

历年卷2

开始时间 2023/06/20 10:23:00

结束时间 2023/06/29 20:59:00

答题时长 13596分钟

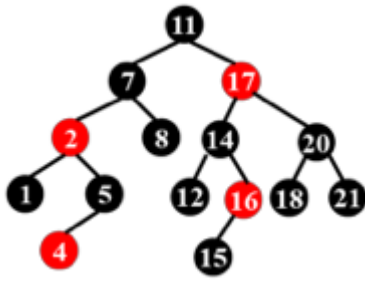
答卷类型 标准答案

总分 100

判断题

得分：暂无 总分：26

- 1-1 In order to solve the maximum finding problem by a parallel algorithm with $T(n) = O(1)$, we need work load $W(n) = \Omega(n^2)$ in return. ~@ (2分)
☐ T ☒ F
- 1-2 For the recurrence equation $T(N) = 9T(N/3) + N^2 \log N$, we obtain $T(N) = O(N^2 \log N)$ according to the Master Theorem. (2分)
☐ T ☒ F
- 1-3 To solve the vertex cover problem, there is a greedy algorithm that collects the vertex with the highest degree (i.e., the one covering the largest number of edges) and remove it from the graph at each stage. This greedy algorithm achieves an approximation ratio of 2. ~@ (2分)
☐ T ☒ F
- 1-4 Let $a = (a_1, a_2, \dots, a_i, \dots, a_j, \dots, a_n)$ denote the list of elements we want to sort. In the quicksort algorithm, if the pivot is selected uniformly at random. Then any two elements get compared at most once and the probability of a_i and a_j being compared is $2/(j-i+1)$ for $j > i$, given that a_i or a_j is selected as the pivot. ~@ (2分)
☐ T ☒ F
- 1-5 In local search, if the optimization function has a constant value in a neighborhood, there will be a problem. ~@ (2分)
☒ T ☐ F
- 1-6 In a Turnpike Reconstruction Problem, given distance set $D = \{2, 2, 4, 6, 6, 8\}$, $x_1 \sim x_4 = (0, 2, 6, 8)$ is the only solution provided that $x_1 = 0$. ~@ (2分)
☒ T ☐ F
- 1-7 In general, for a 3-way merge we need 6 input buffers and 2 output buffers for decreasing the number of passes. ~@ (2分)
☐ T ☒ F
- 1-8 Amortized bounds are weaker than the corresponding worst-case bounds, because there is no guarantee for any single operation. ~@ (2分)
☒ T ☐ F
- 1-9 If a leftist heap can be implemented recursively, so can its counterpart skew heap. ~@ (2分)
☐ T ☒ F
- 1-10 For the document-partitioned strategy in distributed indexing, each node contains a subset of all documents that have a specific range of index. ~@ (2分)
☒ T ☐ F
- 1-11 The following binary search tree is a valid red-black tree. (2分)



~@

☐ T ☒ F

1-12 An $(1 + \epsilon)$ -approximation scheme of time complexity $(n + 1/\epsilon)^3$ is a PTAS but not an FPTAS.~@ (2分)

☐ T ☒ F

1-13 If $P = NP$ then the Shortest-Path (finding the shortest path between a pair of given vertices in a given graph) problem is NP-complete.~@ (2分)

☒ T ☐ F

单选题

得分：暂无 总分：60

2-1 To solve a problem with input size N by divide and conquer, algorithm A divides the problem into 6 subproblems with size $N/2$ and the time recurrences is (3分)

$$T(N) = 6T(N/2) + \Theta(N^2).$$

Now we attempt to design another algorithm B dividing the problem into a subproblems with size $N/4$ and the time recurrences is

$$T(N) = aT(N/4) + \Theta(N^2).$$

In order to beat algorithm A, what is the largest integer value of a for which algorithm B would be asymptotically faster than algorithm A?

- ☐ A. 12
☐ B. 18
☐ C. 24
☒ D. 36

2-2 To solve a problem with input size N by divide and conquer, an algorithm divides the problem into 2 subproblems with size \sqrt{N} (assuming it is an integer) and the time recurrences is (3分)

$$T(N) = 2T(\sqrt{N}) + \log(N).$$

What is the overall time complexity of this algorithm?

