HNCPC Round 6 Tutorial

HolyK

April 2022

1 Problem A. Yonagi and Prime Numbers

Idea: ggcc

Data & Std & Solution & Tutorial: HolyK

设 $dp(x)$ 表示 x 的期望步数。

$$dp(x) = 1 + \frac{1}{50847534}\sum_{p}[p \mid x]dp(x/p) + [p \nmid x]dp(x)$$

状态数为 x 的约数个数，直接记忆化搜索即可。

```cpp
// Author:  HolyK
// Created: Fri Apr 22 17:15:24 2022
#include <bits/stdc++.h>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;

constexpr int P(1e9 + 7);
inline void inc(int &x, int y) {
  x += y;
  if (x >= P) x -= P;
}
inline void dec(int &x, int y) {
  x -= y;
  if (x < 0) x += P;
}
inline int mod(LL x) { return x % P; }
int fpow(int x, int k = P - 2) {
  int r = 1;
  for (; k; k >>= 1, x = 1LL * x * x % P) {
    if (k & 1) r = 1LL * r * x % P;
  }
  return r;
}

constexpr int C(50847534);
```

```cpp
std::map<int, int> f;
std::vector<int> a;
int dfs(int x) {
  if (!x) return 0;
  if (f.count(x)) return f[x];
  int s = C, c = 0;
  for (int i : a) {
    if (x % i) continue;
    inc(s, dfs(x / i));
    c++;
  }
  return f[x] = 1LL * s * fpow(c) % P;
}
void solve() {
  int n, x;
  std::cin >> n;
  x = n;
  for (int i = 2; i * i <= x; i++) {
    if (x % i) continue;
    while (x % i == 0) x /= i;
    a.push_back(i);
  }
  if (x > 1) a.push_back(x);
  std::cout << dfs(n) << "\n";
}

int main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;

  // std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

2 Problem B. Yonagi and Rational Numbers

Idea & Data & Std & Solution: HolyK

Tutorial: ggcc

考虑如何计算 $f(o,l,r,x)$，我们不妨维护分子和分母，且始终满足分子分母互质。

不妨设当前对于当前的 o,l,r，有 $x=\frac{p}{q}(\gcd(p,q)=1)$。

则从 $f(o,l,r,x)$ 到 $f(o,l+1,r,\frac{1}{x+a_l})$ 可以看作：

$$\frac{1}{\frac{p}{q}+a_l}=\frac{q}{p+q\cdot a_l}$$

对于 $f(o,l,r,x)$ 到 $f(o,l+1,r,\frac{1}{x+b_l})$ 同理。

对于新得到的数，我们计算分子分母操作后的 gcd，有：

$$\gcd(q,p+q\cdot a_l)=gcd(q,p+q\cdot a_l-q\cdot a_l)=\gcd(p,q)=1$$

因此我们只需要令最开始的分子分母互质，则之后每一次操作后分子和分母都是互质的。

接着不难发现，每一次操作实际上是 p 和 q 进行一个线性变换，因此不妨用矩阵来维护，则从 $f(o,l,r,x)$ 到 $f(o,l+1,r,\frac{1}{x+a_l})$ 可以看作：

$$\begin{bmatrix}p & q\end{bmatrix}\cdot\begin{bmatrix}0 & 1\\ 1 & a_l\end{bmatrix}=\begin{bmatrix}q & p+q\cdot a_l\end{bmatrix}$$

对于 $f(o,l,r,x)$ 到 $f(o,l+1,r,\frac{1}{x+b_l})$ 同理。

使用矩阵维护之后，因为矩阵满足组合律，因此可以考虑使用线段树维护区间矩阵的乘积。

接着我们再来看题目中的操作：

操作一就是相当于一个单点矩阵修改操作，直接使用线段树单点修改。

操作三可以考虑对于线段树每个节点同时维护 $o=0$ 以及 $o=1$ 的矩阵的乘积。

操作二相当于将线段树一段区间的 $o=0$ 和 $o=1$ 的矩阵乘积交换一下，直接打标记做区间修改即可。

时间复杂度 $O(T^3\cdot n\log n)$，其中 $T=2$ 为矩阵大小。

```cpp
// Author:  HolyK
// Created: Mon Aug 23 14:38:06 2021
#include <bits/stdc++.h>
template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;
constexpr int P(998244353);
inline void inc(int &x, int y) {
```

```
  x += y;
  if (x >= P) x -= P;
}
inline void dec(int &x, int y) {
  x -= y;
  if (x < 0) x += P;
}
inline int mod(LL x) {
  return x % P;
}
int fpow(int x, int k = P - 2) {
  int r = 1;
  for (; k; k >>= 1, x = 1LL * x * x % P) {
    if (k & 1) r = 1LL * r * x % P;
  }
  return r;
}
struct Z {
  int x;
  Z(int v = 0) : x(v < 0 ? v + P : v >= P ? v - P : v) {}
  int val() const {
    return x;
  }
  Z inv() const {
    assert(x);
    return Z(fpow(x));
  }
  Z pow(int k) const {
    return Z(fpow(x, k));
  }
  Z operator-() const {
    return Z(P - x);
  }
  Z &operator+=(const Z &r) {
    inc(x, r.x);
    return *this;
  }
  Z &operator-=(const Z &r) {
    dec(x, r.x);
    return *this;
  }
  Z &operator*=(const Z &r) {
    x = 1LL * x * r.x % P;
    return *this;
```

