CSP-S1 模拟赛

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 Linux 系统终端中，以下哪个命令用于创建一个新文件？（ ）。

   A. `new`

   B. `touch`

   C. `mkdir`

   D. `makefile`

2. 从 $0, 1, 2, 3, 4, 5$ 这些数字中选出不同的三个数组成一个三位数，有多少种方法（ ）。

   A. 80

   B. 100

   C. 120

   D. 180

3. 在有 n 个元素的二叉搜索树中进行查找，其最好、最差时间复杂度分别为（ ）。其中最好时间复杂度的定义为在树的形态和本次查找的元素这两个因素均为最好状态下的时间复杂度，最差时间复杂度的定义为在树的形态和本次查找的元素这两个因素均为最差状态下的时间复杂度。

   A. $\mathcal{O}(1), \mathcal{O}(\log n)$

   B. $\mathcal{O}(1), \mathcal{O}(n)$

   C. $\mathcal{O}(\log n), \mathcal{O}(\log n)$

   D. $\mathcal{O}(\log n), \mathcal{O}(n)$

4. 已知袋中有 2 个相同的红球、3 个相同的绿球、5 个相同的黄球。每次取出一个不放回，全部取出。可能产生 多少种序列？（ ）

   A. 6

   B. 1440

   C. 2520

   D. 3628800

5. 小于或等于给定正整数 n 的数中，与 n 互质的数的个数，我们称为欧拉函数，记作 $\phi(n)$。下面说法错误的是 （ ）

   A. 如果 n 是质数，那么 $\phi(n) = n - 1$。

   B. 两个质数一定是互质数。

   C. 两个相邻的数一定是互质数。

   D. $\phi(1) = 0$

6. 使用邻接矩阵存储包含 n 个点的有向图，则该矩阵的大小为（ ）。

   A. $n \times (n + 1)$

   B. $n \times n$

   C. $n \times (n - 1)$

   D. $\frac{n \times (n-1)}{2}$

7. 设 A 和 B 是两个字符串。我们要用最少的操作次数，将字符串 A 转换为字符串 B。这里所说的操作共有三种：

   - 删除一个字符；
   - 插入一个字符；
   - 将一个字符改为另一个字符。

   我们称该操作次数为这两个字符串的编辑距离。

   请问 `sfdqxbw` 和 `gfxdgbw` 的编辑距离为 （ ）

   A. 1

   B. 2

   C. 3

   D. 4

8. 一对夫妻生男生女的概率相同。这对夫妻有两个孩子，已知其中有一个是女孩，但并不知道这个女孩是姐姐还是妹妹，则另一个是男孩的概率是（ ）

   A. $\frac{1}{4}$

   B. $\frac{1}{3}$

   C. $\frac{1}{2}$

   D. $\frac{2}{3}$

9. 在 C++ 等语言的有些编译器中，对逻辑表达式的计算会采用一种“短路”的策略：在形如 `a&b` 的逻辑表达式中，会先计算 `a` 部分的值，如果 $a = 0$，那么整个逻辑表达式的值就一定为 0，故无需再计算 `b` 部分的值；同理，在形如 `a|b` 的逻辑表达式中，会先计算 `a` 部分的值，如果 $a = 1$，那么整个逻辑表达式的值就一定为 1，无需再计算 `b` 部分的值。请问在计算 `(0|1&0|1|1|(1|1))&(0&1&(1|0)|0|1|0)&0` 这个逻辑表达式时，`a&b` 类型的“短路”和 `a|b` 类型的“短路”各发生了多少次（ ）

   A. $2\ 2$

   B. $2\ 3$

   C. $3\ 2$

   D. $3\ 3$

10. 下列排序算法中所需额外空间的数量级为 $\mathcal{O}(1)$ 的是（ ）

A. 插入排序

B. 计数排序

C. 归并排序

D. 快速排序

11. 以下哪个编译选项，能在使用 `g++` 编译 C++ 源文件时打印警告信息（ ）

    A. `-o`

    B. `-g`

    C. `-w`

    D. `-wall`

12. 以下哪个选项不是树的重心具有的性质（ ）

A. 如果在树中选择某个节点并删除，这棵树将分为若干棵子树，统计子树节点数并记录最大值，取遍树上所有节点，树的重心是使此最大值取到最小的节点

B. 以树的重心为根时，所有子树的大小都不超过整棵树大小的一半

C. 如果在树中选择某个节点，计算所有节点到该节点的距离并记录最大值，取遍树上所有节点，树的重心是使此最大值取到最小的节点

D. 树中所有点到某个点的距离和中，到重心的距离和是最小的

13. 包含 n 个节点且无重边的有向无环图中最多可以有（ ）条边。

A. $\frac{n\times(n-1)}{2}$

B. $n \times (n-1)$

C. $n-1$

D. $2n$

14. 将 $(x+y)^{10}$ 展开为 $\sum_{i=0}^{10} a_i \times x^i \times y^{10-i}$，其中 a_5 的值为（ ）

