「比赛总结」正睿 提高十连 Day1

Jiayi Su (ShuYuMo)

2020-08-31 08:21:06

2020 第一次正睿提高十连测，题目很妙。

Problems

A 将 n 个石子分成 x 堆，使每一堆的石子个数都形如 $ 3k^2 - 3k + 1$ ($k \leq 1$)。最小化 x 输出 x。

$n \leq 10^{11}$

B 给出一个字符串 S ，求出字符串中，有多少连续的子串形如 AA_rA 。

其中 A_r 表示 A 的反串。合法的相同字串出现多次，统计多次。

$|S| \leq 2 \times 10^6$

C 给出一个 n 个点的树（保证这棵树随机生成），点有点权，规定 1 号点为根节点。

随机生成一个 n 排列，按这个排列的顺序依次访问所有树上的节点。每次访问到一个结点 x 时，需要将包括 x 在内的所有 x 子树中的点的点权 V 加上 x 点的当前权值 V_x'。

求出最终所有点权和的期望值。

$n \leq 10^5$

Problem A Stone

Statement

将 n 个石子分成 x 堆，使每一堆的石子个数都形如 $ 3k^2 - 3k + 1$ ($k \leq 1$)。最小化 x 输出 x。

$n \leq 10^11$

Analysis

引理 **1** 对于任意一个正整数 x，均可以被拆分为 3 个形如 $\binom{n}{2}$ 的数字。

引理 **2** 可以证明 x 一定小于等于 8。

证明 引理 **2** 我们需要的数字是 $3k^2 - 3k + 1 = 6 \times \dfrac{n(n+1)}{2} + 1 = 6 \times \binom{n}{2} + 1.$

考虑一定存在 $N - k$ 是 6 的倍数。所以 $N - k + 3$ 必然可以写成 三个形如 $3k^2 - 3k + 1$ 数的和的形式.

而 $N - k + 3$ 最多和 N 相差 5 所以最多补 5 个 1 就可以凑出 N.

引理 **3** $ans \equiv N \pmod 6$

证明 引理 **3** 因为 $\sum 3k^2-3k+1=\sum 6\times\frac{n(n+1)}{2}+1=\sum 6\times\binom{n}{2}+1.$

显然 $\sum 6\times\binom{n}{2}+1\equiv 1\pmod 6$.

易知 $ans\equiv N\pmod 6$

只需要判断答案是 1 还是 7，是 2 还是 8。- 判断是否为 1 直接判断即可。- 判断是否为 2 双指针扫一遍即可。

Problem B Palindrome

Statement

给出一个字符串，求出字符串中，有多少连续的子串形如 AA_rA 。

其中 A_r 表示 A 的反串。合法的相同字串出现多次，统计多次。

$|S|\le 2\times 10^6$

Analysis

先用各种神奇的算法（字符串 Hash 、manacher）求出 len[i] 表示第 i 个字符和第 $i+1$ 个字符之间为中心，向两边扩展最长的回文串.

枚举 AA_rA 的 $\frac{1}{3}$ 位置 i ，在 $[i+1,i+\text{len}[i]]$ 这个区间中有多少 x 满足 $x-\text{len}[x]\le i$，每一个合法的 x 给答案贡献 1.

转化为 问题.

可以采用 或者 后采用 扫描.

Problem C Random

给出一个 n 个点的树（保证这棵树随机生成），点有点权，规定 1 号点为根节点。

随机生成一个 n 排列，按这个排列的顺序依次访问所有树上的节点。每次访问到一个结点 x 时，需要将包括 x 在内的所有 x 子树中的点，点权加 V_x。

$n\le 10^5$

关于期望思考 期望的：标志着很多情况下的期望可以分开算，算完后再合并。例如：点权和的期望 = 点权的期望和。

直观感受一下：期望是一个随机变量的平均情况下的数值。

我们不知道这个随机变量 x 确切的数值，自然也不知道由这个随机变量推导出来的另一个随机变量 X（例如：如果我们不知道 每个点 操作后的 确切点权，那么所有点的点权和的确切数值也无从得知。）