```
  }
  Z &operator/=(const Z &r) {
    x = 1LL * x * fpow(r.x) % P;
    return *this;
  }
  inline friend Z operator+(const Z &a, const Z &b) {
    return Z(a) += b;
  }
  inline friend Z operator-(const Z &a, const Z &b) {
    return Z(a) -= b;
  }
  inline friend Z operator*(const Z &a, const Z &b) {
    return Z(a) *= b;
  }
  inline friend Z operator/(const Z &a, const Z &b) {
    return Z(a) /= b;
  }
};
using Matrix = std::array<Z, 4>;
Matrix operator*(Matrix a, Matrix b) {
  return {
    a[0] * b[0] + a[1] * b[2],
    a[0] * b[1] + a[1] * b[3],
    a[2] * b[0] + a[3] * b[2],
    a[2] * b[1] + a[3] * b[3]
  };
}
constexpr int N(5e5 + 5);
Matrix f[N << 2][2];
int tag[N << 2];
#define ls o << 1
#define rs o << 1 | 1
void pushup(int o) {
  f[o][tag[o]] = f[ls][0] * f[rs][0];
  f[o][tag[o] ^ 1] = f[ls][1] * f[rs][1];
}
void update(int o, int l, int r, int x, int y) {
  if (x <= l && r <= y) {
    tag[o] ^= 1;
    std::swap(f[o][0], f[o][1]);
    return;
  }
  int m = l + r >> 1;
  if (x <= m) update(ls, l, m, x, y);
```

```
  if (y > m) update(rs, m + 1, r, x, y);
  pushup(o);
}
Matrix ask(int o, int l, int r, int x, int y, int z) {
  if (x <= l && r <= y) return f[o][z];
  int m = l + r >> 1;
  z ^= tag[o];
  if (y <= m) return ask(ls, l, m, x, y, z);
  if (x > m) return ask(rs, m + 1, r, x, y, z);
  return ask(ls, l, m, x, y, z) * ask(rs, m + 1, r, x, y, z);
}
int a[2][N];
void build(int o, int l, int r) {
  tag[o] = 0;
  if (l == r) {
    f[o][0] = {0, 1, 1, a[l & 1][l]};
    f[o][1] = {0, 1, 1, a[l & 1 ^ 1][l]};
    return;
  }
  int m = l + r >> 1;
  build(ls, l, m), build(rs, m + 1, r);
  pushup(o);
}
int main() {
  // freopen("t.in", "r", stdin);
  // freopen("t.out", "w", stdout);
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int n;
  std::cin >> n;
  for (int i = 1; i <= n; i++) std::cin >> a[0][i];
  for (int i = 1; i <= n; i++) std::cin >> a[1][i];
  build(1, 1, n);
  int q;
  std::cin >> q;
  while (q--) {

    int opt, o, l, r, p, q, z;
    std::cin >> opt;
    if (opt == 1) {
      std::cin >> z >> p >> q;
      o = 1, l = 1, r = n;
      while (l < r) {
        z ^= tag[o];
```

```
        int m = l + r >> 1;
        if (p <= m) {
          r = m;
          o = ls;
        } else {
          l = m + 1;
          o = rs;
        }
      }
      f[o][l & 1 ^ z][3] += q;
      while (o >>= 1) pushup(o);
    } else if (opt == 2) {
      std::cin >> l >> r;
      update(1, 1, n, l, r);
    } else {
      assert(opt == 3);
      std::cin >> o >> l >> r >> p >> q;
      assert(p && q);
      int g = std::gcd(p, q);
      p /= g, q /= g;
      auto v = ask(1, 1, n, l, r, o);
      std::cout << (v[0] * p + v[2] * q).val() << " " << (v[1] * p + v[3] * q).val()
      ↪  << "\n";
    }
  }
  std::cerr << (double)clock() / CLOCKS_PER_SEC << "\n";
  return 0;
}
```

3 Problem C. Yonagi and Game

Idea & Data & Std & Solution & Tutorial: HolyK

首先将 a_i, b_i 分别从小到大排序。

设 $dp(i,j)$ 表示前 i 对 a,b 钦定了 j 对匹配（$a > b$）的方案数。

转移很简单：

$$dp(i,j) = dp(i-1,j) + dp(i-1,j-1) \cdot (s_i - j + 1)$$

其中 s_i 表示小于 a_i 的 b 的个数。

设 $f(n,k)$ 表示答案，有

$$(n-k)!dp(n,k) = \sum_{i \geq k} \binom{i}{k} f(n,i)$$

二项式反演

$$f(n,k) = \sum_{i \geq k} \binom{i}{k} (-1)^{i-k} (n-i)! dp(n,i)$$

复杂度 $\mathcal{O}(n^2)$.

```cpp
// Author:  HolyK
// Created: Mon Apr 11 18:39:06 2022
#include <bits/stdc++.h>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;


constexpr int P(1e9 + 7);
inline void inc(int &x, int y) {
  x += y;
  if (x >= P) x -= P;
}
inline void dec(int &x, int y) {
  x -= y;
  if (x < 0) x += P;
}
inline int mod(LL x) { return x % P; }
int fpow(int x, int k = P - 2) {
  int r = 1;
```

```
  for (; k; k >>= 1, x = 1LL * x * x % P) {
    if (k & 1) r = 1LL * r * x % P;
  }
  return r;
}

constexpr int N(100005);
int fac[N], ifac[N];
int bin(int n, int m) {
  if (m < 0 || m > n) return 0;
  return 1LL * fac[n] * ifac[m] % P * ifac[n - m] % P;
}
void init(int n) {
  fac[0] = ifac[0] = 1;
  for (int i = 1; i <= n; i++) fac[i] = 1LL * fac[i - 1] * i % P;
  ifac[n] = fpow(fac[n]);
  for (int i = n - 1; i > 0; i--) ifac[i] = 1LL * ifac[i + 1] * (i + 1) % P;
}
void solve() {
  int n;
  std::cin >> n;
  std::vector<int> a(n), b(n);
  for (int i = 0; i < n; i++) {
    std::cin >> a[i];
  }
  for (int i = 0; i < n; i++) {
    std::cin >> b[i];
  }
  std::sort(a.begin(), a.end());
  std::sort(b.begin(), b.end());
  std::vector<int> dp = {1};
  for (int i = 0, j = 0; i < n; i++) {
    while (j < n && b[j] < a[i]) j++;
    std::vector<int> g = dp;
    g.resize(j + 1);
    for (int k = 0; k < dp.size() && k < j; k++) {
      g[k + 1] = (g[k + 1] + 1LL * dp[k] * (j - k)) % P;
    }
    dp.swap(g);
  }
  init(n * 2);
  for (int i = 0; i < dp.size(); i++) {
    dp[i] = 1LL * dp[i] * fac[n - i] % P;
  }
```

