扫码获取答案解析

提高组 CSP-S 2025 初赛模拟卷 2

一、单项选择题（共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项）

1. 在 NOI Linux 系统终端中，（　　）命令可以用来修改文件名。

A. `mkdir`　　B. `cp -a`　　C. `mv`　　D. `touch`

2. 以下关于二叉排序树的表述中，不恰当的一项是（　　）。

A. 若左子树不空，则其所有结点的值均小于根结点的值

B. 二叉排序树的根结点一定有最小值或者最大值

C. 若右子树不空，则其所有结点的值均大于根结点的值

D. 二叉排序树的左右子树也分别为二叉排序树

3. 计算机病毒最不容易攻击的是（　　）。

A. 硬盘　　B. 内存　　C. 只读光盘　　D. U 盘

4. 假设我们有以下 C++代码：

```
unsigned char a=143, b=3, c=4;
a<<=2;
int res = a|(b+c>>1);
```

则 `res` 的值是（　　）。

A. 63　　B. 61　　C. 573　　D. 575

5. 使用邻接表表示一个简单有向图，图中包含 `n` 个顶点、`m` 条边，则该出边表中边结点的个数为（　　）。

A. `n+m`　　B. `n*(n-1)`　　C. `m`　　D. `n*m`

6. （　　）问题不是典型的与动态规划算法相关的问题。

A. 最长公共子序列　　B. 打水

C. 多重背包　　D. 最长递增子序列

7. 解决 RMQ 问题通常使用（　　）。

A. 哈夫曼树　　B. 并查集　　C. 哈希表　　D. 稀疏表

8. 以下关于强连通图的说法中，正确的是（　　）。

A. 图中一定有环

B. 每个顶点的度数都大于 0

C. 对于大于 1 个点的强连通图，任意两个顶点之间都有路径相连

D. 每个顶点至少都连有一条边

9. 甲乙两人进行比赛，每局比赛获胜的概率相同，均为甲获胜的概率为 0.4，乙获胜的概率为 0.6，现规定两人持续比赛，直到有一方比对方获胜局数多两局时获得一场比赛的胜利，则甲获胜的概率是（　　）。

A. 2/5　　B. 4/9　　C. 4/13　　D. 1/3

10. 给定地址区间为 0~12 的哈希表，哈希函数为 h(x) = x % 13，采用二次探查的冲突解决策略（出现冲突后会往后依次探查以下位置：h(x)，h(x)+1²，h(x)-1²，h(x)+2²，h(x)-2²，h(x)+3²，h(x)-3²，...）。哈希表初始为空表，依次存储(26，36，13，18，39，3，0)后，请问 0 存储在哈希表的哪个地址中？（　　）