A. 5

B. 252

C. 30240

D. 63504

15.
```
bool notPrime[N] = {false};
void sieve() {
    for (int n = 2; n * n < N; n++)
        if (!notPrime[n])
            for (int i = n * n; i < N; i += n)
                notPrime[i] = true;
}
```

该函数的时间复杂度为（ ）

A. $\mathcal{O}(n \log \log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n\sqrt{n})$

D. $\mathcal{O}(n^2)$

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

(1)

```
#include <iostream>
using namespace std;
int a[100005];
int cnt[2];
int main()
{
    int n;
    long long now,ans;
    cin>>n;
```

```
10      for(int i=1;i<=n;i++)
11          cin>>a[i];
12      for(int i=1;i<=n;i++)
13          a[i]^=a[i-1];
14      ans=0;
15      for(int i=30;i>=0;i--)
16      {
17          cnt[0]=cnt[1]=0;
18          now=0;
19          for(int j=0;j<=n;j++)
20          {
21              now+=cnt[((a[j]>>i)&1)^1];
22              cnt[(a[j]>>i)&1]++;
23          }
24          ans+=(now<<i);
25      }
26      cout<<ans<<endl;
27      return 0;
28  }
```

假设输入的 n 为不超过 10^5 的正整数，$a[i]$ 为不超过 10^9 的非负整数，完成下面的判断题和单选题：

判断题

16. （2 分）在此数据范围下，ans 在计算过程中可能发生溢出（ ）

    T. 正确

    F. 错误

17. 将第 15 行的 `i = 30` 改为 `i = 29`，程序的输出一定不变（ ）

    T. 正确

    F. 错误

18. 将第 19 行的 `j = 0` 改为 `j = 1`，程序的输出一定不变（ ）

    T. 正确

    F. 错误

19. 本程序在计算 a 数组的所有子序列的异或和的和（ ）

    T. 正确

    F, 错误

单选题

20. 当输入为 `5 1 2 3 4 5` 时，输出为（ ）

    A. 19

    B. 30

    C. 39

    D. 112

21. 在此数据范围下，`now` 可以达到的最大值为（ ）

A. 2500000000

B. 2500050000

C. 5000050000

D. 10000100000

(2)

```cpp
#include <iostream>
using namespace std;
int p[35],a[35],cnt;
long long ans;
int n,nn;
void dfs(int pos,int now,int phi)
{
    if(pos==cnt)
    {
        ans+=1LL*phi*(n/now);
        return;
    }
    for(int i=0;i<=a[pos];i++)
    {
        dfs(pos+1,now,phi);
        now*=p[pos];
        if(i==0)
            phi*=(p[pos]-1);
        else phi*=p[pos];
    }
}
int main()
{
    cin>>n;
    if(n==1)
    {
        cout<<"1"<<endl;
        return 0;
    }
    nn=n;
    for(int i=2;i*i<=nn;i++)
        if(nn%i==0)
        {
            p[cnt]=i;
            while(nn%i==0)
            {
                nn/=i;
                a[cnt]++;
            }
            cnt++;
        }
    if(nn!=1)
    {
        p[cnt]=nn;
        a[cnt]=1;
        cnt++;
    }
```

```
48      dfs(0,1,1);
49      cout<<ans<<endl;
50      return 0;
51  }
```