```
  for (int k = 0; k <= n; k++) {
    int ans = 0;
    for (int i = k; i < dp.size(); i++) {
      int x = 1LL * dp[i] * bin(i, k) % P;
      if (i - k & 1) {
        dec(ans, x);
      } else {
        inc(ans, x);
      }
    }
    std::cout << ans << " \n"[k == n];
  }

}

int main() {
  // freopen("t.in", "r", stdin);

  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;

  // std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

4 Problem D. Yonagi and After Story

Idea & Data & Std & Solution & Tutorial: HolyK

设原边集为 E，涉及二选一的边集为 T。
将 T 的边权全部改为 $-\infty$，然后求最小生成树得到边集 M。
可以发现，$M-T$ 里面的边一定会在最终答案里。
所以我们将 $M-T$ 中的边先连起来，将边权计入答案。
然后将每个联通块缩点，因为相当于原边集只去掉了 T 中的边，所以联通块个数是 $\mathcal{O}(q)$ 的。
用原边集 E 在缩点后的图上再跑一次最小生成树，得到边集 E'。
只有边集 E' 的边可能加入最终的答案中，而 $|E'|=\mathcal{O}(q)$。
考虑对于每一个要求，钦定选中其中一条边，这样有 2^q 种状态。
对于每个状态，用并查集维护 $\mathcal{O}(q)$ 个联通块的联通性。
先将钦定的边先选上，之后再加入 E' 中的边，就可以得到这个状态的答案。
对于所有状态的答案取最小值就可以得到每个要求至少选了一条边的答案。
复杂度 $\mathcal{O}(m\log m+2^q\cdot q\cdot\alpha(q))$.

```cpp
// Author:  HolyK
// Created: Fri Apr 22 15:13:43 2022
#include <bits/stdc++.h>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;

struct Dsu {
  std::vector<int> f;
  void clear() {
    std::iota(f.begin(), f.end(), 0);
  }
  Dsu(int n) : f(n) { clear(); }
  int find(int x) {
    while (x != f[x]) {
      x = f[x] = f[f[x]];
    }
    return x;
  }
  bool same(int x, int y) {
```

```
    return find(x) == find(y);
  }
  bool merge(int x, int y) {
    return !same(x, y) ? f[find(x)] = find(y), true : false;
  }
};

void solve() {
  int n, m;
  std::cin >> n >> m;
  std::vector<std::array<int, 3>> e(m);
  for (auto &[z, x, y] : e) std::cin >> x >> y >> z, x--, y--;
  int q;
  std::cin >> q;
  std::vector<std::array<int, 2>> qq(q);
  Dsu d(n), d1(n);
  for (auto &[x, y] : qq) {
    std::cin >> x >> y;
    x--, y--;
    d.merge(e[x][1], e[x][2]);
    d.merge(e[y][1], e[y][2]);
  }
  LL psum = 0;
  auto ee = e;
  std::sort(ee.begin(), ee.end());
  for (auto &[z, x, y] : ee) {
    if (d.merge(x, y)) {
      d1.merge(x, y);
      psum += z;
    }
  }
  std::vector<int> id(n, -1);
  int k = 0;
  for (int i = 0; i < n; i++) {
    if (d1.find(i) == i) id[i] = k++;
  }
  Dsu d2(k);
  std::vector<std::array<int, 3>> e0;
  for (auto &[z, x, y] : e) x = id[d1.find(x)], y = id[d1.find(y)];
  for (auto &[z, x, y] : ee) {
    x = id[d1.find(x)];
    y = id[d1.find(y)];
    if (x != y && d2.merge(x, y)) {
      e0.push_back({z, x, y});
```

```
    }
  }
  LL ans = 1e18;
  for (int i = 0; i < 1 << q; i++) {
    Dsu d(k);
    LL sum = 0;
    for (int j = 0; j < q; j++) {
      auto &o = e[qq[j][i >> j & 1]];
      sum += o[0];
      d.merge(o[1], o[2]);
    }
    for (auto &[z, x, y] : e0) {
      if (d.merge(x, y)) sum += z;
    }
    smin(ans, sum);
  }
  std::cout << psum + ans << "\n";
}

int main() {
  // freopen("t.in", "r", stdin);

  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;

  // std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

5 Problem E. Yonagi and YOASOBI

Idea: zhrrrr

Data & Std & Solution & Tutorial: HolyK

有很多种方法，一种方法是处理后缀 h 的个数，然后答案为

$$\sum[s_i = a](h-1)h/2$$

当然也可以直接 DP。

复杂度 $\mathcal{O}(n)$.

