2023暑期CSP-S/NOIP模拟赛1 题解

T1 飞：

10分的做法：

考虑对每个风口，依次枚举对应的猪猪（匹配这一只猪猪，或者空着），判断其合法性。一直考虑到最后一只猪猪，计算答案，与历史的答案进行对比。

时间复杂度 $O(n!)$

40分做法：

考虑贪心，对于每个风口，按照从小到大的顺序排序，我们依次来考虑每个风口适合哪个猪猪。

与每个风口相关的，只有能够覆盖这个风口的区间。对于这些区间，由于左端点都已经满足小于风口，那么对于下一个风口，左端点一定满足条件。只需要考虑右端点。

对于右端点，最早失去匹配能力的，是右端点最靠左的，所以需要优先给右端点最靠右的匹配。

具体实现：先对左端点按照左端点排序，对每个风口，枚举一遍区间，选取能跨过这个风口的区间，选出其中的最小值，将对应区间标记为使用过。最后统计有多少个匹配成功。

时间复杂度: $O(nm)$

100分做法：

优化选取最小值的过程，我们可以维护一个优先对列，优先队列中每个节点维护右端点的值，每次取右端点最小的值出来，如果这个右端点还能覆盖，那么就使用这个右端点对应的区间。如果覆盖不了，就舍弃这个右端点对应的区间。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e4 + 10;
int n, m;
int p[maxn];
struct pig{
  int l, r;

  bool operator <(pig tmp) {
    return l < tmp.l;
  }
}a[maxn];

priority_queue<int, vector<int>, greater<int> > q;

int main() {
  ios::sync_with_stdio(0);
  cin.tie(NULL);
```

```
  cin >> n >> m;
  for(int i = 1; i <= n; ++i) cin >> p[i];
  for(int i = 1; i <= m; ++i) cin >> a[i].l >> a[i].r;
  sort(p + 1, p + 1 + n);
  sort(a + 1, a + 1 + m);
  int id = 1, ans = 0;
  for(int i = 1; i <= n; ++i) {
    while(id <= m && a[id].l <= p[i]) q.push(a[id++].r);
    while(!q.empty() && q.top() < p[i]) q.pop();
    if(!q.empty()) ++ans, q.pop();
  }
  cout << ans << '\n';
  return 0;
}

```

T2 魔法:

考虑将问题分解，因为宝石转换与具体的宝石序列无关，可以考虑先预处理出来所有宝石转换。

子问题1：已知一些宝石转换的权值，怎么得到最优的转换方式？

最短路的想法。

多源多汇最短路Floyd，时间复杂度$O(c^3)$ 。（c为宝石的种类数）

子问题2： 已知了两两之间转换的最小权值，怎么计算最优的转换模式？

考虑到问题的规模，可以直接对每个位置枚举转换为哪种颜色的宝石，时间复杂度 $O(n*c)$ 。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 10, inf = 0x3f3f3f3f;
int n, m;
int s[maxn], t[maxn];
int d[410][410];

int main() {
  ios::sync_with_stdio(0);
  cin.tie(NULL);
  cin >> n >> m;
  for(int i = 1; i <= n; ++i) cin >> s[i];
  for(int i = 1; i <= n; ++i) cin >> t[i];
  for(int i = 0; i <= 400; ++i) {
    for(int j = 0; j <= 400; ++j)
      d[i][j] = inf;
    d[i][i] = 0;
  }
  for(int i = 1; i <= m; ++i) {
    int x, y, c;
    cin >> x >> y >> c;
```

```
      d[x][y] = min(d[x][y], c);
    }
    for(int k = 1; k <= 400; ++k)
      for(int i = 1; i <= 400; ++i)
        for(int j = 1; j <= 400; ++j)
          d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
    int sum = 0, flag = 0;
    for(int i = 1; i <= n; ++i) {
      int x = s[i], y = t[i];
      if(x == y) continue;
      int dis = inf, z = -1;
      for(int j = 1; j <= 400; ++j) {
        if(d[x][j] + d[y][j] < dis) dis = d[x][j] + d[y][j], z = j;
      }
      if(z == -1) flag = 1;
      sum += dis;
    }
    if(flag) cout << "-1" << '\n';
    else cout << sum << '\n';
    return 0;
}
```

T3 地狱疣

20分：

模拟

20分特殊点：

只考虑第一部分的结果，由于最后的结果只关心求和，考虑推一下式子。

假设第 x 分钟时有 m 个地狱疣，地狱疣重量为 $a_1, a_2, \dots, a_m$ 总重量为 $\sum_{i=1}^{m} a_i$ 记作 sum_x, 那么 sum_0 就是开始时所有地狱疣的总重量。