假设输入的 n 是不超过 10^9 的正整数，完成下面的判断题和单选题：

判断题

22. 将第 25 行到第 29 行删去，输入 1 时，程序输出不变（ ）

    T. 正确

    F. 错误

23. 将第 31 行的 `i*i<=nn` 改为 `i<=n/i`，程序输出不变（ ）

    T. 正确

    F. 错误

24. （2 分）进入代码第 44 行时，nn 的值一定大于 $\sqrt{n}$（ ）

    T. 正确

    F. 错误

单选题

25. 程序第 31 行到第 41 行的时间复杂度为（ ）

    A. $\mathcal{O}(\log n)$

    B. $\mathcal{O}(\sqrt{n})$

    C. $\mathcal{O}(\sqrt{n} \times \log n)$

    D. $\mathcal{O}(n)$

26. 设 n 的因数数量为 $d(n)$，质因数数量为 $\omega(n)$，则 `dfs` 函数的时间复杂度可以表示为（ ）

    A. $\mathcal{O}(d(n))$

    B. $\mathcal{O}(\omega(n))$

    C. $\mathcal{O}(d(n) \times \omega(n))$

    D. $\mathcal{O}(d(n) \times \omega(n) \times \log n)$

27. 当输入为 2024 时，输出为 （ ）

    A. 2024

    B. 4320

    C. 18900

    D. 32384

(3)

```
1  #include <iostream>
2  #include <algorithm>
```

```cpp
using namespace std;
int a[100005];
int sum[100005];
long long sum2[100005];
int n;
long long k;
pair<long long,long long> check(int x)
{
    int pos=0;
    pair<long long,long long> ret=make_pair(0,0);
    for(int i=1;i<=n;i++)
    {
        while(sum[i]-sum[pos]>x)
            pos++;
        ret.first+=i-pos;
        ret.second+=1LL*(i-pos)*sum[i]-(sum2[i]-sum2[pos]);
    }
    return ret;
}
int main()
{
    int l,r,mid;
    pair<long long,long long> ans;
    cin>>n>>k;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    for(int i=1;i<=n;i++)
        sum[i]=sum[i-1]+a[i];
    for(int i=1;i<=n;i++)
        sum2[i]=sum2[i-1]+sum[i-1];
    l=1;
    r=sum[n];
    while(l<r)
    {
        mid=(l+r)/2;
        ans=check(mid);
        if(ans.first>=k)
            r=mid;
        else l=mid+1;
    }
    ans=check(r);
    cout<<ans.second-(ans.first-k)*r<<endl;
    return 0;
}
```

假设输入的 n 为不超过 10^5 的正整数，输入的 a 均为不超过 10^4 的正整数，完成下面的判断题和单选题：

判断题

28. 将第 43 行的语句移动到第 34 行之后第 35 行之前，程序的输出一定不变（ ）

T. 正确

F. 错误

29. 将第 35 行的 `mid = (l + r) / 2` 改为 `mid = (l + r + 1) / 2`，程序肯定可以正常允许（ ）

T. 正确

F. 错误

30. 输入 `4 4 3 2 1 5`，输出为 9（ ）

T. 正确

F. 错误

单选题

31. `check` 函数的时间复杂度为 （ ）

A. $\mathcal{O}(n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n \log a)$

D. $\mathcal{O}(n^2)$

32. 将第 39 行的 `ans.first>=k` 改为 `ans.first>k`，该程序的输出（ ）

A. 一定不变

B. 可能不变，可能变大，不可能变小

C. 可能不变，可能变小，不可能变大

D. 可能不变，可能变小，也可能变大

33. 当输入为 `6 10 1 2 3 6 5 4` 时，输出为 （ ）

A. 35

B. 44

C. 53

D. 75

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）（最长旅途）

Bessie 正在奶牛大陆上旅行。奶牛大陆由从 1 到 N 编号的 N 座城市和 M 条单向道路组成。第 i 条路从城市 u_i 通向城市 v_i，标签为 w_i。

由城市 x_0 开始的长度为 k 的旅程被定义为一个城市序列 $x_0, x_1, \ldots, x_k$，对于所有的 $0 \leq i < k$，存在由城市 x_i 到 x_{i+1} 的路。保证在奶牛大路上不存在长度无限的旅程，不存在两条路连接一对相同的城市。

对于每座城市，Bessie 想知道从它开始的最长旅程。对于一些城市，从它们开始的最长旅程不唯一，Bessie 将选择其中道路标签序列字典序更小的旅程。一个序列比等长的另一个序列字典序更小，当且仅当在它们不同的第一个位置，前者比后者的元素更小。

输出 Bessie 在每座城市选择的旅途的长度和道路标签之和。

```
1 #include <iostream>
```

```cpp
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;
const int inf=0x3f3f3f3f;
vector<pair<int,int> > mp1[200005],mp2[200005];
int indeg[200005];
int dp[200005];
long long ans[200005];
queue<int> q;
int n,m,u,v,w;
vector<int> level[200005];
void topo()
{
    for(int i=1;i<=n;i++)
        if(!indeg[i])
            q.push(i);
    while(!q.empty())
    {
        int now=q.front();
        q.pop();
        for(int i=0;i<mp2[now].size();i++)
        {
            indeg[mp2[now][i].first]--;
            __1__;
            if(!indeg[mp2[now][i].first])
                q.push(mp2[now][i].first);
        }
    }
}
pair<int,int> p[200005];
pair<int,int> tmp;
int now;
int pos[200005];
bool cmp(int a,int b)
{
    return __2__;
}
int main()
{
    cin>>n>>m;
    for(int i=0;i<m;i++)
    {
        cin>>u>>v>>w;
        mp1[u].push_back(make_pair(v,w));
        mp2[v].push_back(make_pair(u,w));
        indeg[u]++;
    }
    topo();
    for(int i=1;i<=n;i++)
        level[dp[i]].push_back(i);
    for(int i=1;i<=n;i++)
    {
        if(!level[i].size())
            break;
        for(int j=0;j<level[i].size();j++)
```

```
        {
            now=level[i][j];
            p[now].first=inf;
            for(int k=0;k<mp1[now].size();k++)
            {
                if(dp[mp1[now][k].first]!=dp[now]-1)
                    continue;
                tmp.first=mp1[now][k].second;
                tmp.second=pos[mp1[now][k].first];
                if(__3__)
                {
                    p[now]=tmp;
                    __4__;
                }
            }
        }
        sort(level[i].begin(),level[i].end(),cmp);
        for(int j=0;j<level[i].size();j++)
            __5__;
    }
    for(int i=1;i<=n;i++)
        cout<<dp[i]<<" "<<ans[i]<<endl;
    return 0;
}
```