```cpp
// Author:  HolyK
// Created: Mon Jan  3 21:09:44 2022
#include <bits/stdc++.h>
template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;

void solve() {
  std::string s;
  std::cin >> s;
  LL c[3] = {};
  for (char x : s) {
    if (x == 'a') {
      c[0]++;
    } else if (x == 'h') {
      c[2] += c[1];
      c[1] += c[0];
    }
  }
  std::cout << c[2] << "\n";
}

int main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int t = 1;
  // std::cin >> t;
```

```
  while (t--) {
    solve();
  }
  return 0;
}
```

6 Problem F. Yonagi and Robot

Idea & Data & Std & Solution & Tutorial: ggcc

对于一个平面图，有欧拉公式：

$$R + V - E = 2$$

其中 R 为面数，V 为点数，E 为边数。

面数 R 即本题所求。

则我们只需要统计经过的点数和边数即可，使用 std::map 记录每个点或者每条边是否出现，统计一下点数和边数即可，复杂度 $O(T \log T)$。

```
#include <bits/stdc++.h>

#define mp make_pair
using namespace std;
using LL = long long;
using PII = std::pair<int, int>;
const int dx[4] = {0, 1, 0, -1};
const int dy[4] = {1, 0, -1, 0};
const int N = 1e6 + 10;

map<PII, bool> mp1;
map<pair<PII, PII>, bool> mp2;

char ss[N];

int main() {
  std::ios::sync_with_stdio(0);
  std::cin.tie(0);
  int T; cin >> T;
  cin >> (ss + 1);
  int V = 1, E = 0, x = 0, y = 0;
  mp1[mp(0, 0)] = 1;
  for(int i = 1; i <= T; i++) {
    int k;
    if(ss[i] == 'U') k = 1;
    else if(ss[i] == 'D') k = 3;
    else if(ss[i] == 'L') k = 2;
    else k = 0;
    int nx = x + dx[k], ny = y + dy[k];
    if(!mp1.count(mp(nx, ny))) V++, mp1[make_pair(nx, ny)] = 1;
    if(!mp2.count(mp(mp(x, y), mp(nx, ny)))) {
      E++;
      mp2[mp(mp(x, y), mp(nx, ny))] = mp2[mp(mp(nx, ny), mp(x, y))] = 1;
    }
```

```
    x = nx, y = ny;
  }
  cout << 2 + E - V;
  return 0;
}
```

7 Problem G. Yonagi and Tree

Idea & Data & Std & Solution: HolyK

Tutorial: ggcc

不难发现如果给一棵树下面接上任意个白点，对答案都没有影响。

因此不妨先将树建出来，一开始所有点都设为白色，那么操作就变成了每一次修改一个点的颜色。

不妨考虑 $O(n)$ 暴力 dp 计算单次操作，则我们设 $f_{x,0/1}$ 为以 x 为根的子树没有黑点/已经有一个黑点的方案数。

则枚举子儿子 y 可以得到转移：

$$f'_{x,0} = f_{x,0} \cdot (f_{y,0} + f_{y,1})$$

$$f'_{x,1} = f_{x,0} \cdot f_{y,1} + f_{x,1} \cdot (f_{y,0} + f_{y,1})$$

进一步地，考虑动态 dp，则我们需要注意处理两个部分。

一就是轻儿子到父亲的转移，二就是重链上如何转移。

首先是轻儿子的转移，观察儿子 y 到父亲 x 的转移，可以发现是一个矩阵：

$$\begin{bmatrix} f_{x,0} & f_{x,1} \end{bmatrix} \cdot \begin{bmatrix} f_{y,0} + f_{y,1} & f_{y,1} \\ 0 & f_{y,0} + f_{y,1} \end{bmatrix} = \begin{bmatrix} f'_{x,0} & f'_{x,1} \end{bmatrix}$$

由于 $f_{y,0} > 0$ 恒大于 0，因此该矩阵显然可逆，此外该矩阵转移的时候显然满足交换律，因为交换儿子的顺序转移得到的结果是一样的，因此必定是满足交换律的。

因此当我们修改了一个轻儿子的信息时，我们直接在父亲处乘上原本的逆矩阵消除之前的影响，再乘上修改后的值即可。

二就是重链上的转移，由于这个转移他实际上非常奇怪，因此在链上进行上述转移仿佛显得有些不可行。

因此我们不妨考虑直接对链进行信息维护，考虑用线段树维护链上的信息，则我们需要维护：

1. 整条链是一个连通块，且没有黑色/有一个黑色的方案数。

2. 整条链不是一个连通块，且链底的连通块没有黑色/有一个黑色，链顶的联通块没有黑色/有一个黑色的方案数。

则直接线段树转移即可。

上述做法是时间复杂度是 $O(n \log^2 n)$，由于轻儿子信息可减，因此可以改成全局平衡二叉树在 $O(n \log n)$ 的时间内完成。

```cpp
// Author:  HolyK
// Created: Fri Apr 15 10:05:28 2022
#include <bits/stdc++.h>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}
```

```cpp
using LL = long long;
using PII = std::pair<int, int>;
constexpr int P(1e9 + 7);
inline void inc(int &x, int y) {
  x += y;
  if (x >= P) x -= P;
}
inline void dec(int &x, int y) {
  x -= y;
  if (x < 0) x += P;
}
inline int sum(int x, int y) {
  return x + y >= P ? x + y - P : x + y;
}
inline int mod(LL x) { return x % P; }
int fpow(int x, int k = P - 2) {
  int r = 1;
  for (; k; k >>= 1, x = 1LL * x * x % P) {
    if (k & 1) r = 1LL * r * x % P;
  }
  return r;
}

constexpr int N(5e5 + 5);
int n, m, col[N], fa[N];
std::vector<int> g[N];
PII q[N];

int siz[N], son[N], dep[N], top[N];
void dfs1(int x) {
  siz[x] = 1;
  for (int y : g[x]) {
    dfs1(y);
    siz[x] += siz[y];
    if (siz[y] > siz[son[x]]) son[x] = y;
  }
}

using RInfo = std::array<int, 2>;
struct Info {
  int a, b;
  Info(int c = 0) : a(1), b(c) {}
  Info(int a, int b) : a(a), b(b) {}
  Info(RInfo r) : a(sum(r[0], r[1])), b(r[1]) {}
```