A. 7　　B. 6　　C. 5　　D. 9

11. STL 模板中的容器可以分为顺序容器和关联容器，下列选项中，（　　）不属于容器适配器。

A. iterator　　B. stack　　C. queue　　D. priority_queue

12. 下面关于 C++类继承的说法中，错误的是（　　）。

A. 一个类可以继承多个类

B. 一个类可以继承另一个类的子类

C. 一个类可以被多个类继承

D. 抽象类必须被至少一个类继承，否则会发生编译错误

13. 假设 G 是一张有 m 个点、n 条边的图，小明想找一棵有 n 个结点的最小生成树，必须删去若干点和（　　）条边才能将其变成这样的一棵树。

A. m-n-1　　B. m+n-1　　C. 1　　D. m-n+1

14. 一只蚂蚁从一个棱长为2的正方体的一个顶点出发，沿着棱爬，则其爬过所有棱长且回到出发的顶点的最短路径长是（　　）。

A. 28　　B. 32　　C. 36　　D. 40

15. 二叉堆算法插入操作和删除操作的时间复杂度分别为（　　）。

A. $O(\log n)$、$O(\log n)$　　B. $O(n)$、$O(n)$

C. $O(n)$、$O(\log n)$　　D. $O(\log n)$、$O(n)$

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题1.5分，选择题每题3分，共计40分）

（1）

```
01 #include <bits/stdc++.h>
02
03 using namespace std;
04
05 const int N = 3e5 + 10;
06 const int mod = 1e9 + 7;
07
08 int n, a[N];
09
10 using ii = pair<int, int>;
11
12 int dp[N];
13 int sum[N];
14
15 int main() {
16     ios::sync_with_stdio(0);
17     cin.tie(0);
18     cin >> n;
19     for (int i = 1; i <= n; ++i) cin >> a[i];
20     set<ii> s;
21     sum[0] = 1;
22     for (int i = 1, res = 0; i <= n; ++i) {
23         ii u;
24         while (!s.empty()) {
25             u = (*s.rbegin()); //*rbegin是反向迭代器
26             if (u.first > a[i]) {
```

```
27              res = (res - dp[u.second] + mod) % mod;
28              s.erase(u);
29          } else break;
30      }
31      if (s.empty()) dp[i] = sum[i - 1];
32      else dp[i] = (res + sum[i-1] - sum[u.second]) % mod;
33      if (dp[i] < 0) dp[i] += mod;
34      sum[i] = (sum[i - 1] + dp[i]) % mod;
35      s.insert({a[i], i});
36      res = (res + dp[i]) % mod;
37  }
38  int ans = 0;
39  for (ii u: s) ans = (ans + dp[u.second]) % mod;
40  cout << ans << endl;
41  return 0;
42 }
```

保证输入 1≤N≤300000，a 是一个排列。

■ 判断题（每题 2 分）

16. 本段代码的时间复杂度为 $O(N)$。（　　）
17. 本段代码不可能输出负数。（　　）
18. 第 19 行代码从++i 改成 i++后，程序仍能正常运行，并且输出结果不变。（　　）

■ 选择题

19. 对于输入

```
3
2 3 1
```

程序的输出是（　　）。

A. 1　　B. 2　　C. 3　　D. 4

20. 对于输入

```
4
2 1 4 3
```

程序的输出是（　　）。

A. 2　　B. 4　　C. 6　　D. 8

（2）

```
01 #include <bits/stdc++.h>
02 #define v first
03 #define id second
04 using namespace std;
05
06 typedef long long ll;
07
08 const int N=100010;
09 int n,d,cnt;
10 ll a[N],num[N];
11 int f[N],g[N];
12 typedef pair<int,int> pii;
13
14 pii mx[N<<2];
15
16 pii query(int u,int l,int r,int ql,int qr)
17 {
18     if(ql>qr) return {-1,0};
19     if(ql<=l&&r<=qr) return (mx[u].v?mx[u]:make_pair(-1,0));
20     int mid=(l+r)>>1;
21     if(ql<=mid&&qr>mid) return max(query(u<<1,l,mid,ql,qr),
           query(u<<1|1,mid+1,r,ql,qr));
22     else if(ql<=mid) return query(u<<1,l,mid,ql,qr);
23     else return query(u<<1|1,mid+1,r,ql,qr);
24 }
25
26 void modify(int u,int l,int r,int pos,pii v)
27 {
28     if(l==r) {mx[u]=max(mx[u],v); return;}
29     int mid=(l+r)>>1;
30     if(pos<=mid) modify(u<<1,l,mid,pos,v);
31     else modify(u<<1|1,mid+1,r,pos,v);
32     mx[u]=max(mx[u<<1],mx[u<<1|1]);
33 }
34
35 void print()
36 {
37     int mx=0;
38     for(int i=1;i<=n;i++) if(f[i]>f[mx]) mx=i;
```

```
39    printf("%d\n",f[mx]);
40    vector<int> ans;
41    ans.push_back(mx);
42    while(g[mx]) mx=g[mx],ans.push_back(mx);
43    reverse(ans.begin(),ans.end());
44    for(int x:ans) printf("%d ",x); puts("");
45 }
46
47 int main()
48 {
49    scanf("%d %d",&n,&d);
50    for(int i=1;i<=n;i++)
51    {
52       scanf("%lld",&a[i]);
53       num[i]=a[i];
54    }
55    sort(num+1,num+n+1);
56    cnt=unique(num+1,num+n+1)-num-1;
57    for(int i=1;i<=n;i++)
58    {
59       int x=lower_bound(num+1,num+cnt+1,a[i])-num;
60       int l=upper_bound(num+1,num+cnt+1,a[i]-d)-num-1;
61       int r=lower_bound(num+1,num+cnt+1,a[i]+d)-num;
62       pii ans = max({{0,0}, query(1,1,cnt,1,l),
                 query(1,1,cnt,r,cnt)});
63       f[i]=ans.v+1,g[i]=ans.id;
64       modify(1,1,cnt,x,{f[i],i});
65    }
66    print();
67    return 0;
68 }
```