34. `__1__` 处应填（ ）

    A. `dp[mp2[now][i].first]=dp[now]`

    B. `dp[mp2[now][i].first]=dp[now]+1`

    C. `dp[mp2[now][i].first]=max(dp[mp2[now][i].first],dp[now])`

    D. `dp[mp2[now][i].first]=max(dp[mp2[now][i].first],dp[now]+1)`

35. `__2__` 处应填（ ）

    A. `a<b`

    B. `a>b`

    C. `p[a]<p[b]`

    D. `p[a]>p[b]`

36. `__3__` 处应填（ ）

    A. `tmp.first<p[now].first`

    B. `tmp.first>p[now].first`

    C. `tmp<p[now]`

    D. `tmp>p[now]`

37. `__4__` 处应填（ ）

    A. `ans[now]=tmp.first+ans[level[i-1][tmp.second]]`

    B. `ans[now]=mp1[now][k].first+ans[level[i-1][tmp.second]]`

C. `ans[now]+=tmp.first`

D. `ans[now]+=ans[level[i-1][tmp.second]]`

38. `__5__` 处应填（ ）

A. `pos[j]=level[i][j]`

B. `pos[level[i][j]]=j`

C. `level[i][j]=j`

D. `level[i][j]=pos[j]`

（2）（最小值和）

给定一个长度为 n 的正整数数组 a，下标从 1 开始

求其每个连续区间的最小值之和

即求 $\sum_{i=1}^{n}\sum_{j=i}^{n}\min(a_i,a_{i+1},\ldots,a_j)$

本题有很多种做法，这里使用一种时间复杂度为 $\mathcal{O}(n\log n)$ 的分治做法

```
#include <iostream>
#include <algorithm>
using namespace std;
int a[100005],lg[100005],st[100005][20],p[100005][20];
pair<int,int> query(int l,int r)
{
    int t=lg[r-l+1];
    if(st[l][t]<=st[r-(1<<t)+1][t])
        return make_pair(st[l][t],p[l][t]);
    else return make_pair(st[r-(1<<t)+1][t],p[r-(1<<t)+1][t]);
}
long long solve(int l,int r)
{
    if(__1__)
        return 0;
    pair<int,int> mn=query(l,r);
    long long left=__2__;
    long long right=__3__;
    return left+right+1LL*mn.first*__4__;
}
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    for(int i=2;i<=n;i++)
        lg[i]=lg[i>>1]+1;
    for(int i=1;i<=n;i++)
    {
        st[i][0]=a[i];
        p[i][0]=i;
    }
    for(int i=1;i<20;i++)
```

```
35          for(int j=1;j+(1<<i)-1<=n;j++)
36              if(st[j][i-1]<=st[j+(1<<(i-1))][i-1])
37              {
38                  st[j][i]=st[j][i-1];
39                  __5__;
40              }
41              else
42              {
43                  st[j][i]=st[j+(1<<(i-1))][i-1];
44                  p[j][i]=p[j+(1<<(i-1))][i-1];
45              }
46      cout<<solve(1,n)<<endl;
47      return 0;
48  }
```

39. `__1__` 处应填（ ）

    A. `l>r`

    B. `l>=r`

    C. `l==r`

    D. `l!=r`

40. `__2__` 处应填（ ）

    A. `solve(l,mn.first-1)`

    B. `solve(l,mn.first)`

    C. `solve(l,mn.second-1)`

    D. `solve(l,mn.second)`

41. `__3__` 处应填（ ）

    A. `solve(mn.first,r)`

    B. `solve(mn.first+1,r)`

    C. `solve(mn.second,r)`

    D. `solve(mn.second+1,r)`

42. `__4__` 处应填（ ）

    A. `(r-mn.second)*(mn.second-l)`

    B. `(r-mn.second+1)*(mn.second-l)`

    C. `(r-mn.second)*(mn.second-l+1)`

    D. `(r-mn.second+1)*(mn.second-l+1)`

43. `__5__` 处应填（ ）

    A. `p[j][i]=j`

    B. `p[j][i]=p[j][i-1]`

    C. `p[j][i]=p[j-1][i]`

D. `p[j][i]=j+(1<<(i-1))-1`