```
  Info operator+(const Info &r) const {
    return Info(1LL * a * r.a % P, (1LL * a * r.b + 1LL * b * r.a) % P);
  }
  Info operator-(const Info &r) const {
    int i = fpow(r.a);
    int aa = 1LL * i * a % P;
    int bb = (b - 1LL * aa * r.b % P + P) * i % P;
    return Info(aa, bb);
  }
};


RInfo cal(Info i, RInfo r) {
  return {
    mod(1LL * i.a * r[0]),
    mod(1LL * i.b * r[0] + 1LL * i.a * r[1])
  };
}
using CInfo = std::array<RInfo, 2>;
RInfo operator+(RInfo a, RInfo b) {
  return {sum(a[0], b[0]), sum(a[1], b[1])};
}
CInfo operator+(CInfo a, CInfo b) {
  return {a[0] + b[0], a[1] + b[1]};
}

RInfo rakeR(RInfo a, RInfo b) {
  return {mod(1LL * a[0] * b[0]), mod(1LL * a[0] * b[1] + 1LL * a[1] * b[0])};
}

CInfo rakeC(RInfo a, RInfo b) {
  return {RInfo{mod(1LL * b[0] * a[0]), mod(1LL * b[0] * a[1])},
          RInfo{mod(1LL * b[1] * a[0]), mod(1LL * b[1] * a[1])}};
}
CInfo rake(RInfo a, CInfo b) {
  return {RInfo{mod(1LL * (b[0][0] + b[0][1]) * a[0]),
                mod(1LL * b[0][0] * a[1] + 1LL * b[0][1] * (a[0] + a[1]))},
          RInfo{mod(1LL * (b[1][0] + b[1][1]) * a[0]),
                mod(1LL * b[1][0] * a[1] + 1LL * b[1][1] * (a[0] + a[1]))}};
}
CInfo compress(CInfo a, CInfo b) {
  return {
      RInfo{
        mod(1LL * b[0][0] * a[1][0] + 1LL * b[0][1] * (a[0][0] + a[1][0])),
```

```
        mod(1LL * b[0][0] * a[1][1] + 1LL * b[0][1] * (a[0][1] + a[1][1]))},
      RInfo{
        mod(1LL * b[1][0] * a[1][0] + 1LL * b[1][1] * (a[0][0] + a[1][0])),
        mod(1LL * b[1][0] * a[1][1] + 1LL * b[1][1] * (a[0][1] + a[1][1]))}};
}

constexpr CInfo CNull = {};

struct Node {
  int ls, rs, fa, col;
  Info v;
  RInfo r;
  CInfo c;
  Node() : ls(0), rs(0), fa(0), col(0), v(), r{1, 0}, c{} {}
} t[N];

void pushup(int o) {
  int col = t[o].col;
  RInfo rl = t[t[o].ls].r;
  RInfo rm = cal(t[o].v, RInfo{!col, col});
  RInfo rr = t[t[o].rs].r;
  t[o].r[0] = 1LL * rl[0] * rm[0] % P * rr[0] % P;
  t[o].r[1] = (1LL * rl[1] * rm[0] % P * rr[0] + 1LL * rl[0] * rm[1] % P * rr[0] + 1LL
  ↪  * rl[0] * rm[0] % P * rr[1]) % P;

  CInfo cr = t[o].rs ? rake(rm, t[t[o].rs].c) : CNull;
  if (t[o].rs) {
    cr = cr + rakeC(rm, rr), rm = rakeR(rm, rr);
  }
  auto &c = t[o].c;
  if (t[o].ls) {
    auto &cl = t[t[o].ls].c;
    c = compress(cl, cr) + rake(rl, cr) + rakeC(rl, rm);
    c[0][0] = (c[0][0] + 1LL * rm[0] * (cl[0][0] + cl[1][0])) % P;
    c[0][1] = (c[0][1] + 1LL * rm[0] * (cl[0][1] + cl[1][1])) % P;
    c[1][0] = (c[1][0] + 1LL * rm[0] * cl[1][0] + 1LL * rm[1] * (cl[0][0] + cl[1][0]))
    ↪  % P;
    c[1][1] = (c[1][1] + 1LL * rm[0] * cl[1][1] + 1LL * rm[1] * (cl[0][1] + cl[1][1]))
    ↪  % P;
  } else {
    c = cr;
  }
}
```

```cpp
int a[N], b[N], root;
int build(int p, int l, int r) {
  if (l > r) return 0;
  int m = std::lower_bound(b + l, b + r, (b[l - 1] + b[r]) / 2) - b;
  int o = a[m];
  t[o].fa = p;
  t[o].ls = build(o, l, m - 1);
  t[o].rs = build(o, m + 1, r);
  pushup(o);
  return o;
}
void dfs2(int x) {
  if (!son[x]) {
    int p = top[x], m = 0;
    while (p != x) {
      a[++m] = p;
      b[m] = siz[p] - siz[son[p]];
      p = son[p];
    }
    a[++m] = x;
    b[m] = 0;
    for (int i = 1; i <= m; i++) b[i] += b[i - 1];
    p = fa[top[x]];
    x = build(p, 1, m);
    if (p) {
      t[p].v = t[p].v + (t[x].r + t[x].c[1]);
    } else {
      root = x;
    }
    return;
  }

  for (int y : g[x]) {
    if (y != son[x]) {
      top[y] = y;
      dfs2(y);
    }
  }