保证输入数据中 $1\leqslant n\leqslant 10^5$，$0\leqslant d\leqslant 10^9$，$1\leqslant a[i]\leqslant 10^{15}$。

■ **判断题（每题 2 分）**

21. d 非 0 时，第 60 行与第 61 行代码中的 l 与 r 值一定不一样。（　　）

22. 第 23 行后应该再加一行代码 else return make_pair(-1,0)，否则在一些情况下，函数不会返回值，导致未定义行为。（　　）

■ 选择题

23. 这段代码的时间复杂度是（　　）。

A. $O(N)$　　B. $O(N\log N)$　　C. $O(N\log^2 N)$　　D. $O(N^2)$

24. 如果 a[i]的值为 10，d 的值为 3，a 的值为 1 5 6 8 10 12 14 17，那么 l 和 r 的值分别为（　　）。

A. 3 7　　B. 4 7　　C. 3 6　　D. 4 6

25. 以下说法中正确的是（　　）。

A. 如果需要单点修改，区间查询 max 值，在本段代码中也可以使用树状数组，时间复杂度不变。

B. 如果输入合法，在本段代码中，upper_bound(num+1,num+cnt+1,a[i]-d)和 lower_bound(num+1,num+cnt+1,a[i]+d)一定不同。

C. modify(1,1,cnt,x,{f[i],i})这段代码会递归$\lceil\frac{cnt}{2}\rceil$次。

D. 第60行代码中，int后面的内容可以改成l=lower_bound(num+1,num+cnt+1,a[i]-d)-num，与原来 l=upper_bound(num+1,num+cnt+1,a[i]-d)-num-1 的作用相同。

（3）

```
01 #include <bits/stdc++.h>
02 #define int long long
03 using namespace std;
04 const int N=5e5+10;
05 vector<int> e[N];
06 int w[N];
07 int L[N],R[N];
08 int n,m,res;
09
10 void add(int a,int b)
11 {
12     e[a].push_back(b),e[b].push_back(a);
13 }
14
15 void dfs(int u,int fa)
16 {
17     vector<int> S;
18     if(u<=m) return;
```

```
    for(auto v:e[u])
    {
        if(v==fa) continue;
        dfs(v,u);
        S.push_back(L[v]),S.push_back(R[v]);
    }
    sort(S.begin(),S.end());
    int sz=S.size();
    if(sz&1) L[u]=R[u]=S[sz/2];
    else L[u]=S[sz/2-1],R[u]=S[sz/2];
    for(auto v:e[u])
    {
        if(v==fa) continue;
        if(L[u]>R[v]) res+=L[u]-R[v];
        else if(L[u]<L[v]) res+=L[v]-L[u];
        else res+=0;
    }
}

signed main()
{
    ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    cin>>n>>m;
    for(int i=1,a,b;i<n;i++)
    {
        cin>>a>>b;
        add(a,b);
    }
    for(int i=1;i<=m;i++) cin>>w[i],L[i]=R[i]=w[i];
    if(n==2)
    {
        cout<<abs(w[1]-w[2])<<"\n";
        return 0;
    }
    dfs(m+1,0);
    cout<<res<<"\n";
    return 0;
}
```