- ☐ A. $O(\log(N))$
☐ B. $O((\log(N))^2)$
☒ C. $O(\log(N) \log \log(N))$
☐ D. $O(\sqrt{N} \log(N))$

2-3 Sorting-by-merging is a classic serial algorithm. It can be translated directly into a reasonably efficient parallel algorithm. A recursive description follows. (3分)

MERGE-SORT($A(1), A(2), \dots, A(n); B(1), B(2), \dots, B(n)$)

Assume that $n = 2^l$ for some integer $l \geq 0$

if $n = 1$ then return $B(1) := A(1)$

else call, in parallel, MERGE-SORT($A(1), \dots, A(n/2); C(1), \dots, C(n/2)$) and

- MERGE-SORT($A(n/2+1), \dots, A(n); C(n/2+1), \dots, C(n)$)
- Merge $(C(1), \dots, C(n/2))$ and $(C(n/2+1), \dots, C(n))$ into $(B(1), B(2), \dots, B(n))$ with time $O(n)$

Then the MERGE-SORT runs in ____ .

- ☒ A. $O(n \log n)$ work and $O(\log^2 n)$ time
- ☐ B. $O(n \log n)$ work and $O(\log n)$ time
- ☐ C. $O(n \log^2 n)$ work and $O(\log^2 n)$ time
- ☐ D. $O(n \log^2 n)$ work and $O(\log n)$ time

2-4 In Activity Selection Problem, we are given a set of activities $S = \{a_1, a_2, \dots, a_n\}$ that wish to use a resource (e.g. a room). Each a_i takes place during a time interval $[s_i, f_i]$. (3分)

Let us consider the following problem: given the set of activities S , we must schedule them all using the minimum number of rooms.

Greedy1:

Use the optimal algorithm for the Activity Selection Problem to find the max number of activities that can be scheduled in one room. Delete and repeat on the rest, until no activities left.

Greedy2:

- Sort activities by start time. Open room 1 for a_1 .
- for $i = 2$ to n
if a_i can fit in any open room, schedule it in that room;
otherwise open a new room for a_i .

Which of the following statement is correct?

- ☐ A. None of the above two greedy algorithms are optimal.
- ☐ B. Greedy1 is an optimal algorithm and Greedy2 is not.
- ☒ C. Greedy2 is an optimal algorithm and Greedy1 is not.
- ☐ D. Both of the above two greedy algorithms are optimal.

2-5 ** Load balancing problem: ** (3分)

We have n jobs $j = 1, 2, \dots, n$ each with processing time p_j being an integer number.

Our task is to find a schedule assigning n jobs to 10 identical machines so as to minimize the makespan (the maximum completion time over all the machines).

We adopt the following local search to solve the above load balancing problem.

****LocalSearch: ****

Start with an arbitrary schedule.

Repeat the following until no job can be re-assigned:

- Let l be a job that finishes last.
- If there exists a machine i such that assigning job l to i allows l finish earlier, then re-assign l to be the last job on machine i .
- If such a machine is not unique, always select the one with the minimum completion time.

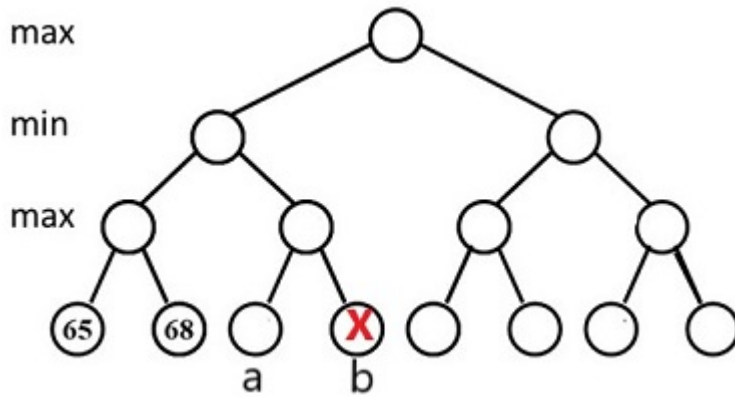
We claim the following four statements:

1. The algorithm LocalSearch finishes within polynomial time.
2. The Load-balancing problem is NP-hard.
3. Let OPT be the makespan of an optimal algorithm. Then the algorithm LocalSearch finds a schedule with the makespan at most of 1.8 OPT.
4. This algorithm finishes within $O(n^2)$.