  top[son[x]] = top[x];
  dfs2(son[x]);
}

void solve() {
```

```
  std::cin >> m >> t[1].col;
  n = 1;
  col[1] = t[1].col;
  for (int i = 0; i < m; i++) {
    int opt, x;
    std::cin >> opt >> x;
    if (opt == 1) {
      std::cin >> col[++n];
      g[x].push_back(n);
      fa[n] = x;
      x = n;
    } else {
      col[x] ^= 1;
    }
    q[i] = {x, col[x]};
  }

  dfs1(1), top[1] = 1, dfs2(1);
  for (int i = 0; i < m; i++, std::cout << sum(t[root].r[1], t[root].c[1][1]) << "\n")
  ↪  {
    auto [x, c] = q[i];
    if (t[x].col == c) continue;
    t[x].col = c;
    while (x) {
      int p = t[x].fa;
      if (p && t[p].ls != x && t[p].rs != x) {
        t[p].v = t[p].v - (t[x].r + t[x].c[1]);
        pushup(x);
        t[p].v = t[p].v + (t[x].r + t[x].c[1]);
      } else {
        pushup(x);
      }
      x = p;
    }
  }
}

int main() {
  // freopen("t.in", "r", stdin);

  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;
```

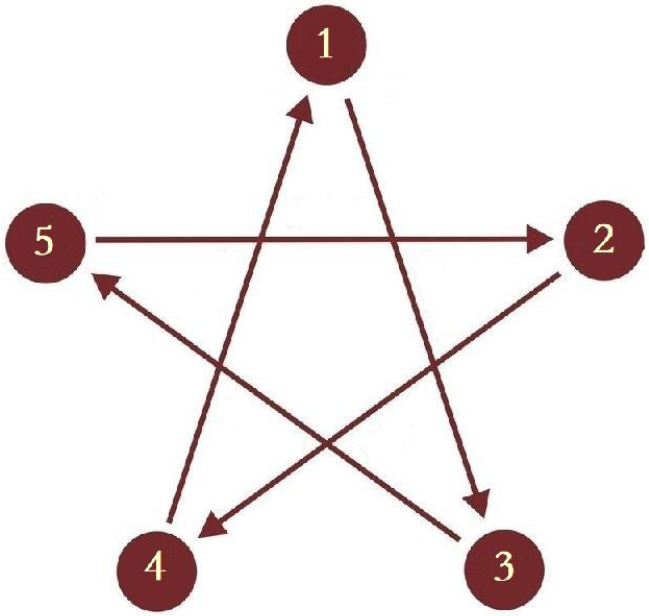
```
  // std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

8 Problem H. Good God Can Count

Idea & Data & Std & Solution & Tutorial: ggcc

观察一轮操作的本质是是什么。

我们以 i 和 $(i+1) \bmod 5+1$ 为一个 pair，如下图：



箭头指向的为一个 pair。

我们想要令 1 号点加一，只需要令 3 5 和 2 4 这两个 pair 各自减一，再令所有元素加一，即可。

想要令 1 号点减一，并且使 2 号和 5 号点加一，只需令 1 3 和 4 1 这两个 pair 各自减一，再令所有元素加一，即可。

其他情况通过手玩不难发现都可以通过这样的方式得到。

则每次操作就是先令所有元素各自加一，再选择两个不同的 pair 各自减一，$\binom{5}{2}$ 正好对应 10 种操作。

不难发现可以通过解方程的方式算出每一个 pair 最后被选择的次数，设第 i 个 pair 出现次数是 c_i。

那么问题变成，有五个元素，每次选择两个不同的元素，最后使得第 i 个元素出现次数是 c_i 的方案数，且满足 $\sum c_i = k \cdot 2$。

考虑令两个不同元素选择有序，即 1 3 和 3 1 是不同的选择，最后答案乘上 $\frac{1}{2^k}$ 即可。

我们先不考虑每次选择两个不同元素的限制，那么方案数就是 $\frac{(2k)!}{\prod c_5!}$。

然后考虑容斥，那么就是考虑枚举第 i 个 pair，在 b_i 次选择中出现了重复，那么方案数乘上容斥系数就是：

$$(-1)^{\sum b_i} \frac{(\sum (c_i - 2 \cdot b_i))!}{\prod (c_i - 2 \cdot b_i)!} \frac{k!}{(k - \sum b_i)! \prod b_i!}$$

整理一下可以得：

$$\frac{k! \cdot (2 \cdot k - 2 \cdot \sum b_i)!}{(k - \sum b_i)!} \prod \left(\frac{(-1)^{b_i}}{(c_i - 2 \cdot b_i)! \cdot b_i!} \right)$$

