1. A skip list is originally empty ant then the keys are inserted into the list one by one in the given order.

The numbers in the parentheses denote the level of the corresponding node given by the numer of coint tosees.

Each time, the coin is being tossed until it comes up tails. Draw the resulting list.

16(3) 23(2) 18(2) 5(2) 15(1) 19(1) 33(1) 11(2) 21(2) 4(1) 22(2) 6(2) 17(4) 10(1) 9(1) 28(4)

**Solution:**

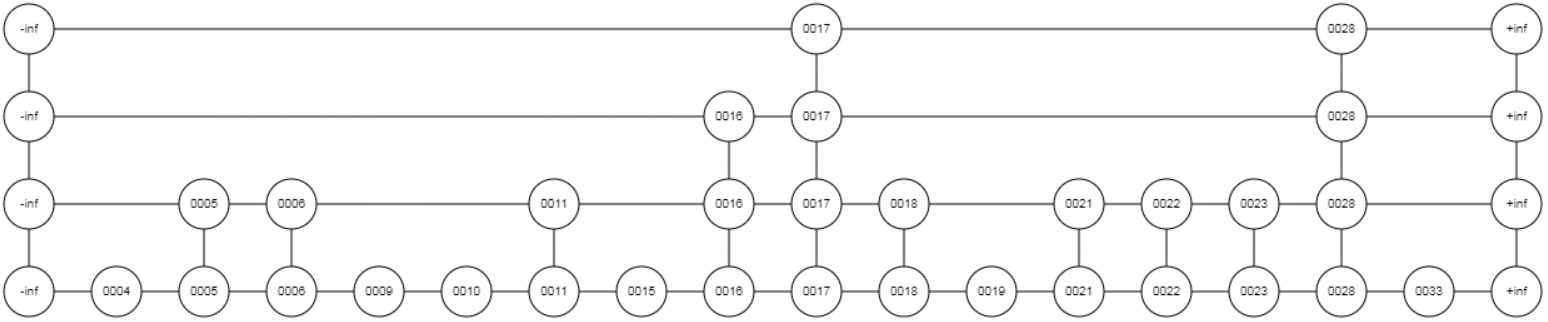
A skip list and the inserts can be visualized here: <https://people.ok.ubc.ca/ylucet/DS/SkipList.html>

Coin flips has to be set to:

1101010100001010010101110001110

Then the inserts can be entered one by one.

This is the resulting list:



2. There are two skip lists of length N. Describe an effective algorithm which merges these two skip lists into one skip list of length 2N. What is the asymptotic complexity of your solution?

**Solution:**

Let the first skip list be denoted A and the second skip list be denoted B. Let A[i] or B[i] denote the i-th element of A or B, respectively (where i=1,...,N). Assume that A[1]<=B[1] (if not let then A denote the second list and B the first list). We describe an algorithm which merges elements of B into A in O(n) time. Use indices pA and pB pointing to elements of A and B, respectively. Set them initially to 1. Repeat two following procedure: If A[pA+1]<B[pB], increase pA by 1, otherwise insert B[pB] into A right after the element A[pA] and increase pA as well as pB by 1 (this means, pA will point to the newly inserted element). Do this procedure until all elements of B were inserted into A.

3. There is a skip list of length N. Split the list into two separate skip lists, first of which will contain only the odd keys of the original list and the second one will contain only the even keys of the original skip list. Can this task be accomplished in time Θ(N)?

**Solution:**

This can be considered as a reversed operation to the merge described in the previous assignement. It can be accomplished in Θ(N) time. Let the list be denoted A. Create a new list B for e.g. all the odd elements. Go through the elements of A in their order. Whenever there is an odd element, remove it from A and insert it as the last element of B.

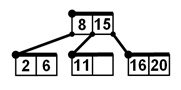
4. Is it possible to reverse the order of the keys in a skip list in time which is asymptotically less than Θ(N ∙ log(N))?

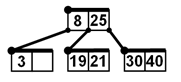
5. Describe an effective operation IncreaseKey and DecreaseKey in a skip list. The ammount by which the key value is increased/decreased will be the parameter of the operation.

6. Describe how to use a skip list as a priority queue. Compare the complexity of operations Insert and ExtractMin to the complexity of the same operations in a binary heap. Which of these two data structures performs asymptotically better?

7. The rule governing the choice of the level value of a node in a skip list were experimentally modified. The new rule says, that the choice of level value of each node is randomly uniformly distributed on an interval <1, ⎡log2N⎤ >.

N is the actual size of the list. There is a hypothesis that Insert and Delete will perform asymptotically slower in this list than they do in a standard skip list. Confirm or disapprove the hypothesis.

