ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2020

Homework 3

Due date: 23:59, September 28, 2020

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL NAME to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero grade.

1: (3'+2'+4')Hash Collisions

Consider a hash table consisting of $m = 10$ slots, and suppose we want to insert integer keys $A = [43, 23, 1, 0, 15, 31, 4, 7, 11, 3]$ orderly into the table.
First, let's suppose the hash function $h_1(k) = (11k + 4) \bmod 10$.
Collisions should be resolved via chaining. If there exists collision when inserting a key, the inserted key(collision) is stored at the end of a chain.

Question 1. *In the table below is the hash table implemented by array. Insert all the keys into the table below. If there exists a chain, please use $\rightarrow$ to represent a link between two keys.(3')*

Index	0	1	2	3	4	5	6	7	8	9
Keys										

Question 2. *What is the load factor λ?(2')*

Question 3. *Suppose the hash function is modified into*

$$h_2(k) = ((11k + 4) \ mod \ c) \ mod \ 10$$

for some positive integer c. Find the smallest value of c such that no collisions occur when inserting the keys from A. Please use some codes to provide the solution, including but not limited to: pseudocode, Python/C code, natural language, etc and insert all the keys into the table below.(4')

Index	0	1	2	3	4	5	6	7	8	9
Keys										

2: (3'+3'+5')Open Addressing

Use the following hash function for Question 4 and 5:

$$h(x) = (11x) \bmod 13$$

Question 4. *Insert the following keys into a size-13 hash table using open addressing and linear probing. Perform the insertions in the given order, one at a time. You just need to draw the final state of the hashtable after all insertions.(3')*

$$20, 56, 78, 9, 3, 45, 17, 169$$

Question 5. *Insert the following keys into a size-13 hash table using open addressing and quadratic probing. Perform the insertions in the given order, one at a time. You just need to draw the final state of the hashtable after all insertions.(3')*

$$20, 56, 78, 9, 3, 45, 17, 169$$

Question 6. *Suppose that we are using a hash table with m buckets, where $m > 1$. Recall that in open addressing we store an element in the first empty bucket among $A_0(k), A_1(k), \cdots, A_{m-1}(k)$. In this problem,*

$$A_i(k) = (h(k) + i^2 + i) \bmod m$$

where $h(k)$ is some hash function.
Prove that the probe sequence will always include at most $(m+1)/2$ distinct buckets.(5')

3: (3*1'+4'+3')Insertion Sort and Bubble Sort

The **question7-9** are multiple-choice questions, each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 0.5 point if you select a non-empty subset of the correct answers.
Note that you should write you answers of ***question1-3*** *in the table below.*

Question 7	Question 8	Question 9

Question 7. *In the lecture we have learnt that different sorting algorithms are suitable for different scenarios. Then which of the following options is/are suitable for insertion sort?*

(A) Each element of the array is close to its final sorted position.

(B) A big sorted array concatenated with a small sorted array

(C) An array where only few elements are not in its final sorted position.

(D) None of the above.

Question 8. *Which of the following statements about basic bubble sort without flag is/are true?*

(A) The maximum number of comparisons for an n-element array is $\frac{n(n+1)}{2}$.

(B) The time complexity is $O(nlogn)$.

(C) There are two for loops in the implementation, one nested in the other.

(D) None of the above.

Question 9. *The worst case time complexity for both insertion sort and bubble sort will be the same if: (assume bubble sort is flagged bubble sort)*

(A) the input array is already sorted.

(B) the input array is reversely sorted.

(C) the input array is a list containing n copies of the same number.

(D) None of the above.

Question 10. ***(4')Insertion Sort using Linked List***

In the lecture, we have learnt the insertion sort implementation using array. In this question, you are required to implement insertion sort using single linked list. Since it is not easy to traverse single linked list from back to front, we can traverse from front to back instead if it is needed.

Fill in the blanks to complete the algorithm. Please note that there is at most one statement (ended with ;) in each blank line.

```
\begin{enumerate}
\item
\end{enumerate}
struct ListNode
{
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(nullprt) {}
 };

ListNode* insertionSort(ListNode* head) {
    //if linked list is empty or only contains 1 element, return directly
    if(!head || !(head->next)){return head;}
    //create dummy node
    ListNode *dummy = new ListNode(-1);
    dummy->next = head;
    //split linked list into sorted list and unsorted list
    ListNode *tail = head;//tail of sorted list
    ListNode *sort = head->next;//head of unsorted list
    //insertion sort
    while(sort)
    {
        if(sort->val < tail->val)
        {
            ListNode *ptr = dummy;
            while(ptr->next->val < sort->val) ptr = ptr->next;
            ____________________________;
            ____________________________;
            ____________________________;
            ____________________________;
        }
        else
        {   //no need to insert
            tail = tail->next;
            sort = sort->next;
        }
    }
    ListNode *ans = dummy->next;
```

```
    delete dummy; dummy = nullptr;
    return ans;
}
```

Question 11. ***(3')H-index Computation***

H-index is often used to evaluate the academic influence of researchers. According to Wikipedia: if a scientist has index h, then h of his/her N papers have ***at least*** *h citations each, and the other $N-h$ papers have* ***no more than*** *h citations each.*

Given an array of citations (each citation is a non-negative integer) of a researcher, please design an algorithm to compute the researcher's h-index. For example, if a researcher's citations array is [3,0,6,1,5], then his/her H-index is 3.

Hint: you can preprocess the citations array using sorting algorithm.

(1) State your algorithm using either natural language or pseudo code.(2')

(2) Is it possible that the same researcher will have different H-indices? State your reason.(1')