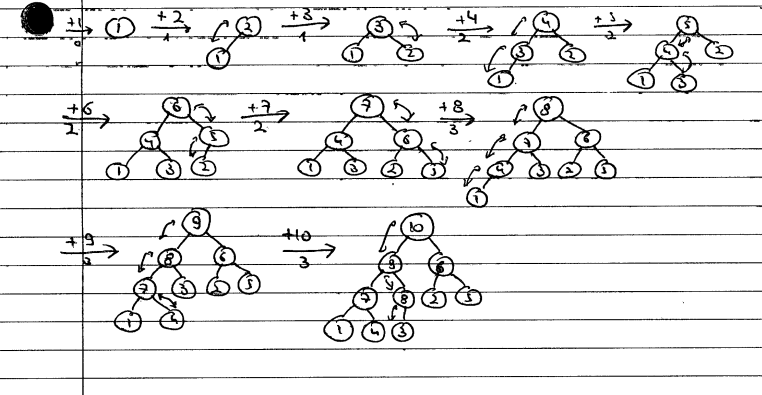
Question A.3





Question A.4





c) Total number of swaps in part a) was 8 and in part b) it was 19. For this particular heap of size 10 the code in part a) is better since we have to do less swaps.