How many statments are correct ?

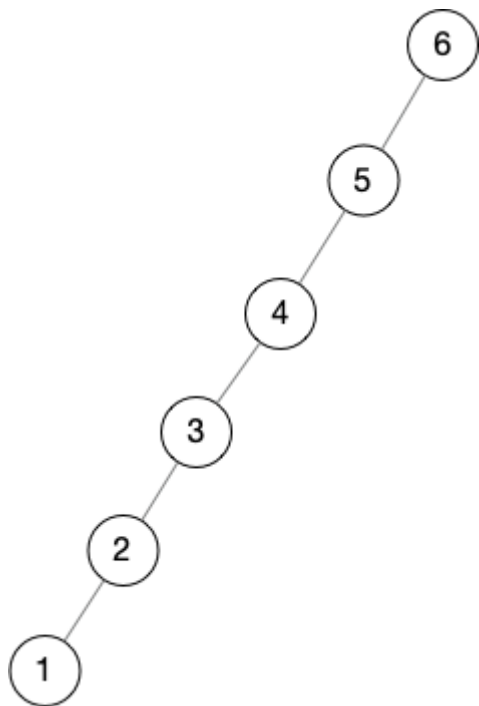
- ☐ A. 0
- ☐ B. 1
- ☐ C. 2
- ☒ D. 3
- ☐ E. 4

- 2-6** Given the following game tree, if node **b** is pruned with α - β pruning algorithm, which of the following statements about the value of node **a** is correct? (3分)



- ☐ A. greater than 65
☐ B. less than 65
☒ C. greater than 68
☐ D. less than 68
- 2-7** A replacement selection is applied to generate the max run with a priority queue of 5 records. When the sequence of numbers is $\{ 11, 81, 17, 14, 94, 28, 35, X, \dots \}$ and the length of the first run is 7, what is the sufficient condition of X ? (3分)
- ☒ A. less than 17
☐ B. greater than 17
☐ C. less than 35
☐ D. less than 94
- 2-8** After inserting number 20 into a binomial queue of 6 numbers $\{ 12, 13, 14, 23, 24, 35 \}$, which of the followings is impossible? (3分)
- ☐ A. the LeftChild link of the node 20 is NULL
☐ B. the NextSibling link of the node 20 is NULL
☒ C. the NextSibling link of node 14 may point to node 20
☐ D. the LeftChild link of node 12 may point to node 14
- 2-9** The potential function Q of a binomial queue is the number of the trees. After merging two binomial queues $H1$ with 22 nodes and $H2$ with 13 nodes, what is the potential change $Q(H1 + H2) - (Q(H1) + Q(H2))$? (3分)
- ☐ A. 2
☐ B. 0
☐ C. -2
☒ D. -3
- 2-10** Start from N single-node splay trees, let's merge them into one splay tree in the following way: each time we select two splay trees, delete nodes one by one from the smaller tree and insert them into the larger tree. Then which of the following statements is NOT true? (3分)
- ☐ A. In any sequence of $N - 1$ merges, there are at most $O(N \log N)$ inserts.
☐ B. Any node can be inserted at most $\log N$ times.
☐ C. The amortized time bound for each insertion is $O(\log^2 N)$.
☒ D. The amortized time bound for each merge is $O(\log N)$.
- 2-11** For the result of accessing the keys 1 and 2 in order in the splay tree in the following figure, let's define $\text{size}(v) = \text{number of nodes in subtree of } v \text{ (} v \text{ included)}$ and potential $\phi = \sum_v \lfloor \log_2 \text{size}(v) \rfloor$, where $\lfloor x \rfloor$ means the greatest integer no larger than x . (3分)
- How many of the following statements is/are TRUE?

