Design and Analysis of Algorithms

Practice-sheet : **Amortized Analysis and Fibonacci heaps**

1. **Deleting half elements**

   Design a data structure to support the following two operations for a dynamic multiset S of integers, which allows the duplicate values:

   - *Insert*(S, x): inserts x into S.
   - *Delete-Larger-Half*(S): delete the largest $\lceil |S|/2 \rceil$ elements from S.
   - *Report-max*: report the largest element from the set.

   Explain how to implement this data structure so that any sequence of m operations mentioned above run in $O(m)$ time. Your implementation should also include a way to output the elements of S in $O(|S|)$ time.

2. **Simulating a queue using stacks**

   Show how to implement a queue with two ordinary stacks so that amortized cost of each *Enqueue* and each *Dequeue* operation is $O(1)$.

3. **Computing π function of a pattern** In the KMP algorithm for pattern matching, we assumed that the π function associated with the pattern $P[1..m]$ was available. You have to design $O(m)$ time algorithm for computing π.

   Execute your algorithm to compute the π function for the pattern "ababbabbabbababbabb".

   Hint: There is a great amount of similarities between the π function and **Computing-F**(i) function discussed in the class.

4. **Multi-stack**

   A multistack consists of an infinite series of stacks S_0 , S_1, S_2 , ..., where the ith stack S_i can hold up to 3^i elements. Whenever a user attempts to push an element onto any full stack S_i , we first pop all the elements off S_i and push them onto stack S_{i+1} to make room. (Thus, if S_{i+1} is already full, we first recursively move all its members to S_{i+2}.) Moving a single element from one stack to the next takes O(1) time.

   (a) In the worst case, how long does it take to push one more element onto a multistack containing n elements?

   (b) Prove that the amortized cost of a push operation is $O(\log n)$, where n is the maximum number of elements in the multistack.

5. **cyclic pattern matching**

   Give a linear time algorithm to determine whether a text T is a cyclic rotation of another string T'. For example, *arc* and *car* are cyclic rotations of each other.

6. **A short and clean code for Decrease-key in Fibonacci Heap**

   Write a neat pseudocode for the Decrease-key(H, x) in a Fibonacci Heap ?

7. **Delete-key in a Fibonacci heap**

   Design an efficient algorithm for deleting an element from a Fibonacci Heap. The amortized cost must be $O(\log n)$.

8. **A surprising property for Fibonacci Heap**

   Let v be any node in a Fibonacci heap. We showed that if the size of the subtree rooted at v is m, then the degree of v is $O(\log m)$. Can we say the same thing about the height as well ? That is, will the height of v be bounded by $O(\log m)$? Note that all operations, including merging of Fibonacci heaps is allowed.
   Hint: There exists a sequence of operations that may result in a Fibonacci heap which will be a single tree that is just a vertical chain of m elements. Invent one such sequence.