题目保证输入是一棵树，其中叶子结点标号为[1,M]，2≤M≤N≤500000，其他读入的数据均不会大于 500000。

■ 判断题（每题 2 分）

26. 不考虑 vector 的复杂度，这段代码的时间复杂度为 $O(N)$。（ ）

27. 删除第 28 行代码，并删除第 27 行代码，程序仍然可以输出相同的结果。（ ）

28. 当 n=2 的时候，程序输出不会大于 500000。（ ）

■ 选择题

29. 本段代码中 res 可能达到的最大值为多少？（ ）

A. 250000000000　　B. 125000000000

C. 124999250001　　D. 62500000000

30. 假如 n=5，m=3，w[1]=10，w[2]=20，w[3]=30，在程序正确运行的前提下，理论上本题 res 的最小值是多少？（ ）

A. 20　　B. 15　　C. 10　　D. 5

31. 以下说法中正确的是（ ）。

A. 本段代码中每次 sort 的复杂度最大为 $O(n\log n)$，因此，该程序的复杂度为 $O(n^2\log n)$。

B. 本段代码中的 dfs(m+1,0)改成 dfs(1,0)，结果不变。

C. 如果读入 n 条边，程序一定会读入一个环，导致 dfs 过程出现死循环。

D. 如果给定的是一条链，则输出一定是|w[1]-w[2]|。

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）题目描述：

给定一个只有左右括号的字符串，然后用 H、G 两种字符来标记这个序列，所有标记 H 的括号可以组成一个正确的括号序列，所有标记 G 的括号也组成一个正确的括号序列，然后输出这种标记方案的总数 mod 2012 的值。可以用一个 dp 来解决问题。

```
01 #include <iostream>
02 #include <cstring>
03 #include <cmath>
04 #include <cstdio>
05
06 using namespace std;
07
08 const int mod=2012;
```

```
09 char s[1005];
10 int n,f[1005][1005],g[1005][1005];
11
12 int main()
13 {
14     scanf("%s",s+1);
15     n=①;
16     int oo=0;
17     f[0][0]=1;
18     for(int i=1;i<=n;++i)
19     {
20         if(s[i]=='(')
21         {
22             oo++;
23             for(int j=0;j<=oo;++j)
24             {
25                 if(j) g[j][oo-j] = (g[j][oo-j] + f[j-1][oo-j])%mod;
26                 if(oo-j) ②;
27             }
28         }
29         else if(s[i]==')')
30         {
31             ③;
32             for(int j=0;j<=oo;++j)
33             {
34                 g[j][oo-j]=④;
35                 g[j][oo-j]=(g[j][oo-j]+f[j][oo-j+1])%mod;
36             }
37         }
38         for(int j=0;j<=oo;++j) ⑤;
39     }
40     printf("%d",f[0][0]);
41     return 0;
42 }
```