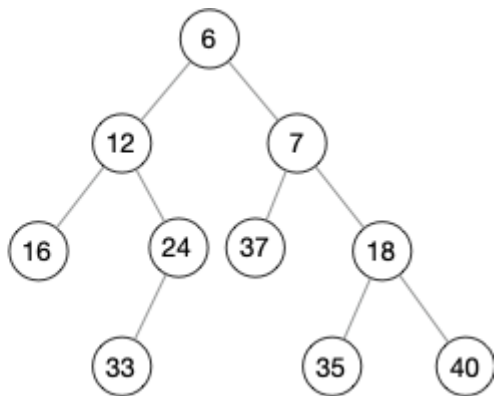
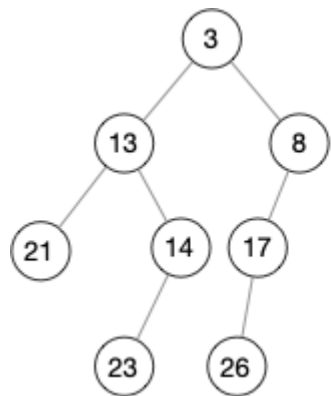
- the potential change from the **initial** tree to the **resulted** tree is -4
- 1 is the sibling of 4
- 5 is the child of 6



- ☐ A. 0
- ☐ B. 1
- ☒ C. 2
- ☐ D. 3

2-12 Merge the two skew heaps in the following figure. How many of the following statements is/are FALSE? (3分)

- the null path length of 8 is the same as that of 12
- 40 is the left child of 18
- the depths of 18 and 33 are the same



- ☒ A. 0
- ☐ B. 1
- ☐ C. 2
- ☐ D. 3

2-13 Insert { 9, 8, 7, 2, 3, 5, 6, 4 } one by one into an initially empty AVL tree. How many of the following statements is/are FALSE? (3分)

- the total number of rotations made is 5 (Note: double rotation counts 2 and single rotation counts 1)
- the expectation (round to 0.01) of access time is 2.75
- there are 2 nodes with a balance factor of -1

- ☒ A. 0
- ☐ B. 1
- ☐ C. 2
- ☐ D. 3

2-14 To build a leftist heap, we can start from placing all the keys as single-node heaps on a queue, and perform the following until only one heap is on the queue: dequeue two heaps, merge them, and enqueue the result. (3分)

Then the best description of the time complexity of this procedure is:

- ☒ A. $O(N)$
- ☐ B. $O(\log N)$
- ☐ C. $O(N \log N)$
- ☐ D. $O(\sqrt{N})$

2-15 Assume that there are 10000 documents in the database, and the statistical data for one query are shown in the following table. One metric for evaluating the relevancy of the query is F- α score, which is defined as $((1+\alpha) \cdot (\text{precision} \cdot \text{recall})) / (\alpha \cdot \text{precision} + \text{recall})$. Then the F-0.5 ($\alpha=0.5$) score for this query is: (3分)

	Relevant	Irrelevant
Retrieved	4800	1200
**Not Retrieved **	3200	800

- ☐ A. 0.80
- ☒ B. 0.72
- ☐ C. 0.60
- ☐ D. 0.65

2-16 After inserting { 3, 4, 5, 6, 1, 2, 7 } into an initially empty red-black tree, which of following is **False**? (3分)

- ☐ A. The resulting tree is a full tree.
- ☐ B. 4 is the root with the black height as 2.
- ☐ C. 3 is the right child of 2, and the color of 3 is red.
- ☒ D. 5 is the left child of 6, and the color of 5 is black.

2-17 Assume $P \neq NP$, please identify the false statement. (3分)

- ☐ A. There cannot exist a ρ -approximation algorithm for bin packing problem for any $\rho < 3/2$.
- ☐ B. In the minimum-degree spanning problem, we are given a graph $G=(V, E)$ and wish to find a spanning tree T of G so as to minimize the maximum degree of nodes in T . Then it is NP-complete to decide whether or not a given graph has minimum-degree spanning tree of maximum degree two.
- ☒ C. In the minimum-degree spanning problem, we are given a graph $G=(V, E)$ and wish to find a spanning tree T of G so as to minimize the maximum degree of nodes in T . Then there exists an algorithm with approximation ratio less than $3/2$.
- ☐ D. In the knapsack problem, for any given real number $\epsilon > 0$, there exists an algorithm with approximation ratio less than $1 + \epsilon$.

2-18 If P and NP are different, which of the following statements is true? (3分)

- ☒ A. There is no polynomial time algorithm to solve the vertex cover problem.
- ☐ B. $P \cap NP\text{-Complete} \neq \emptyset$.
- ☐ C. We can find polynomial time solution for Hamilton cycle problem.
- ☐ D. $P = NP\text{-Complete}$.

