ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2021

Homework 2

Due date: 23:59, October 10, 2021

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

Consider a hash table consisting of $M = 11$ slots, and suppose we want to insert integer keys $A = [43, 23, 1, 0, 15, 31, 4, 7, 11, 3]$ orderly into the table. We give the hash function $h1()$ in the form of pseudocode as below:

```
int h1(int key) {
    int x = (key + 7) * (key + 7);
    x = x / 16;
    x = x + key;
    x = x % 11;
    return x;
}
```

Question 1. *Suppose that collisions are resolved via chaining. If there exists collision when inserting a key, the inserted key(collision) is stored at the end of a chain. 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.*

Index	0	1	2	3	4	5	6	7	8	9	10
Keys	31→4	43→15	23	0		1			7	11→3	

Question 2. *What is the load factor λ?*
$\lambda = \frac{10}{11}$

Question 3. *Suppose that collisions are resolved via linear probing. The integer key values listed below are to be inserted, in the order given. Show the home slot (the slot to which the key hashes, before any probing), the probe sequence (do not need to write the home slot again) for each key, and the final contents of the hash table after the following key values have been inserted in the given order:*

Key Value	43	23	1	0	15	31	4	7	11	3
Home Slot	1	2	5	3	1	0	0	8	9	9
Probe Sequence					2,3,4		1,2,3,4,5,6			10

Final Hash Table:

Index	0	1	2	3	4	5	6	7	8	9	10
Keys	31	43	23	0	15	1	4		7	11	3

Question 4. *Suppose that collisions are resolved via a kind of quadratic probing, with the probe function as below:*

$$(k^2 + k)/2$$

This means that if the key is hashed to the i-th slot, we will check the $(i + (k^2 + k)/2)$-th slot for the k-th probing. Show the home slot (the slot to which the key hashes, before any probing), the probe sequence (do not need to write the home slot again) for each key, and the final contents of the hash table after the following key values have been inserted in the given order:

Key Value	43	23	1	0	15	31	4	7	11	3
Home Slot	1	2	5	3	1	0	0	8	9	9
Probe Sequence					2,4		1,3,6			10

Final Hash Table:

Index	0	1	2	3	4	5	6	7	8	9	10
Keys	31	43	23	0	15	1	4		7	11	3

Question 5. *Suppose that we are using a hash table with m slots, where $m > 1$. Recall that in open addressing we store an element in the first empty slot among $A_0(k), A_1(k), \cdots, A_{m-1}(k)$. In this problem,*

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

where h(k) is some hash function.
Prove that the probe sequence will always include at most (m+1)/2 distinct slots.
We can easily get that $A_i(k) = A_{m-i-1}(k)$. When m is even, the distinct slots is $m/2$ and when m is odd, the distinct slots is $(m + 1)/2$.

2: (3*1'+4'+4')Insertion Sort and Bubble Sort

The **question6-8** 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 ***question6-8*** *in the table below.*

Question 6	Question 7	Question 8
ABC	AB	ABC

Question 6. *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 7. *Applying insertion sort and the most basic bubble sort without a flag respectively on the same array, for both algorithms, which of the following statements is/are true? (simply assume we are using swapping for insertion sort)*

(A) There are two for-loops, which are nested within each other.

(B) They need the same amount of swaps.

(C) They need the same amount of element comparisons.

(D) None of the above.

Question 8. *The 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 9. ***(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.

```
struct ListNode
{
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(nullptr) {}
 };

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;
            tail->next = sort->next;//q1
            sort->next = ptr->next;//q2
            ptr->next = sort;//q3
            sort = tail->next;//q4
        }
        else
        {   //no need to insert
            tail = tail->next;
            sort = sort->next;
        }
    }
    ListNode *ans = dummy->next;
    delete dummy; dummy = nullptr;
    return ans;
}
```

Question 10. ***(4')Reverse Engineering***

Consider the following unsorted array, where each letter represents an unknown value, and the array after an unknown number of iterations of selection sort whose implementation is given below. Assume no two elements are equal and we want them sorted in an ascending order.

```
#include<iostream>
using namespace std;
template<class T>
void selection(T data[], int n){
    for(int i=0,j,least;i<n-1;i++){
        for(j=i+1,least=i;j<n;j++){
            if(data[j]<data[least])
                least=j;
        }
        swap(data[least],data[i]);
    }
}
```

Unsorted:
A,B,C,D,E,F,G,H

After ? Iteration of Selection Sort:
F,H,C,D,A,E,G,B

For each relation below, write <, >, or ? for insufficient information regarding the relation between the two objects.

G _____ F
G _____ A
G _____ B
G _____ E

1. >, F is smallest element since it's placed in the front.
2. >, on the 5th iteration, A is moved down the array before G.
3. ?, we don't know whether they're in the right places or they haven't gotten to swap yet.
4. ?, same reaon as 3.