ADS2021Midterm-yy

开始时间 2022/04/15 16:30:00	结束时间 2022/04/29 12:00:00	答题时长 45分钟	
答卷类型 标准答案	总分 100		

判断题

得分：暂无　总分：20

1-1 A 2-3 tree with 12 leaves may have at most 10 nonleaf nodes. (2分)

- T
- F

1-2 In a red-black tree, the number of internal nodes in the subtree rooted at x is no more than $2^{bh(x)}-1$ where $bh(x)$ is the black-height of x. (2分)

- T
- F

1-3 For the recurrence equation $T(N)=aT(N/b)+f(N)$, if $af(N/b)=f(N)$, then $T(N)=\Theta(f(N)log_bN)$. (2分)

- T
- F

1-4 When measuring the relevancy of the answer set of a search engine, the recall is low means that most of the relevant documents are not retrieved. (2分)

- T
- F

1-5 In an AVL tree, it is possible to have this situation that the balance factors of a node and both of its children are all +1. (2分)

- T
- F

1-6 Finding the minimum key from a splay tree will result in a tree with its root having no left subtree. (2分)

- T
- F

1-7 A skew heap is a heap data structure implemented as a binary tree. Skew heaps are advantageous because of their ability to merge more quickly than balanced binary heaps. The worst case time complexities for Merge, Insert, and DeleteMin are all $O(N)$, while the amorited time complexities for Merge, Insert, and DeleteMin are all $O(logN)$. (2分)

- T
- F

1-8 It is guaranteed that an exhaustive search can always find the solution in finite time. (2分)

- T
- F

1-9 Making N insertions into an initially empty binomial queue takes $O(N)$ time in the worst case. (2分)

- T
- F

1-10 To solve a problem by dynamic programming instead of recursions, the key approach is to store the results of computations for the subproblems so that we only have to compute each different subproblem once. Those solutions can be stored in an array or a hash table. (2分)

- T
- F

单选题

得分：暂无　总分：40

2-1 When solving a problem with input size N by divide and conquer, if at each stage the problem is divided into 4 sub-problems of equal size $N/5$, and the conquer step takes $O(logN)$ to form the solution from the sub-solutions, then the overall time complexity is: (4分)

- A. $O(N)$
- B. $O(logN)$
- C. $O(logN^2)$
- D. $O(N^{log4/log5})$

2-2 Which of the following statements about Inverted File Index is FALSE? (4分)

- A. Inverted File contains a list of pointers to all occurrences of term in the text
- B. Inverted File Index is used for accelerate the speed of information retrieval
- C. Inverted File Index is a kind of data structure based on binary search tree
- D. Index is a mechanism for locating a given term in a text

2-3 Insert (88, 70, 61, 96, 120, 90) one by one into an initially empty AVL tree. Then the preorder traversal sequence of the resulting AVL tree is: (4分)

- A. 61,70,88,90,96,120
- B. 90,70,61,88,96,120
- C. 88,70,61,90,96,120
- D. 88,70,61,96,90,120

2-4 After deleting 10 from the red-black tree given in the figure, which one of the following statements must be FALSE? (4分)

- A. 11 is the parent of 6, and 14 is red
- B. 8 is the parent of 15, and 7 is black
- C. 11 is the parent of 15, and there are 2 red nodes in the tree
- D. 8 is the parent of 15, and there are 2 red nodes in the tree

2-5 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.

Then the best description of the time complexity of this procedure is: (4分)

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

2-6 You are to maintain a collection of lists and to support the following operations. (4分)

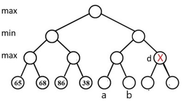
(i) `insert(item, list)`: insert `item` into `list` (cost = 1).

(ii) `sum(list)`: sum the items in `list`, and replace the `list` with a list containing one item that is the sum (cost = length of `list`).

We show that the amortized cost of an insert operation is O(1) and the amortized cost of a sum operation is O(1).
If we assume the potential function to be the number of elements in the list, which of the following is FALSE?

- A. For insert, the actual cost is 1.
- B. For insert, the change in potential is 1. The amortized cost is 2.
- C. For sum, the actual cost is k.
- D. For sum, the change cost is 2-k. The amortized cost is 2.

2-7 Given the following game tree, node **d** is pruned with α-β pruning algorithm, which of the following statements about the value of node **a** or node **b** is correct? (4分)



- A. both are greater than 68
- B. both are less than 68
- C. one of them greater than 68
- D. one of them less than 68

2-8 Delete the minimum number from the given binomial queue in the following figure. Which one of the following statements is FALSE? (4分)

- A. there are three binomial trees after deletion, which are B_0, B_1 and B_4
- B. 11 is the root of a binomial tree
- C. 23 is not the root of any resulting binomial tree
- D. 29 and 23 are both children of 4

2-9 Rod-cutting Problem: Given a rod of total length N inches and a table of selling prices P_i for lengths $L=1,2,\cdots,M$. You are asked to find the maximum revenue R_N obtainable by cutting up the rod and selling the pieces. For example, based on the following table of prices, if we are to sell an 8-inch rod, the optimal solution is to cut it into two pieces of lengths 2 and 6, which produces revenue $R_8=P_2+P_6=5+17=22$. And if we are to sell a 3-inch rod, the best way is not to cut it at all. (4分)

Length L	1	2	3	4	5	6	7	8	9	10
Price P_L	1	5	8	9	10	17	17	20	23	28

Which one of the following statements is FALSE?