不难发现对于每一个 i 是一个多项式，我们直接卷积卷一下即可。

复杂度 $O(t \cdot n \log n)$，其中 $t = 5$。

```
#include <bits/stdc++.h>
```

```cpp
#define I inline
#define fi first
#define se second
#define LL long long
#define mp make_pair
#define pii pair<int,int>
#define fo(i, a, b) for(int i = int(a); i <= int(b); i++)
#define fd(i, a, b) for(int i = int(a); i >= int(b); i--)
#define bp __builtin_popcount
#define vi vector<int>
#define cr const int&
using namespace std;constexpr int P(998244353), G(3), L(1 << 20);
inline void inc(int &x, int y) {
  x += y;
  if (x >= P) x -= P;
}
inline void dec(int &x, int y) {
  x -= y;
  if (x < 0) x += P;
}
inline int mod(LL x) { return x % P; }
int fpow(int x, int k = P - 2) {
  int r = 1;
  for (; k; k >>= 1, x = 1LL * x * x % P) {
    if (k & 1) r = 1LL * r * x % P;
  }
  return r;
}
int w[L], fac[L], ifac[L], inv[L], _ = [] {
  w[L / 2] = 1;
  for (int i = L / 2 + 1, x = fpow(G, (P - 1) / L); i < L; i++) w[i] = 1LL * w[i - 1]
  ↪  * x % P;
  for (int i = L / 2 - 1; i >= 0; i--) w[i] = w[i << 1];
  fac[0] = 1;
  for (int i = 1; i < L; i++) fac[i] = 1LL * fac[i - 1] * i % P;
  ifac[L - 1] = fpow(fac[L - 1]);
  for (int i = L - 1; i; i--) {
    ifac[i - 1] = 1LL * ifac[i] * i % P;
    inv[i] = 1LL * ifac[i] * fac[i - 1] % P;
  }
  return 0;
}();
void dft(int *a, int n) {
  for (int k = n >> 1; k; k >>= 1) {
```

```
    for (int i = 0; i < n; i += k << 1) {
      for (int j = 0; j < k; j++) {
        int &x = a[i + j], y = a[i + j + k];
        a[i + j + k] = 1LL * (x - y + P) * w[k + j] % P;
        inc(x, y);
      }
    }
  }
}
void idft(int *a, int n) {
  for (int k = 1; k < n; k <<= 1) {
    for (int i = 0; i < n; i += k << 1) {
      for (int j = 0; j < k; j++) {
        int x = a[i + j], y = 1LL * a[i + j + k] * w[k + j] % P;
        a[i + j + k] = x - y < 0 ? x - y + P : x - y;
        inc(a[i + j], y);
      }
    }
  }
  for (int i = 0, inv = P - (P - 1) / n; i < n; i++)
  a[i] = 1LL * a[i] * inv % P;
  std::reverse(a + 1, a + n);
}
inline int norm(int n) {return 1 << std::__lg(n * 2 - 1);}

int a[6], b[6], S[L], A[L];

void solve() {
  fo(i, 0, 4) cin >> a[i];
  int k; cin >> k;
  LL sum = 0;
  fo(i, 0, 4) {
    a[i] = -a[i] + k;
    sum += a[i];
  }
  if(sum & 1) {cout << "0\n"; return ;}
  sum >>= 1;
  LL o = 0;
  fo(i, 0, 4) {
    b[i] = sum - a[i] - a[(i + 1) % 5], o += b[i];
    if(b[i] > k || b[i] < 0) {cout << "0\n"; return ;}
  }
  if(o != k * 2) {cout << "0\n"; return ;}
  int ll = 1; S[0] = 1;
```

```
  fo(i, 0, 4) {
    fo(j, 0, b[i] >> 1) {
      A[j] = (LL)ifac[b[i] - j * 2] * ifac[j] % P;
      if(j & 1) A[j] = -A[j], A[j] += P;
    }
    int len = norm(b[i] / 2 + 1 + ll);
    fo(j, ll, len - 1) S[j] = 0;
    fo(j, b[i] / 2 + 1, len - 1) A[j] = 0;
    dft(S, len), dft(A, len);
    fo(j, 0, len - 1) S[j] = (LL)S[j] * A[j] % P;
    idft(S, len);
    ll += b[i] >> 1;
  }
  int ans = 0;
  fo(i, 0, ll - 1) {
    ans = (ans + (LL)S[i] * fac[i] % P * fac[k] % P * ifac[i] % P * ifac[k - i] % P *
    ↪  fac[k * 2 - i * 2]) % P;
  }
  ans = (LL)ans * fpow(inv[2], k) % P;
  cout << ans << '\n';
}

int main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int T; cin >> T;
  while(T--) solve();
  return 0;
}
```

9 Problem I. Matrix

Idea & Data & Std & Solution & Tutorial: liqing

构造题。

考虑构造一个结构为 $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 的矩阵，其中 0 代表着一个 B 行 A 列的矩阵。

```cpp
// Author:  liqing
// Created: Wed Apr 20 17:02:37 2022
#include<bits/stdc++.h>

using namespace std;
int main()
{
    //freopen("a.in","r",stdin);
    //freopen("a.out","w",stdout);
    cin.tie(nullptr)->sync_with_stdio(false);
    int n,m,a,b;cin>>n>>m>>b>>a;
    for(int i=1;i<=a;i++)
    {
        for(int j=1;j<=b;j++)cout<<1;
        for(int j=b+1;j<=m;j++)cout<<0;
        cout<<endl;
    }
    for(int i=a+1;i<=n;i++)
    {
        for(int j=1;j<=b;j++)cout<<0;
        for(int j=b+1;j<=m;j++)cout<<1;
        cout<<endl;
    }
    return 0;
}
```