32. ①处应填（　　）。

A. strlen(s)　　B. strlen(s+1)　　C. strlen(s)+1　　D. strlen(s+1)-1

33. ②处应填（　　）。

A. g[j][oo-j] = (g[j][oo-j] + f[j][oo-j-1]) % mod.

B. `g[j][oo-j] = (g[j][oo-j] + f[j-1][oo-j]) % mod`

C. `g[j][oo-j] = (g[j][oo-j] + f[j][oo-j]) % mod`

D. `g[j][oo-j] = (g[j][oo-j] + f[j-1][oo-j-1]) % mod`

34. ③处应填（　　）。

A. `oo++`　　B. `oo--`　　C. `oo=0`　　D. `oo=n`

35. ④处应填（　　）。

A. `(g[j][oo-j] + f[j][oo-j+1]) % mod`

B. `(g[j][oo-j] + f[j][oo-j]) % mod`

C. `(g[j][oo-j] + f[j+1][oo-j+1]) % mod`

D. `(g[j][oo-j] + f[j+1][oo-j]) % mod`

36. ⑤处应填（　　）。

A. `f[j][oo-j] = g[j][oo-j], g[j][oo-j] = 0`

B. `f[j][oo-j+1] = g[j][oo-j], g[j][oo-j] = 0`

C. `f[j][j] = g[j][oo-j], g[j][oo-j] = 0`

D. `f[j][j+1] = g[j][oo-j], g[j][oo-j] = 0`

（2）题目描述：

对 n 个单词进行加密，过程如下。

选择一个英文字母表的排列作为密钥。

将单词中的 a 替换为密钥中的第一个字母，b 替换为密钥中的第二个字母，以此类推。

请你构造一组密钥，使得对所有单词加密并且按照字典序升序排列后，最初的第 ai 个单词位于第 i 个位置，如果不能，输出 NE，否则输出 DA 并且下一行输出一种可行的密钥。

可以想到，如果把字典序限制看成一条有序边，可以按照拓扑排序来分配密钥，如果出现环，则说明无解。

```
01 #include <bits/stdc++.h>
02
03 #define LL  long long
04 using namespace std;
05
06 const LL M1=300,M2=30;
07 LL n;
08 string a[M1],a_sort[M1];
```

```
09 LL du[M1];
10 queue<LL>qu;
11 vector<LL>ve[M2];
12 LL ans[M2];
13
14 int main() {
15     cin>>n;
16     for(LL i=1;i<=n;i++) cin>>a[i];
17     for(LL i=1;i<=n;i++) {
18         LL b;
19         cin>>b;
20         ①;
21     }
22
23     for(LL i=1;i<n;i++) {
24         bool flag=true;
25         for(LL j=0;j<②;j++)
26             if(a_sort[i][j]!=a_sort[i+1][j]) {
27                 ③;
28                 du[a_sort[i+1][j]-'a']++;
29                 flag=false;
30                 break;
31             }
32         if(flag&&④) {
33             cout<<"NE";
34             return 0;
35         }
36     }
37
38     for(LL i=0;i<26;i++) if(du[i]==0) qu.push(i);
39
40     LL cnt=0;
41     while(!qu.empty()) {
42         LL v=qu.front();
43         qu.pop();
44         ans[v]=cnt;
45         cnt++;
46         for(LL i=0;i<(LL)ve[v].size();i++) {
47             du[ve[v][i]]--;
48             if(⑤) qu.push(ve[v][i]);
49         }
```

```
50    }
51
52    if(cnt!=26) cout<<"NE";
53    else {
54        cout<<"DA"<<endl;
55        for(LL i=0;i<26;i++) {
56            cout<<(char)(ans[i]+'a');
57        }
58    }
59    return 0;
60 }
```

37. ①处应填（　　）。

A. a_sort[i]=a[b]　　B. a_sort[i]=a[i]

C. a_sort[b]=a[b]　　D. a_sort[b]=a[i]

38. ②处应填（　　）。

A. a_sort[i].size()

B. a_sort[i+1].size()

C. min(a_sort[i].size(), a_sort[i+1].size())

D. max(a_sort[i].size(), a_sort[i+1].size())

39. ③处应填（　　）。

A. ve[a_sort[i][j]-'a'].push_back(a_sort[i][j]-'a')

B. ve[a_sort[i][j]-'a'].push_back(a_sort[i+1][j]-'a')

C. ve[a_sort[i+1][j]-'a'].push_back(a_sort[i][j]-'a')

D. ve[a_sort[i+1][j]-'a'].push_back(a_sort[i+1][j]-'a')

40. ④处应填（　　）。

A. a_sort[i].size()>=a_sort[i+1].size()

B. a_sort[i].size()<=a_sort[i+1].size()

C. a_sort[i].size()>a_sort[i+1].size()

D. a_sort[i].size()<a_sort[i+1].size()

41. ⑤处应填（　　）。

A. !du[ve[v][i]]　　B. du[ve[v][i]]

C. ~du[ve[v][i]]　　D. -du[ve[v][i]]