我们考虑从 x 时刻到 $x+1$ 时刻：

$$\begin{aligned} \text{sum}_{x+1} &= \text{sum}_x + \sum_{i=1}^{m-1}(a_i + a_{i+1}) \\ &= \text{sum}_x + \sum_{i=1}^{m-1} a_i + \sum_{i=2}^{m} a_i \\ &= \text{sum}_x + 2 * \sum_{i=1}^{m} a_i - a_1 - a_m \\ &= \text{sum}_x + 2 * \text{sum}_x - a_1 - a_m \\ &= 3 * \text{sum}_x - a_1 - a_m \end{aligned}$$

这样我们就得到关于 sum_i 的递推式了。

通过矩阵快速幂优化递推的过程，时间复杂度可以做到 $O(log_2 N)$

具体的递推式为（其中 $num = a_1 + a_m$ ）

$$\begin{pmatrix} sum_{x+1} \\ num \end{pmatrix} = \begin{pmatrix} 3 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} sum_x \\ num \end{pmatrix}$$

100分做法：

当 t_2 不为0的时候，也就是 t_1 结束以后，会对所有的序列进行一个重新排序。

观察上面的式子，递推的过程仍然没有发生变化，但是 num 的值会发生变化，a_1 和 a_m 会变成 t_1 时刻结束时刻的最小值和最大值。

很明显，由于初始的序列有序，最小值仍然会是 a_1 ，但是最大值会发生变化。

通过手动模拟一下运算过程可以发现，a_m 一定是在 $t = 0$ 时刻的次大值和最大值之间。

令 p, q 分别为 $t = 0$ 时刻的次大值和最大值，手推一下 p, q 的序列有：

- $t = 0$ ，序列为 p, q
- $t = 1$ ，序列为 $p, p+q, q$
- $t = 2$ ，序列为 $p, 2p+q, p+q, p+2q, q$
- $t = 3$ ，序列为 $p, 3p+q, 2p+q, 3p+2q, p+q, 2p+3q, p+2q, p+3q, q$

最大项一定是中间的 q 系数大的那一项，且他的系数分别为 fib_{t-1} 和 fib_t 。

$$\begin{pmatrix} fib_{x+1} \\ fib_x \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} fib_x \\ fib_{x-1} \end{pmatrix}$$

最后使用矩阵快速幂计算一下就可以得到最大值

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e6 + 10;
typedef long long LL;

const LL mod = 998244353;
int n;
LL t1, t2;
int a[maxn];

inline int fadd(int x, int y) {
  return x + y >= mod ? x + y - mod : x + y;
}

struct Matrix {
  int a[2][2];

  void _set(int x) {
    a[0][0] = a[0][1] = a[1][0] = a[1][1] = x;
  }

  void _set_I() {
```

```
    a[0][1] = a[1][0] = 0;
    a[0][0] = a[1][1] = 1;
  }

  void init(int _00, int _01, int _10, int _11) {
    a[0][0] = _00, a[0][1] = _01, a[1][0] = _10, a[1][1] = _11;
  }

  Matrix operator +(const Matrix& B) {
    Matrix res;
    res._set(0);
    for(int i = 0; i <= 1; ++i)
      for(int j = 0; j <= 1; ++j)
        res.a[i][j] = fadd(a[i][j], B.a[i][j]);
    return res;
  }

  Matrix operator *(const Matrix& B) {
    Matrix res;
    res._set(0);
    for(int i = 0; i <= 1; ++i)
      for(int j = 0; j <= 1; ++j)
        for(int k = 0; k <= 1; ++k)
          res.a[i][j] = fadd(res.a[i][j], 1ll * a[i][k] * B.a[k][j] % mod);
    return res;
  }

  Matrix &operator += (Matrix B) {
    *this = *this + B;
    return *this;
  }

  Matrix &operator *= (Matrix B) {
    *this = *this * B;
    return *this;
  }

  void Out() {
    cout << "[" << a[0][0] << " " << a[0][1] << '\n';
    cout << a[1][0] << " " << a[1][1] << "]\n";
  }
};

Matrix operator ^(Matrix A, LL b) {
  Matrix res;
  res._set_I();
  while(b > 0) {
    if(b & 1) res = res * A;
    A = A * A;
```

```cpp
    b >>= 1;
  }
  return res;
}

int calc(int s0, int Min, int Max, LL t) {
  Matrix res;
  res.init(3, -1, 0, 1);
  res = res ^ t;
  return (1ll * res.a[0][0] * s0 % mod + 1ll * res.a[0][1] * fadd(Min, Max) % mod)
% mod;
}
/*
[st   [3 -1   [s_{t-1}
num] =   0 1] *  num]
*/