但是如果我们知道每个点在 下取到的值 x，我们一样可以通过随机变量 x 下取到的值推出 X 在 下取到的值。

关于本题，我们同样可以拆开来看，要求 求出结点的期望和，那我们可以求出每个点的期望权值，然后再相加就能得到所有结点的期望和。

对于一个有根树上的每一个结点，可能能对这个结点的权值产生贡献的 它的祖先。

这个点 a 最终权值的期望 $V_a^{'}$ 就是点 a 的初始权值和所有 a 的祖先对点 a 的期望贡献之和。

a 的祖先 p 对 a 的期望贡献可以拆成 $V_p^{'}$ 与 对 a 的贡献次数的乘积。

对于每一条树链上的任意一个点 x 来说某一个祖先 p 对其贡献的次数，只与这个祖先 p 和这个点 x 的距离有关。

不妨定义 $\text{dp}[i]$ 为结点 1 到往下的第 i 个点的期望贡献次数（: 第 1 个点对第 1 个点的贡献次数定义为 $\text{dp}[1]$ ）。

对于每一条树链，我们都可以采用同一套 dp 值进行处理，因为对于每个点来说，其祖先的对其贡献次数只与他们之间的相对位置有关。

考虑如何预处理出 $\mathrm{dp}[i]$：

考虑结点 1 如何对其往下第 $i-1$ 结点 x 产生贡献。(不妨设：这一条树链上结点的编号由 依次为 $1-k$)。

考虑贡献的传递过程，结点 1 的贡献传给 x 有两种情况：

- 由 1 直接传递给 x
- 由 1 传递给 x 的中间的若干结点，经过多次传递后传递给 x 。

这些方案对应的贡献为 1 ，所以只要求出方案数量，就可以求出结点 1 对 x 的贡献次数。

考虑这些方案对应着什么——n 的排列中 以 1 开始的上升序列的个数。

于是，$\mathrm{dp}[i]$ 就等于在 n 的排列中 以 1 开始的上升序列的个数。

枚举一个上升序列的长度 d ，统计其数量即可

$$\mathrm{dp}[i]=\sum_{d=1}^{i}\binom{i-1}{d-1}\times\binom{i}{d}\times(i-d)!$$

其中：- $\binom{i-1}{d-1}$ 指在 $i-1$ 中选定 $d-1$ 个数字。

- $\binom{i}{d}$ 指在 i 个位置中选 d 个将这 d 个数字放下。
- $(i-d)!$ 指 剩下的 $(i-d)$ 个位置的数字任意排列。

最后需要注意一点，因为树是随机生成的，其实这就保证了树的高度（树链的最长长度）在 $O(\sqrt{n})$ 级别，预处理和统计答案相当于是线性的。

Codes

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <cmath>
using namespace std;
const int _ = 1e7 + 100;
#define LL long long
LL read(){ LL x; scanf("%lld", &x); return x; }
LL A[_], Q;
void init(){
    for(int i = 1; ; i++) { A[i] = ( (i) *1ll* (i - 1) ) >> 1; if(A[i] > 1e11) break; else Q = i; }
}
int doit(LL x){
    if(x % 6 != 1 && x % 6 != 2) return x % 6 == 0 ? 6 : x % 6;
    if(x % 6 == 1){
        x = (x - 1) / 6;
        if( ( *lower_bound(A + 1, A + Q + 1, x) ) == ( x ) ) return 1;
        else return 7;
    } else {
```

```
        x = (x - 2) / 6;
        int L = 1, R = Q;
        while(L <= R){
            if(A[L] + A[R] == x) return 2;
            while(A[R] + A[L] < x) L++;
            while(A[R] + A[L] > x) R--;
        }
        return 8;
    }
}
int main(){
    int T = read();
    init();
    while(T--) printf("%d\n", doit(read()));
    return 0;
}
```

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <cmath>
#include <ctime>
#include <algorithm>
using namespace std;
const int _ = 2e7 + 100;
#define LL long long
int read(){ int x; scanf("%d", &x); return x; }
char S[_];
int  n;
namespace Hash{
    const int MOD  = 998244353;
    const int base = 'z' + 10;
    int ahs[_];
    int bhs[_];
    int POW[_];
    int pow(int a, int b, int p){ int ans = 1; while(b) { if(b & 1) ans = (ans *1LL* a) % p; a = (a *1l
    int suba(int L, int R) { return (  ahs[R] - (ahs[L - 1] *1ll* POW[R - L + 1] % MOD)  +0ll+ MOD) % M
    int subb(int L, int R) { return (  bhs[L] - (bhs[R + 1] *1ll* POW[R - L + 1] % MOD)  +0ll+ MOD) % M
    void initHash(int x){
        POW[0] = 1;
        for(int i = 1; i <= x; i++) POW[i] = POW[i - 1] *1ll* base % MOD;
        for(int i = 1; i <= x; i++){
            ahs[i] = ( (ahs[i - 1] *1ll* base) +0ll+ S[i] ) % MOD;
        }
        for(int i = x; i >= 1; i--){
            bhs[i] = ( (bhs[i + 1] *1ll* base) +0ll+ S[i] ) % MOD;
        }
```

```
    }
}
using Hash::initHash;
using Hash::suba;
using Hash::subb;
int mlen[_];
int qury[_];

int main(){
    clock_t t0 = clock();
    freopen("in.txt", "r", stdin);
    scanf("%s", S + 1); n = strlen(S + 1);
    initHash(n);
    for(int i = 1; i < n; i++){
        if(S[i] != S[i + 1]) { mlen[i] = 0; continue; }
        int L = 0;
        int R = i;
        int ans = 0;
        while(L < R){
            int mid = L + ((R - L + 1) >> 1);
            if(suba(i - mid + 1, i) == subb(i + 1, i + mid)) ans = mid, L = mid;
            else R = mid - 1;
        }
        mlen[i] = ans;
    }

    for(int i = 1; i < n; i++) qury[i] = i - mlen[i];

    int ans = 0;
    for(int i = 1; i < n; i++){
        for(int j = i + 1; j <= i + mlen[i];j++) ans += (j - mlen[j] <= i);
    }
    printf("%d", ans);
    return 0;
}
```

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <stack>
#include <iostream>
#define MOD 1000000007
using namespace std;
int read(){ int x; scanf("%d", &x); return x; }
const int _ = 1e5 + 100;
const int MAXD = 1e3 + 100;
int head[_];
```

```cpp
struct edges{
    int node;
    int nxt;
}edge[_ << 1];
int tot = 0;
void add(int u, int v){
    tot++;
    edge[tot].node = v;
    edge[tot].nxt  = head[u];
    head[u]        = tot;
}
int n;
int rt = 1;
int C[MAXD + 100][MAXD + 100];
int frc[_];
int dp[_];
int pow(int a, int b, int P)  { int ans = 1; while(b) { if(b & 1) ans = (ans *1ll* a) % P; a = (a *1ll*
int inv(int x){ return pow(x, MOD - 2, MOD); }
const int DB = 20;
void init(){
    C[0][0] = 1;
    for(int i = 1; i <= MAXD; i++){
        C[i][0] = 1;
        for(int j = 1; j <= i; j++){
            C[i][j] = ( C[i - 1][j] +0ll+ C[i - 1][j - 1] ) % MOD;
        }
    }
    frc[0] = 1;
    for(int i = 1; i <= _ - 10; i++) frc[i] = ( frc[i - 1] *1ll* (i) ) % MOD;
    for(int i = 1; i <= MAXD; i++){
        dp[i] = 0;
        for(int d = 1; d <= i; d++){
            dp[i] = (dp[i] +0ll+ (
                (  ( (C[i - 1][d - 1] *1ll* C[i][d]) % MOD ) *1ll* frc[i - d]  ) % MOD
                    ) % MOD) % MOD;
        }
    }
    for(int i = 1; i <= MAXD; i++) dp[i] = (dp[i] *1ll* inv(frc[i])) % MOD;
}
int ans = 0;
struct Stack{ int v[_]; int top; Stack(){ top = 0; } inline int *GetHead(){ return &v[0]; } inline int*
Stack S;
int val[_];
int NodeVal[_];
void dfs0(int o, int fa, int dep){
    S.push(o);
    int now = dep;
```

```
    for(int *i = S.GetHead(); i != S.GetTail(); i++){
        val[o] = (  val[o] +0ll+ (dp[now] *1ll* NodeVal[*i]) % MOD  ) % MOD;
        now --;
    }
    for(int i = head[o]; i ; i = edge[i].nxt){
        int ex = edge[i].node;
        if(ex == fa) continue;
        dfs0(ex, o, dep + 1);
    }
    S.pop();
}
int main(){
    n = read();
    init();
    for(int i = 1; i < n; i++){ int u = read(), v = read(); add(u, v); add(v, u); }
    for(int i = 1; i <= n; i++) NodeVal[i] = read();
    dfs0(1, -1, 1);
    int ans = 0;
    for(int i = 1; i <= n; i++) ans = (ans +0ll+ NodeVal[i]) % MOD;
    for(int i = 1; i <= n; i++) ans = (ans +0ll+ val[i]) % MOD;
    printf("%d\n", (ans *1ll* frc[n]) % MOD);
    return 0;
}
```