10 Problem J. 1 and 2

Idea & Solution & Tutorial: ggcc

Std & Data: HolyK

首先，我们构造的上界显然是 $p-1$。

考虑如何用最小的步数构造出 $p-1$，设 $p-1$ 的最高位为 mx，$p-1$ 的二进制位中有 s 个 1。

显然，当 $k > mx+s$ 时，我们直接令一开始一直乘二即可使 $k = mx+s$，即可构造出 $p-1$。

当 $mx+1 \le k \le mx+s$ 时，我们考虑直接从最高位的 1 开始往低位的 1 取，能取就取即可。

当 $1 \le k < mx+1$ 时，此时取不到最高位的 1，我们直接取 2^{k-1} 即可。

当 $k = 0$ 时，答案为 0。

```cpp
// Author:  HolyK
// Created: Wed Apr 20 18:08:50 2022
#include <bits/stdc++.h>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;

void solve() {
  LL p, k;
  std::cin >> p >> k;
  p--;
  if (!k || !p) {
    std::cout << "0\n";
    return;
  }
  int m = std::min(std::__lg(p), k - 1);
  k -= m + 1;
  LL ans = 1LL << m;
  for (int i = m - 1; i >= 0 && k; i--) {
    if (ans + (1LL << i) <= p) {
      ans += 1LL << i;
      k--;
    }
  }
  std::cout << ans << "\n";
```

```
}

int main() {
  // freopen("t.in", "r", stdin);

  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;

  std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

11 Problem K. Picking Trash

Idea & Data & Std & Solution & Tutorial: Changxv

首先我们建了 n 个垃圾，所花的体力为 $n \times m$。然后我们设自己会回去丢 k 次垃圾，那么所花的体力值为 $k \times m$ 。所以，总体力花费就等于在运送垃圾的路上的花费的体力值加上 $m(n+k)$。

我们会回去丢 k 趟垃圾，考虑其中一趟，这一趟中选择捡垃圾的地点 **依次**为 $x_1, x_2, \ldots, x_l$，于是考虑消耗的体力值：

$$\begin{aligned} & x_1 + 4(x_1 - x_2) + 9(x_2 - x_3) + \cdots + (l+1)^2(x_l - 0) \\ = & (4+1)x_1 + (9-4)x_2 + \cdots + ((l+1)^2 - l^2)x_l \\ = & 5x_1 + 5x_2 + 7x_3 + \cdots + (2l+1)x_l \end{aligned}$$

所以贪心来看，每一趟一定是最开始走到最远点，然后往回依次捡垃圾；综合所有趟（假设 k 趟），一定是最远的 k 个点作为每趟的第一个点，之后的 k 个点作为每趟第二个点……，以此类推。

我们枚举 k 求解，将距离前缀和以后就是一个调和级数的复杂度，总的时间复杂度为 $O(n \log n)$。

这道题还有个小坑就是有些情况中途计算的时候会爆 `long long`，但是我们知道一个朴素的 $k = n$ 的解是满足要求的且答案在 `long long` 范围内。所以在计算过程中注意处理这种情况。

```cpp
#include <bits/stdc++.h>
#define vc vector

using namespace std;
using ll = long long;

int main() {
  cin.tie(0)->sync_with_stdio(0);
  int n, m;
  cin >> n >> m;
  if (n == 0) return cout << "0\n", 0;
  vc<ll> d(n + 1);
  for (int i = 1; i <= n; ++i) {
    cin >> d[i];
    d[i] += d[i - 1];
  }
  ll ans = 1e18;
  for (int k = 1; k <= n; ++k) {
    ll xi = 3, sum = 0;
    for (int i = n; i >= 1; i -= k) {
      sum += (d[i] - d[max(0, i - k)]) * max(xi, 5ll);
      xi += 2;
      if (sum >= ans) break;
    }
    ans = min(ans, sum + (ll)(k + n) * m);
  }
  cout << ans << '\n';
```

```
}
```

12 Problem L. Min, Sum and Max

Idea & Data & Solution & Std & Tutorial: HolyK

如果左端点是固定的，容易发现只需要将右端点排序之后，取两两相邻的算一下答案即可。因为区间变长，最小值变小，要求最大值那肯定是区间越短越好。

拓展一下，对于同一个 $f(l_1, r_1)$，我们只需要求当前 y 的前驱和后继即可。

考虑在笛卡尔树上进行这个过程，把每个数对按照 x 插入到笛卡尔树上，用 `std::set` 维护子树的所有数对。合并两个子树时，$f(l_1, r_1)$ 是确定的，所以启发式合并 `std::set` 的同时查询前驱后继，用 st 表查询区间最小值维护答案即可。

复杂度 $\mathcal{O}(n \log^2 n)$.

```
// Author:  HolyK
// Created: Sat Oct 23 15:36:43 2021
#include <bits/stdc++.h>
template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
inline bool smax(T &x, const U &y) {
  return x < y ? x = y, 1 : 0;
}

using LL = long long;
using PII = std::pair<int, int>;

constexpr int N(2e5 + 5);
int n, m, a[N], st[18][N], lg[N];
int get(int i, int j) {
  return a[i] < a[j] || a[i] == a[j] && i < j ? i : j;
}
int ask(int l, int r) {
  int k = lg[r - l + 1];
  return get(st[k][l], st[k][r - (1 << k) + 1]);
}

std::set<int> s[N];
int ans;

int ins(std::set<int> &s, int x) {
  auto [it, ok] = s.insert(x);
  if (!ok) return a[x];
  int r = -1e9;
  if (it != s.begin()) {
    smax(r, a[ask(*std::prev(it), x)]);
```

```
  }
  if (++it != s.end()) {
    smax(r, a[ask(x, *it)]);
  }
  return r;
}
void merge(int x, int y) {
  if (!x) return;
  if (s[x].size() > s[y].size()) std::swap(s[x], s[y]);
  for (auto u : s[x]) {
    smax(ans, ins(s[y], u) + a[y]);
  }
  s[x].clear();
}
int solve(int l, int r) {
  if (l > r) return 0;
  int o = ask(l, r);
  merge(solve(l, o - 1), o);
  merge(solve(o + 1, r), o);
  return o;
}

int main() {
  // freopen("3.in", "r", stdin);
  // freopen(".out", "w", stdout);
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  std::cin >> n >> m;
  for (int i = 1; i <= n; i++) {
    std::cin >> a[i];
    st[0][i] = i;
  }

  for (int i = 2; i <= n; i++) lg[i] = lg[i >> 1] + 1;
  for (int i = 1; i < 18; i++) {
    for (int j = 1; j + (1 << i) - 1 <= n; j++) {
      st[i][j] = get(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
    }
  }

  for (int i = 1, x, y; i <= m; i++) {
    std::cin >> x >> y;
    smax(ans, ins(s[x], y) + a[x]);
  }
```

```
  solve(1, n);

  std::cout << ans << "\n";
  return 0;
}
```