- A. This problem can be solved by dynamic programming
- B. If $N>M$, we have $R_N=\max_{1\le i\le M}\{R_i+R_{N-M}\}$
- C. If $N\le M$, we have $R_N=\max\{P_N,\max_{1\le i\le N}\{R_i+R_{N-i}\}\}$
- D. The time complexity of this algorithm is $O(N^2)$

2-10 Which one of the following problems can be best solved by dynamic programming? (4分)

- A. Mergesort
- B. Closest pair of points problem
- C. Quicksort
- D. Longest common subsequence problem

程序填空题

得分：暂无　总分：40

5-2 Giving an array of N integers, please calculate the maximum subsegment sum.

```
#include <algorithm>
#include <cstdio>
using namespace std;
const int N = 100010;
int a[N];
long long work(int l, int r) {
    if (l == r)
        return a[l]                          (4分);
    int mid = (l + r) >> 1;
    long long res = 0;
    res = max(res, work(l, mid));
    res =  max(res, work(mid + 1, r))        (4分);
    long long mxl = 0, suml = 0, mxr = 0, sumr = 0;
    for (int i = mid; i >= l; i--) {
        suml += a[i];
        mxl = max(mxl, suml);
    }
    for (int i = mid + 1; i <= r; i++) {
        sumr += a[i]                         (4分);
        mxr = max(mxr, sumr)                 (4分);
    }
    res =  max(res, mxl + mxr)               (4分);
    return res;
}
int main()
{
    int T;
    scanf("%d",&T);
    while (T--) {
        int n;
        scanf("%d", &n);
        for(int i = 1; i <= n; i++)
            scanf("%d", &a[i]);
        printf("%lld\n", work(1, n));
    }
    return 0;
}
```

5-1 Suppose we are given n points $p_1,p_2\ldots,p_n$ located on the x-axis. x_i is the x-coordinate of p_i. Let us further assume that $x_1=0$, and the points are given from left to right. These n points determine $\frac{n(n-1)}{2}$ (not-necessarily unique) distances $d_1,d_2\ldots,d_{\frac{n(n-1)}{2}}$ between every pair of points of the form $|x_i-x_j|$ $(i\neq j)$.

The *Turnpike reconstruction problem* is to reconstruct a point set from the distances.

This algorithm is to read the number n and $\frac{n(n-1)}{2}$ distances d_i, then print one valid sequence of points p_i. Please complete the following program.

```
#include <algorithm>
#include <cstdio>
const int MAXN = 1000, MAXD = MAXN * (MAXN - 1) / 2;
int p[MAXN], d[MAXD], n, m;
int id[MAXD];
bool used[MAXD];
int binary_search(int x, int m) {
    int l = 0, r = m;
    while (l < r) {
        int mid = (l + r) / 2;
        if (d[mid] < x || id[mid] == x && used[mid]))
            l = mid + 1                      (4分);
        else
            r = mid;
    }
    return l;
}
bool recursive(int now, int top, int m) {
    int i;
    for (i = 0; i < now; i++) {
        id[top + i] = binary_search(abs(p[i] - p[now]), m);
        if(d[top + i] != abs(p[i]- p[now])   (4分) && !used[id[top + i]])
            used[id[top + i]] = true;
        else break;
    }
    if (i == now) {
        if (now == n - 1)
            return true;
        while (used[m - 1])
            m--;
        p[now + 1] = d[m - 1];
        if (recursive(now + 1, top + now, m))
            return true;
        if (now <= 1)
            return false;
        p[now + 1] = p[1]- d[m - 1]          (4分);
        if (recursive(now + 1, top + now, m))
            return true;
    }
    for(int j = 0; j < i; j++)
        used[id[top + j]] = false            (4分);
    return false;
}
int main()
{

    scanf("%d", &n);
    m = n * (n - 1) / 2;
    for (int i = 0; i < m; i++)
        scanf("%d", &d[i]);
    std::sort(d, d + m);
    p[0] = 0;
    if (!recursive(0, 0, m                   (4分))) {
        puts("NO ANSWER");
        return 0;
    }
    std::sort(p, p + n);
    for (int i = 0; i < n; i++)
        printf("%d\n", p[i]);
    return 0;
}
```