國立交通大學 106 學年度碩士班考試入學招生試題

科目：計算機概論（5081）　　考試日期：106 年 2 月 9 日 第 2 節

系所班別：資訊管理與財務金融學系　　組別：資管碩乙組　　第 1 頁，共 3 頁

【不可使用計算機】＊作答前請先核對試題、答案卷（試卷）與准考證之所組別與考科是否相符!!

1. Explain the following terms.

(a) (4%) Deep learning

(b) (4%) Blockchain

(c) (4%) P2P network

(d) (4%) GPU computing

2. Search algorithms

(a) (6%) Compare semantic search and social search

(b) (8%) Write a PageRank algorithm

3. (10%) Tree traversal refers to the process of visiting each node in a tree data structure. We have a binary tree T, and each node of this tree contains only one English character data. The post-order and in-order traversal sequences of T are BFDIJGHECA and BAFDCIGJEH, respectively. Please draw the binary tree T and write down the pre-order traversal sequence of T.

4. (13%) The heapsort algorithm can be divided into two major parts. In the first step, a max heap is implemented in an array with the layout of a complete binary tree. The root node is a[1] and the child nodes of node a[i] are stored in a[i*2] and a[i*2+1]. In the second step, a sorted array is created by repeatedly swapping the largest element from the heap root to the last heap node, and decreasing the heap size simultaneously. The heap is updated after each swap action to maintain the max heap. Try to program the subroutines "swap" and "adjust" to let the heapsort can work correctly with a n-element unsorted array from x[1] to x[n].

```
void heapsort(int x[], int n)
{
    int i;
    for (i = n / 2;i > 0;i--)
        adjust(x, i, n);
    for (i = n;i > 1;i--)
    {
        swap(x, 1, i);
        adjust(x, 1, i-1);
    }
}
```

5. (5 %) What's the difference between "deadlock prevention" and "deadlock avoidance" in operating system design?

6. (12 %) Calculate the average turnaround time of the following scheduling algorithms:

(a) First-comes, first-service

(b) Nonpreemptive shortest-job-first

(c) Preemptive Shortest-remaining-time-first

Process	Arrival Time	Burst Time
P1	0	8
P2	3	6
P3	5	2
P4	8	4

7.

(a) (4%) The binding of method definition may occur in compile time or run time. What is late binding? What is polymorphism and how does it relate to late binding?

(b) (6%) How does a divide-and-conquer approach solve a problem? Derive the running time T(N) of merge-sort based on using the divide-and-conquer approach for sorting N number of elements. You need to derive the recurrence equation of T(N) and explain the running time of "divide", "conquer" and "combine" steps in the merge-sort, respectively.

8. Assume that you are designing an Item-Sharing Website on the Internet. The data structure *pushedItems* links the items that each user had pushed. Let *N* be the total number of users and *M* be the total number of items, respectively. An array *userItems*[ ] is used to record the pushed items for *N* users. The pushed items of each user are stored as a link list. Assume that each user has a unique user ID from 0 to *N*-1; each item has a unique item ID in the range of 0 to *M*-1. Assume that a similarity function *Sim*(X, Y) has already been implemented to compute the similarity of two users X and Y. The K nearest neighbors of a user X are the top-K users with the highest similarity values with user X. An array *userNbrs*[ ] is used to record each user's K nearest neighbors for *N* users. The K nearest neighbors of each user are stored as a link list and are sorted according to their similarity values in descending (from large to small) order.
A recommendation is conducted to recommend items to a target user X based on the items' recommendation scores (recScore). The recScore of an item J is calculated as the summation of the similarity values of target user X's K nearest neighbors who had pushed item J. The recScore of an item J, which had already been pushed by the target user, is -1.

(a) (8%) Write a program to derive the K nearest neighbors of each user based on the similarity function *Sim*(X, Y). Use the data structure *userNbrs*[ ] to store the K nearest neighbors. Your program should insert a user into the neighbor list by comparing the similarity values. Your program should also handle the constraint that the number of neighbors in the neighbor list is no greater than K.

(b) (12%) Write a program to recommend two items with the top-2 highest recommendation scores for a given target user X. The item that had already been pushed by the target user X should not be recommended. Your program should traverse the link lists of *userNbrs*[X] and *userItems*[ ] to calculate the recommendation scores of items. An array *recScores*[ ] is used to record the recommendation scores of items. Your program should output the two recommended items (itemID and name) based on *recScores*[ ] and *items*[ ]. Analyze the time complexity of your program based on K, N, and M.

```
struct  item_info  {
    char    name[30];
};
item_info    items[M];  double  recScores[M];
struct  itemListNode  {
    int  itemID;
    struct  itemListNode  *link;
};
typedef  struct  itemListNode  * pushedItems;     pushedItems    userItems[N];

struct  nbrListNode  {
    int       nbrID;
    double  similarity;
    struct    nbrListNode  *link;
};
typedef  struct  nbrListNode  * nbrList;       nbrList    userNbrs[N];
```