2-19 If X is a problem in class NP, then how many of the following statements is/are TRUE? (3分)

- There is no polynomial time algorithm for X .
- There is a polynomial time algorithm for X .
- If X can be solved deterministically in polynomial time, then $P = NP$.

- ☒ A. 0
☐ B. 1
☐ C. 2
☐ D. 3

2-20 In the maximum satisfiability problem (MAX SAT), the input consists of n Boolean variables x_1, \dots, x_n , m (3分)

clauses C_1, \dots, C_m (each of which consists of a disjunction (that is an "or") of some number of the variables and their negations, e.g. $x_3 \vee \bar{x}_5 \vee x_{11}$, where \bar{x}_i is the negation of x_i), and a nonnegative weight w_j for each clause C_j . The objective of the problem is to find an assignment of the true/false to the x_i that maximizes the weight of the satisfied clauses.

A variable or a negated variable is a literal. The number of literals in a clause is called its length. Denote l_j to be the length of a clause C_j . Clauses of length 1 are called unit clauses.

Randomized algorithm RA: Setting each x_i to true with probability p independently.

Which of the following statement is false?

- ☐ A. Let $p = 1/2$, the randomized algorithm RA is a 2-approximation algorithm.
- ☒ B. If $l_j \geq 3$ for each clause C_j . Let $p = 1/2$, the randomized algorithm RA is a $9/8$ -approximation algorithm.
- ☐ C. If MAX SAT instances do not have unit clauses \bar{x}_i , we can obtain a randomized $\frac{2}{\sqrt{5}-1} \approx 1.618$ -approximation algorithm for MAX SAT.
- ☐ D. One could obtain a better bound on optimal solution than $\sum_{j=1}^m w_j$ for MAX SAT.

程序填空题

得分：暂无 总分：6

5-1 The function **FindKey** is to check if a given **key** is in a B+ Tree with its root pointed by **root**.

Return **true** if **key** is in the tree, or **false** if not. The B+ tree structure is defined as following:

```
static int order = DEFAULT_ORDER;
typedef struct BpTreeNode BpTreeNode;
struct BpTreeNode {
    BpTreeNode** childrens; /* Pointers to childrens. This field is not used by leaf nodes.
    ElementType* keys;
    BpTreeNode* parent;
    bool isLeaf; /* 1 if this node is a leaf, or 0 if not */
    int numKeys; /* This field is used to keep track of the number of valid keys.
    In an internal node, the number of valid pointers is always numKeys + 1. */
};
```

```
bool FindKey(BpTreeNode * const root, ElementType key){
    if (root == NULL) {
        return false;
    }
    int i = 0;
    BpTreeNode * node = root;
    while (!node->isLeaf (3分)) {
        i = 0;
        while (i < node->numKeys) {
```

```

        if ( key >= node->keys[i] (3分)) i++;
        else break;
    }
    node = node->childrens[i];
}
for(i = 0; i < node->numKeys; i++){
    if(node->keys[i] == key)
        return true;
}
return false;
}

```

函数题

得分：暂无 总分：8

6-1 Decode (8分)

Suppose that a string of English letters is encoded into a string of numbers. To be more specific, **A-Z** are encoded into **0-25**. Since it is not a prefix code, the decoded result may not be unique. For example, **1213407** can be decoded as **BCBDEAH**, **MBDEAH**, **BCNEAH**, **BVDEAH** or **MNEAH**. Note that **07** is not **7**, hence cannot be decoded as **H**.

Your job is to tell in how many different ways we can decode a numeric string.

Format of function:

```
int Decode( char NumStr[] );
```

where **NumStr** is a string consisting of only the numbers **0-9**.

The function **Decode** is supposed to return the number of different ways we can decode **NumStr**.

Since the answer might be super large, you only need to output the answer modulo 1000000007.

Sample program of judge:

```

#include <stdio.h>
#include <string.h>

#define MAXN 100
#define BASE 1000000007

int Decode( char NumStr[] );

int main()
{
    char NumStr[MAXN];

    scanf("%s", NumStr);
    printf("%d", Decode(NumStr));

    return 0;
}

/* Your function will be put here */

```

Sample Input:

1213407

Sample Output:

