「比赛总结」正睿 国庆

Jiayi Su (ShuYuMo)

2020-11-18 11:41:50

link

给出一个长度为 n 的 01 串 s 和整数 k ，求一个 s 最长的子串 t 使得 t 中 0 的个数是 1 的个数的 k 倍，输出最长的 t 的长度。

设 $ S_0[i]$ 表示 $S[1..i]$ 中 0 的数量。$S_1[i]$ 表示 $S[1..i]$ 中 1 的数量。字串 $S[L,R]$ 满足条件当且仅当 $S_0[R]-S_0[L-1]=(S_1[R]-S_1[L-1])\times k$。即：$S_0[R]-S_1[R]\times k=S_0[L-1]-S_1[L-1]\times k$。

```
#define i64 long long
const int _ = 1e6 + 100;
int n, k;
char S[_];
int S0[_], S1[_];
map<long long, int >M;
int main(){
    ios::sync_with_stdio(false);
    cin >> n >> k >> (S + 1);
    for(int i = 1; i <= n; i++) S0[i] = S0[i - 1] + (S[i] == '0'), S1[i] = S1[i - 1] + (S[i] == '1');
    int ans = 0;
    M[0] = 0;
    for(int i = 1; i <= n; i++){
        i64 now = S0[i] -0ll- k *1ll* S1[i];
        if(M.count(now)) ans = max(ans, i - M[now]);  else M[now] = i;
    }
    printf("%d\n", ans); cerr << "std's ans = " << ans << endl;
    return 0;
}
```

有两个长度为 n 的排列 A,B ，你每次可以进行三种操作：- 删除 A 的第一个元素 a - 删除 B 的第一个元素 b - 删除 A 的第一个元素 a 和 B 的第一个元素 b ，要求 $a\neq b$ 求把 A,B 都删光需要的最少操作次数。

$n\leq 10^6$

当数列 A,B 的首项相同的时候，直接贪心的选择操作 3 。需要决策当 A,B 首项相同的时候选择删除那边的，然后又可以一直删，直到首项再次相同的时候 再决策，直接 DP 可能被卡住的地方即可。

设数字 x 在 A 中的位置是 i，在 B 中的位置是 j，记 数字 x 的位置为二元组 (i,j)。由于 A,B 是两个排列，对于数字 x ，其位置一定是唯一的 (i,j)。设 $\mathrm{dp}[i]$ 为 (设 A_i 的位置为 (i,j)) ，删光 $A[i..n]$ 和 $B[j..n]$ 的代价。

对于数值 x 的位置 (i,j) 设 $ = i - j 　　dp[i] A_i$ 的位置 (i,j) 这些是一一对应的。设 $\Delta(x)$ 为 $dp[x]$ 对应的 Δ。

转移就是

$dp[i] = 1 + \min_{\Delta(x)=\Delta(i)-1,\Delta(y)=\Delta(i)+1} \{\ dp[x] + \mathrm{dist}(i, x),\quad dp[y] + \mathrm{dist}(i, y)\ \}$。

从后往前 dp 记录以下 Δ 即可。为了方便，其中 Last 为 $\Delta()$ 的反函数。

```
int n, A[_], B[_], POOL[(_ << 1) + 100], PosInB[_], *Last = &POOL[_ + 10];
int dp[_];
int main(){
    rep(i, 1, n = read()) Read(A[i]); rep(i, 1, n) Read(B[i]), PosInB[B[i]] = i;
    clear(dp, 0x3f); clear(POOL, -1);
    per(i, 1, n) {
        int det = i - PosInB[A[i]];
        if(Last[det + 1] != -1) to_min(dp[i], dp[Last[det + 1]] + 1 + (Last[det + 1] - (i + 1)));
        else to_min(dp[i], 1 + max(n - (i + 1) + 1, n - (PosInB[A[i]]) + 1 ));
        if(Last[det - 1] != -1) to_min(dp[i], dp[Last[det - 1]] + 1 + (Last[det - 1] - (i)));
        else to_min(dp[i], 1 + max(n - (i) + 1, n - (PosInB[A[i]] + 1) + 1));
        Last[det] = i;
    }
    int ans = 0;
    if(Last[0] == -1) ans = n; else ans = dp[Last[0]] + Last[0] - 1;
    printf("%d", ans);
    return 0;
}
```

对于长度为 n 的置换 A, B ，求是否存在正整数 k 使得 $A^k = B$

定义置换的乘法为 $C = (A \cdot B), C_i = A_{B_i}$

定义 $A^1 = A, A^n = A^{n-1} \cdot A(n > 1)$

如果存在 k 输出 Yes 否则输出 No 。

$n \le 10^6$

可以转化为对线性同于方程判断是否有解的问题。

这里的线性同于方程组的模数 $\le 10^6$ 且不互质，LCM 很大 无法 $exCRT$ 合并。

$O(n \log n)$：从 1 n 枚举 i 尝试求出 $x \bmod i$ 的数值，易知 这个值可以从 $x \bmod k i$ 的值得到，检查方程组中所有的方程，看看是否冲突即可。根据调和级数，这样做的时间复杂度为 $O(n \log n)$。

```
#define i64 long long
#define walk(now, ex) for(int i = head[now], ex; ex = edge[i].node, i; i = edge[i].nxt)
bool vis[_];
vector<int> G[_];
int D[_], SZ[_], BL[_];
int Dis[_], MD[_];
int cnt = 0;
void clear(){
    memset(head, 0, sizeof(head)); tot = 0;
    memset(vis, false, sizeof(vis));
    for(int i = 1; i <= cnt; i++) G[i].clear(); cnt = 0;
}
void dfs(int now, int target){
```

```
    G[target].push_back(now);  vis[now] = 1;
    walk(now, ex) { if(vis[ex]) continue; dfs(ex, target); }
}
bool CMP(const pair<int, int > & A, const pair<int, int > & B) { return A.fi < B.fi; }
int pos[_];
bool PdExist(int *Md, int *a, int n) {
    static vector< pair<int, int > > M, MO; M.clear(); MO.clear();
    for(int i = 1; i <= n; i++) M.push_back(make_pair(Md[i], a[i]));
    sort(M.begin(), M.end(), CMP);
    for(int i = 0; i < M.size(); i++) {
        int L = i, R = i;
        while(R + 1 < M.size() && M[L].fi == M[R + 1].fi) R++;
        for(int j = L; j <= R; j++) if(M[j].se != M[L].se) return false;
        MO.push_back(M[L]);
        i = R;
    }
    memset(pos, -1, sizeof(pos));
    int MAX = 0; for(int i = 0; i < MO.size(); i++) MAX = max(MAX, MO[i].fi), pos[MO[i].fi] = i;
    for(int i = 1; i <= MAX; i++){
        int tmp = -1;
        for(int j = i; j <= MAX; j += i) if(pos[j] != -1) { if(tmp == -1) tmp = MO[pos[j]].se % i; else
    }
    return true;
}

void doit(){
    clear();
    for(int i = 1; i <= n; i++) add(A[i], i);
    for(int i = 1; i <= n; i++) if(!vis[i]) dfs(i, ++cnt);
    for(int i = 1; i <= cnt; i++) {
        for(int j = 0; j < G[i].size(); j++){
            D[G[i][j]] = j;
            SZ[G[i][j]] = G[i].size();
            BL[G[i][j]] = i;
        }
    }
    for(int i = 1; i <= n; i++) if(BL[A[i]] != BL[B[i]]) { return (void)puts("No");}
    for(int i = 1; i <= n; i++) Dis[i] = ( D[B[i]] - D[A[i]] + SZ[A[i]] ) % SZ[A[i]];

    for(int i = 1; i <= n; i++) MD[i] = SZ[A[i]];
    int r = PdExist(MD, Dis, n);
    return (void)puts(r ? "Yes" : "No");
}

int main(){
    while(scanf("%d", &n) == 1){
        for(int i = 1; i <= n; i++) Read(A[i]);
```

```
        for(int i = 1; i <= n; i++) Read(B[i]);
        doit();
    }
    return 0;
}
```