資料結構期末考 (CS 235100)

2017.6.12 13:10-15:25

授課教師：沈之涯

1. (8%) Hash.

(a) (2%) What is a hashing function?

(b) (4%) What is a collision? What is an overflow?

(c) (2%) In static hashing, what is the loading density?

2. (13%) Graph Algorithms.

(a) (4%) Can Floyd-Warshall algorithm find shortest paths in a weighted graph with positive or negative edge and negative cycles? Please provide an example to illustrate.

(b) (3%) Complete the pseudo code blow of the Floyd-Warshall algorithm.

```
1 let dist be a |V| × |V| array of minimum distances initialized to ∞(infinity)
2 for each vertex v
3       dist[v][v] ← 0
4 for each edge (u, v)
5       dist[u][v] ← w(u, v)   // the weight of the edge(u, v)
6 for k from 1 to |V|
7       for i from 1 to |V|
8           for j from 1 to |V|
9               if dist[i][j] > dist[i][k] + dist[k][j]
10                  _____________________
11              end if
```

(c) (2%) What is the time complexity of Floyd-Warshall algorithm?

(d) (4%) Under what condition Dijkstra's algorithm will not work? Please illustrate with an example.

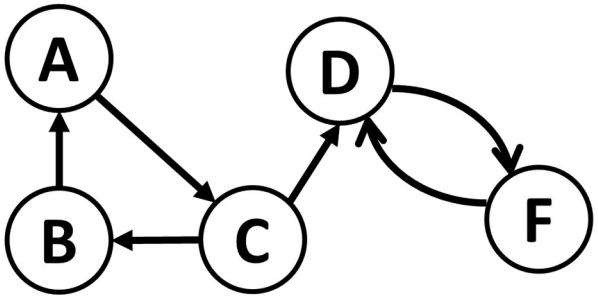
3. (10%) Sorting Algorithms.

(2% for each sorting algorithm. Points for each sorting are given only when all the 4 cases are correct.)

	Bset Case	Average Case	Worst Case	Stable
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	yes
Insertion Sort				
Selection Sort				
Quick Sort				
Merge Sort				
Heap Sort				

4. (11%) A strongly connected component of a directed graph $G = (V, E)$ is a maximal set of vertices $C \subseteq V$ s.t. $\forall$ pair of vertices u and v in C, there is a path from u to v and from v to u.

(a) (3%) What is the number of strongly connected components in the following graph?



(b) (8%) Propose an algorithm to find the strongly connect component of a graph$G = (V, E)$ with time complexity= $O(|V| + |E|)$. (Hint: DFS. If your algorithm cannot achieve $O(|V| + |E|)$ time, points will be given according to the performance and correctness.)

5. (6%) Please describe Quick Sort on the following numbers and show the steps.

{12, 90, 17, 23, 35, 1, 16, 9, 17}

6. (11%) Topological sort can find a linear list on AOV network.

(a) (3%) Does this linear list unique? Explain briefly with a simple example.

(b) (3%) What is the condition for network that could cause the topological ordering fail? Explain briefly with a simple example.

(c) (5%) Write the steps of topological sort and analysis the time complexity of the topological sort.

7. (8%) Huffman Tree.

(a) (3%) The frequencies and the corresponding characters are listed in the following table. Please transform these characters into binary codes by using Huffman algorithm.
We restrict that the character with a smaller frequency should be the left-child of the root, and the larger one should be the right-child of the root.

A	45
B	13
C	12
D	16
E	9
F	5

(b) (5%) Prove the correctness of the Huffman algorithm.

8. (16%) This problem set is about minimum spanning trees. Sollin's algorithm is a minimum spanning tree. It works as follows.

- **Step 1:** Start with a forest that has n spanning trees (each has one vertex).
- **Step 2:** Select one minimum cost edge for each tree. This edge has exactly one vertex in the tree.
- **Step 3:** Delete multiple copies of selected edges and if two edges with the same cost connecting two trees, keep only one of them.
- **Step 4:** Repeat until we obtain only one tree.

(a) (4%) Given a connected weighted graph G, is minimum spanning tree of G unique? If YES, please justify you answer; If NO, please provide a counterexample.
(b) (4%) What is the worst-case time complexity of the Sollin's algorithm mentioned above? (Please analyze step by step).
(c) (8%) Please prove that Sollin's algorithm is able to find the minimum spanning tree.

9. (5%) The comparison-based sorting algorithms are those using only comparisons between elements to gain order information about an input sequence. What is the lower bound of comparison sort? Justify your answer. (1% if only give the lower bound without explain)

10. (12%) Counting sort is an integer sorting algorithm. It sorts a collection of objects according to the key of objects which are small integers. It operates by counting the number of objects that have each distinct key value, and using arithmetic on those counts to determine the positions of each key value in the output sequence. The following is the pseudocode of counting sort.

```
COUNTING-SORT (A, B, k)
1. let C[0..k] be a new array;
2. for i = 0 to k do
3.     C[i] = 0;
4. for j = 0 to A.length do
5.     C[A[i]] + +;
6. for i = 1 to k do
7.     C[i] = C[i] + C[i − 1];
8. for j = A.length downto 1 do
9.     B[C[A[j]]] = A[j];
10.    C[A[j]] − − ;
```

It works as follows.

Assume that the data are in array $A[1..n]$.
Counting sort requires two extra arrays:
$B[1..n]$ holds the sorted output and $C[0..k]$ provides temporary working storage.

- Step1: For each $i = 0$ to k, determine the number of elements $== i$ and stored this value in $C[i]$.
- Step 2: for each $i = 0$ to k, determine the number of input elements $\leq i$ and stored this value in $C[i]$.
- Step 3: Use the information in C to complete sorting and store the result in B

(a) (4%) Given an array {2,5,3,0,2,3,0,3}, use the counting sort to sort this array and show the steps.
(b) (5%) What is the worst-case (please given an example) and the worst-case time complexity?
(c) (3%) Dose counting sort stable or not? If YES, please justify you answer; If NO, please provide a counterexample.