****8. There is a discussion about the sum of level values of all nodes in a skip list of length N. Professor Highman says that the sum is on average proportional to N ∙ log(N). Professor Lowman says that the sum is on average proportional to just N. Professor Middleman says that both cases might happen and that it depends on the data. Decide whether any of the three professors is right.

****

11. Insert keys 14 and 10 into the left B-tree and insert keys 7, 5 into the right B-tree. What will be the contents of the root after the insertion?

**Solution:**

The left B-tree: Insert(14) puts key 14 to the only empty cell. Insert(10) tries to put 10 in the same node as well, however it is already full. From this reason, the node is split into two nodes where the first one stores key 10, the second one stores key 14 and key 11, which is the median of the sequnce 10, 11, 14, is inserted into the parent node. The parent (which is the root) is also full. It splits into two nodes. The first one stores key, the second one stores key 15. A new root is created and it stores key 11, which is the median of the sequnce 8, 11, 15.

The right B-tree: The process is similar. The root contains key 8.

12. Suppose that a B-tree of order 1 is originally empty. Insert, in the given order, the given keys into the tree

25, 13, 37, 32, 40, 20, 22.

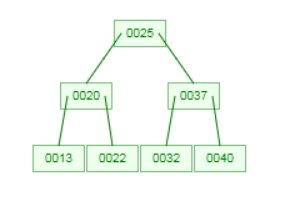
Draw the tree after each insertion. Next, delete the keys from the resulting B-tree in the order:

13, 25, 40, 22, 20, 37, 32. Draw the tree after each deletion.

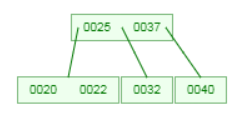
**Solutions:**

Operations over a B-tree can be simulated in this application: <https://www.cs.usfca.edu/~galles/visualization/BTree.html>

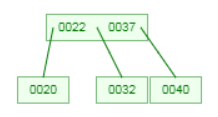
B-tree after the inserts:



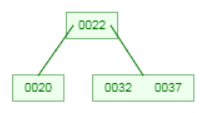
After delete(13):



After delete(25):



After delete (40):



etc.

13. Two empty B-trees of order 1 (max 2 keys in a node) are isomorphic. Let T1 and T2 be two unempty B-trees with the respective roots R1 and R2. T1 and T2 are isomorphic iff both 1. and 2. holds:

1. The root of T1 contains the same number of keys as the root of T2.

2. The left subtree of R1 is isomorphic to the left subtree of R2, the right subtree of R1 is isomorphic

to the rightsubtree of R2 and the middle subtree (if it exists) of R1 is isomorphic to the middle subtree of R2.

What is the number of non-isomorphic B-trees with A) 0, B) 1, C) 3, D) 4, E) 7 nodes?

14. Extend your solution to the previous problem and find a general recursive formula which specifies the number of non-isomorphic B-trees of order 1 for any given number of nodes in the tree.

15. Suppose that a B+ tree of order 1 is originally empty. Insert, in the given order, the given keys into the tree

32, 18, 31, 59, 20, 23, 24, 36, 60, 58, 15, 57.

Draw the tree after each insertion. Next, delete the keys from the resulting B+ tree in the order:

23, 31, 15, 24, 36, 20, 32, 18, 59, 60, 58, 57. Draw the tree after each deletion.

**Solution:**

Here is an application to visualize a B+ tree: <https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html>

16. We are given a) B-tree b) B+ tree. The order of the tree is k=10 and it contains exactly 100 000 keys. What is the maximum and minimum possible height of the tree? What is the maximum and minimum possible number of keys in the tree?

**Solution:**

a) Roughly said, the height of a B-tree is minimum possible if each its node if full, i.e., if it contains 2\*k=20 keys. Then each node has 2\*k+1 children. Let us derive the number of nodes in such a tree of height H.

The root level contains 1 node, the next level contains 21 nodes, etc. In general, an i-th level (where i=0,1,...,H) has 21i nodes. In total there are nodes. If all the nodes are full, the tree contains keys. We have 100 000 keys. It must hold (we search for the smallest H fulfilling this inequality). We derive , hence . (Note that the ceiling means that not all the nodes will contain 20 keys).

The height of a B-tree is maximum possible if the root has 1 key and the other nodes are half-empty. In this case a B-tree of height H has 1+2+2\*11+2\*112+...+2\*11H-1 = nodes and they contain keys. We search for a minimum H fulfilling . We derive , hence (now we take the floor of the expression, some of the nodes may contain more than 10 keys).

b) If we have a B+ tree instead of B-tree, then we need to calculate how many leaves are there and how many keys they store (all the 100 000 keys are stored only in leaves).