Matrix fib(LL t, int f1, int f0) {
  Matrix res;
  res.init(1, 1, 1, 0);
  res = res ^ t;
  Matrix fib_0;
  fib_0.init(f1, 0, f0, 0);
  return res * fib_0;
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  cin >> t1 >> t2;
  for(int i = 1; i <= n; ++i) cin >> a[i];
  int sum_0 = 0;
  for(int i = 1; i <= n; ++i) {
    a[i] = a[i] % mod;
    sum_0 = fadd(sum_0, a[i]);
  }
  int sum_x = calc(sum_0, a[1], a[n], t1);
  Matrix fib_x = fib(t1, 1, 0);
  int sum_y = calc(sum_x, fadd(a[1], 0), (1ll * fib_x.a[0][0] * a[n] + 1ll *
fib_x.a[1][0] * a[n - 1]) % mod, t2);
  cout << fadd(sum_y, mod) << "\n";
  return 0;
}
```

T4 选拔赛：

20分做法：

模拟

40分做法：

能够满足条件的特性值具有什么性质?

- 是区间内所有数的约数，也即区间的公约数

如果满足条件的话，区间公约数还要等于其中的某个数，那么这一定是区间的最大公约数。

由于约数一定小于等于区间原来的数，满足条件的值一定是区间的最小值

也即问题转换为 区间公约数=区间最小值

$$gcd(a_l, \cdots, a_r) \le min(a_l, \cdots, a_r) \le a_i$$

100做法：

gcd和min使用线段树 / st表 / 莫队进行维护都可以

统计个数可以使用vector或者主席树

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 10;

int n, q;
int s[maxn];
int f[maxn][25];
int g[maxn][25];

int rt[maxn];
int b[maxn];
int ln;

int Gcd(int x, int y){ return !y ? x : Gcd(y, x % y); }

struct T_Tree{
    int sm[maxn * 40];
    int lc[maxn * 40];
    int rc[maxn * 40];
    int cnt;

    T_Tree() {
    cnt=0;
  }

    void build(int &id, int l, int r){
        id = ++cnt;
        sm[id] = lc[id] = rc[id] = 0;
```

```cpp
        if(l == r) return;
        int mid = (l + r) >> 1;
        build(lc[id], l, mid), build(rc[id], mid+1, r);
    }

    void ins(int &id, int o, int l, int r, int vl){
        id = ++cnt;
        lc[id] = lc[o], rc[id] = rc[o], sm[id] = sm[o] + 1;
        if(l == r) return;
        int mid = (l + r) >> 1;
        if(vl <= mid) ins(lc[id], lc[o], l, mid, vl);
        else ins(rc[id], rc[o], mid+1, r, vl);
    }

    int query(int id, int o, int l, int r, int vl){
        if(l == r) return abs(sm[o] - sm[id]);
        int mid = (l+r) >> 1;
        if(vl <= mid) return query(lc[id], lc[o], l, mid, vl);
        else return query(rc[id], rc[o], mid+1, r, vl);
    }
}tr;

int query_gcd(int l, int r){
    int lg = log2(r - l + 1);
    return Gcd(f[l][lg], f[r - (1 << lg) + 1][lg]);
}

int query_min(int l, int r){
    int lg = log2(r - l + 1);
    return min(g[l][lg], g[r - (1 << lg) + 1][lg]);
}

int main(){
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
    for(int i = 1; i <= n; ++i)
    cin >> s[i], b[++ln] = f[i][0] = g[i][0] = s[i];
    sort(b + 1, b + ln + 1);
  ln = unique(b + 1,b + ln + 1) - b - 1;
    for(int i = 1; i <= 25; ++i)
        for(int j = 1; j + (1 << i) - 1 <= n; ++j)
            f[j][i] = Gcd(f[j][i - 1], f[j + (1 << (i - 1))][i - 1]);
    for(int i = 1; i <= 25; ++i)
        for(int j = 1; j +(1 << i) - 1 <= n; ++j)
            g[j][i] = min(g[j][i - 1], g[j + (1 << (i - 1))][i - 1]);
    tr.build(rt[0], 1, ln);
    for(int i = 1; i <= n; ++i){
        int ap = lower_bound(b + 1, b + ln + 1, s[i]) - b;
```

```
        tr.ins(rt[i], rt[i - 1], 1, ln, ap);
    }
  cin >> q;
    for(int i = 1; i <= q; ++i){
        int L, R;
    cin >> L >> R;
        int gd=query_gcd(L, R);
        int mn=query_min(L, R);
        if(gd != mn) {
      cout << R - L + 1 << '\n';
      continue;
    }
        gd = lower_bound(b + 1, b + ln + 1, gd) - b;
    cout << R - L + 1 - tr.query(rt[L-1], rt[R], 1, ln, gd) << '\n';
    }
    return 0;
}
```