1. Binary heap is stored in 1D array of keys. There is a simple formula describing how to calculate the indices of the left and right children of a given key when we know the index of the key. Write down this formula. Write a similar formula for a ternary heap and for a general *d*-ary heap.

2. A binary heap with minimum key in the root contains exactly *n*3 elements, where *n* is a positive integer. We have to remove *n*2∙lg(*n*) smallest elements from this heap. What is asymptotic complexity of this process? Will the complexity be different if we consider a binomial heap instead?

3. The depth of a *d*-ary heap is *h* and it is also the depth of all its leaves. Therefore, the number of leaves is *dh*. What is the minimum and maximum possible number of keys comparisons which are perfomed in this heap during one operation DeleteMin?

4. What is the maximum possible degree (= number of children) of a key in a binomial heap with N elements?

5. A Fibonacci heap is originally empty. Next, we insert 2*n* + 5 different keys (*n* > 2). Then we perform operation DeleteMin and consolidate the heap. How many binomial trees which have their roots in the root list of the heap will be there in the heap after the consolidation?

6. In general, the degree of a node in a binomial heap is not bounded from above, it can be arbitrarily big (providing the heap itself is sufficiently big). The children of the node are the roots of other binomial subtrees and the node stores the references to those roots in some structure S. There are two possibilities:

1. The references in S are stored in sorted order according to the order of the trees they refer to.

2. The references in S are stored in random order.

Consider how the speed of operations Insert and DeleteMin is affected by implementing either possibility1 or 2. Which one would you recommend and why?

7. Choose some particular data structure(s) which is, in your opinion, suitable for representing a binomial heap in computer memory. Show that your implementation preserves asymptotic complexity of operation Insert, AccessMin and DeleteMin. You can use more types of structures if you want to.

8. There are *n* (*n* ≥ 2) different integer keys and an empty binomial heap. We insert all keys into the heap in random order. What is the asymptotic complexity of this process? Is it possible that for some particular ordering of keys the complexity of inserting them all into the heap would be smaller?

9. A binomial heap keeps minimum values in the roots of its trees. The task is to find a maximum value in this heap and delete it from the heap. How fast (asymptotically) can the task be completed?

10. Let us suppose that binomial heap H is not completely filled, i.e. some of its binomial trees are empty. The number of unempty trees is *k.* How many leaves are there in the whole heap? Suppose H contains exactly *n* keys. What is the relation between *n* and the maximum possible value of *k*?