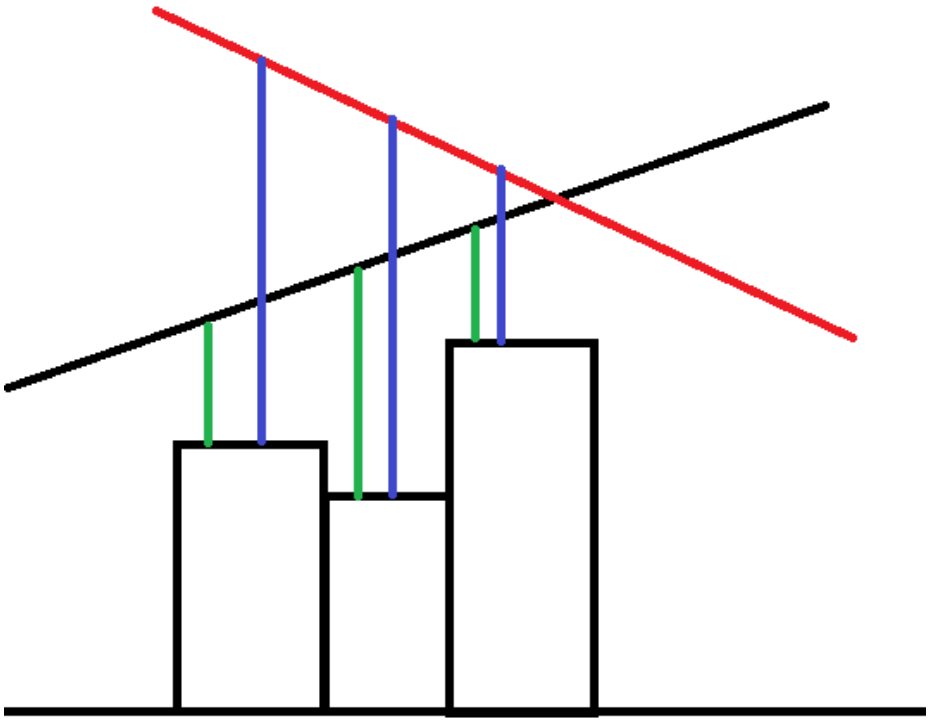
chino with minimum -二维凸包+斜率优化dp

题面

给定长度为n的数组a, 和询问m个长度为n的等差数组b_i, 每个b_i用b_{i0}和k_i来描述.
对于每个询问b_i，记$c_i = \{c_{ij} | c_{ij} = b_{ij} - a_j\}$, 求$min\{c_{ij}\}$
$n, m \in [1, 10^5], a_i \in [0, 10^{12}], b_{i0} \in [0, 10^{12}], k_i \in [-10^7, 10^7]$

分析

题目所求很直白, 但是因为m太大了, 如果我们每次都在线枚举$i \in [1, n]$去找到$min\{c_{ij}\}$显然会tle的.
考虑一下数形结合, b_i相当于给定的直线段, a_i相当于一个给定的图形
那么每个$min\{c_{ij}\}$, 就是直线b_i竖直方向上到达图形的最短距离.
对于一根直线, 我们不难想到, 如果a_i顺着斜率不下降, 那么后面的a_i才可能会更新答案, 否则一定对答案没贡献
具体而言：$k_i > 0$, 若$a_j < a_{j+1}$, 则$c_{i,j+1}$可能更新答案, 否则$c_{i,j+1} > ans$
进一步的, 不难发现, 被其他包着的a_i一定对答案没有贡献, 也就是说a_i的凸包上的点才会有贡献
但是这样枚举a_i还是会tle的, 事实上如果会斜率优化dp的话
那么将询问的直线顺着凸包的斜率排的话, 那么答案在凸包上是单调不下降的
当然直接从上面的结论, a_i要顺着斜率才有可能更新答案反过来考虑的话, 也能想到离线化处理询问, 然后将直线顺着凸包的斜率排序
具体而言, 因为这个凸包的斜率至多是一个坡, 即从$k > 0 \to k < 0$, 所以我们将k_i也从大到小排即可



不难证明, 中间那个肯定对答案没贡献

代码

```cpp
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
#define LL __int128
using namespace std;
```

```cpp
int qpow(int a, int b, int p) {int ret = 1; for(a %= p; b; b >>= 1, a = a * a % p) if(b & 1) ret = ret * a % p; return ret; }
int qpow(int a,int b) {int ret = 1; for(; b; b >>= 1, a *= a) if(b & 1) ret *= a; return ret; }
int gcd(int x,int y) {return y ? gcd(y, x % y) : x; }
pair<int,int> exgcd(int a,int b) { if(!b) return {1, 0}; pair<int,int> ret = exgcd(b, a % b); return {ret.second, ret.first - a / b * ret.second }; }
int lcm(int x,int y){ return x / gcd(x, y) * y; }

const int N = 5 + 1e5;

struct node{
    int x, y;
};
struct line{
    int k, b, i;
};

int mul(node a1, node a2, node b1, node b2) {
    return (a1.x - a2.x) * (b1.y - b2.y) - (a1.y - a2.y) * (b1.x - b2.x);
}
double d(node a, node b) {
    double x1 = a.x, y1 = a.y, x2 = b.x, y2 = b.y;
    return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
};
node a[N];
line q[N];
int ans[N];
signed main() {
    int n, m; cin >> n >> m;
    for (int i = 0; i < n; i++) {
        cin >> a[i].y;
        a[i].x = i;
    }
    for (int i = 0; i < m; i++) {
        cin >> q[i].b >> q[i].k;
        q[i].i = i;
    }
    sort(a + 1, a + n, [&](auto u, auto v) {
        int z = mul(u, a[0], v, a[0]);
        if (z < 0) {
              return 1;
        } else if (!z && u.x < v.x) return 1;
        return 0;
    });
    stack<node> s;
    for (int i = 0; i < n; i++) {
        if (s.size() < 2) {
            s.push(a[i]);
        } else {
            node q1 = s.top(); s.pop();
            node q2 = s.top();
            while (s.size() > 1 && mul(q2, q1, q1, a[i]) > 0) {
                q1 = q2; s.pop();
                q2 = s.top();
            }
            s.push(q1);
            s.push(a[i]);
        }
    }
    vector<node> c;
    while (s.size()) {
        c.push_back(s.top());
        s.pop();
    }
    reverse(c.begin(), c.end());
    sort(q, q + m, [&](line u, line v) {
```

```
        return u.k > v.k;
    });
    for (int i = 0, j = 0, sz = c.size(); i < m; i++) {
        for (; j < sz - 1; j++) {
            if (q[i].k * (c[j + 1].x - c[j].x) - (c[j + 1].y - c[j].y) > 0) {
                ans[q[i].i] = q[i].k * c[j].x + q[i].b - c[j].y;
                //cout << "i:" << q[i].i << ",j:" << j << endl;
                break;
            }
        }
        if (j == sz - 1) {
            ans[q[i].i] = q[i].k * c[j].x + q[i].b - c[j].y;
        }
    }
    for (int i = 0; i < m; i++) {
        cout << ans[i] << endl;
    }
}
```