Homework 3, Individual Problem 9

Graded by: Group 7 (Susan Joseph, Qi Su, Bradford Hovinen)
Adopted from a solution by: Saku Kukkonen

Problem Statement

Josephus permutation

The ***Josephus problem*** is defined as follows. Suppose that n people are arranged in a circle and that we are given a positive integer $m \leq n$. Beginning with a designated first person, we proceed around the circle, removing every mth person. After each person is removed, counting continues around the circle that remains. This process continues until all n people have been removed. The order in which the people are removed from the circle defines the ***(n,m)-Josephus permutation*** of the integers 1,2, ..., n. For example, the (7,3)-Josephus permutation is $\langle$ 3, 6, 2, 7, 5, 1, 4$\rangle$.

a.

Suppose that m is a constant. Descripe an $O(n)$-time algorithm that, given an integer n, outputs the (n,m)-Josephus permutation.

Solution

Assume that we have procedures LIST-SEARCH(L,k), LIST-INSERT(L,x), LIST-DELETE(L,x) and NEW-NODE(), which all work with circular linked lists. NEW-NODE() will create and return a new empty node.

JOSEPHUS(n)

```
1  L ← NIL
2  for i ← 1 to n do
3      node ← NEW-NODE()
4      key[node] ← i
5      LIST-INSERT(L, node)
7  p ← LIST-SEARCH(L, 1)
8  for i ← 1 to n do
9      for j ← 1 to m − 1 do
10         p ← next[p]
12     print key[p]
13     q ← next[p]
14     LIST-DELETE(L, p)
15     p ← q
```

The algorithm creates a circular linked list with n elements (lines 2–5). Key values of elements are in order 1, 2,..., n. The algorithm find the first element of the list (element which has key value 1) and starts to go through list printing key value of every mth node and deleting corresponding node, and repeats until n values has been printed (lines 8–15).

Third for-loop (lines 9–10) runs $m - 1$ times and moves pointer p to forward in every iteration. At the start, p points to the first node. If we want to get to the mth node, we have to move pointer $m-1$ times. After third loop key value of node pointed by p will be printed, node will be deleted and p will point to the next node after deleted node in the list. Second for-loop (lines 8–15) will be repeated n times.

Analysis

Everything inside first for-loop (lines 2–5) takes $O(1)$ time and loop will be repeated n times. Therefor creating circular linked list with n elements will take $O(n)$ time. List-Search at the line 7 will take in the worst case $\Theta(n)$ time. Second for-loop (lines 8–15) contains third for-loop (lines 9–10). Because m was a constant, this loop will take constant amount of time. Also List-Delete at line 14 will take $O(1)$ time. All other operation inside third for-loop will take $O(1)$ time and therefore time needed for one third for-loop iteration is $O(1)$. Whole third loop will be repeated n times and therefore overall time needed is $O(n)$. Overall time for whole algorithm is $O(n) + \Theta(n) + O(n) = O(n)$.

b.

Suppose that m is not a constant. Describe an $O(n \lg n)$-time algorithm that, given integers n and m, outputs the (n,m)-Josephus permutation.

Solution

Josephus(n,m)

Assume that we have procedures OS-Insert(T,x), OS-Select(x,i), OS-Delete(T,x) and New-Node(), which all work with order-statistic trees. New-Node() will create and return new empty node.

```
1  T ← NIL
2  for i ← 1 to n do
3      node ← NEW_NODE()
4      key[node] ← i
5      OS-Insert(T, node)
7  c ← 1
8  s ← n
9  for i ← 1 to n do
10     c ← c + m − 1
11     if c > s
12       then
13           c ← (c − 1)(mod s) + 1
15     node ← OS-Select(T, c)
16     print key[node]
17     OS-Delete(T, node)
18     s ← s − 1
```

Josephus(n,m)-algorithm creates order-statistic tree T which has n nodes and keys of nodes are 1, 2,. . . , n (lines 2– 5). After this algorithm starts to select nodes, print keys, and delete nodes according to Josephus permutation (lines 9–18).

Variable c stars with value 1 and at line 10 it will increased by $m-1$ to be m. Value of c represents rank value of node to be selected. Value of c is increased by $m-1$, because removing one node in every iteration decreases rank of the node to be selected next time by 1. At the line 11 algorithm checks if $c > s$. At the first iteration c can not be greater than s, because at the begining s eguals to n and $m \leq n$. Algorithm selects cth element (element which has cth smallest key value) from the tree T, prints key value of selected element and removes element from the tree T. At the end of for-loop, value of s will be changed to be current number of nodes in the tree T i.e. $s - 1$.

If value of c at the line 11 is greater than s this means that algorithm has reached rightmost node of tree T. Algorithm counts new value for c (line 13) using modulo-operator. Because (mod s) returns a value which is between 0 and $s - 1$, and value of c should be in the range 1 – s, 1 is subtracted from s before modulo-operation and 1 is added to the result. For example if value of c was two greater than s then $c - 1 (\mathrm{mod}\ s) + 1$ will return number two. For-loop will run n iterations. In each iteration it prints one value and removes one node, therefore all key values will be printed out.

Analysis

Inserting one node to order-statistic tree takes $O(\lg n)$ time at line 5. This will be repeated n times and therefore overall time needed creating order-statistic tree is $O(n \lg n)$. Everything inside the second for-loop (lines 9–18) takes constant amount of time except OS-Select at line 15 and OS-Delete at line 17. These both takes $O(\lg n)$ time. Whole for-loop will be repeated n times and therefore time needed for this part of algorithm is $O(n \lg n)$. Because both creating three and the second for-loop take $O(n \lg n)$ time, overall time needed for algorithm is $O(n \lg n)$.