# Bamboo Team Notes

## Contents



# 1 Arithmetic

## 1.1 Extended Euclide

```
int bezout(int a, int b) {
    // return x such that ax + by == gcd(a, b)
    int xa = 1, xb = 0;
    while (b) {
        int q = a / b;
        int r = a - q * b, xr = xa - q * xb;
        a = b; xa = xb;
        b = r; xb = xr;
    }
    return xa;
}

pair<int, int> solve(int a, int b, int c) {
    // solve ax + by == c
    int d = __gcd(a, b);
    int x = bezout(a, b);
    int y = (d - a * x) / b;
    c /= d;
    return make_pair(x * c, y * c);
}

int main() {
    int a = 100, b = 128;
    int c = __gcd(a, b);
    int x = bezout(a, b);
    int y = (c - a * x) / b;
    cout << x << ' ' << y << endl;
    pair<int, int> xy = solve(100, 128, 40);
    cout << xy.first << ' ' << xy.second << endl;
    return 0;
}
```

## 1.2 System of linear equations

```
// extended version, uses diophantine equation solver to solve system of congruent equations
pair<int, int> solve(int a, int b, int c) {
        // solve ax + by == c
        int d = __gcd(a, b);
        int x = bezout(a / d, b / d);
        int y = (d - a * x) / b;
        c /= d;
        return make_pair(x * c, y * c);
}

int lcm(int a, int b) {
        return a / __gcd(a, b) * b;
}

int solveSystem(vector<int> a, vector<int> b) {
        // xi mod bi = ai
        int A = a[0], B = b[0];
        // x mod B = A
        for (int i = 1; i < a.size(); ++i) {
                int curB = b[i], curA = a[i];
                // x = Bi + A = curB * j + curA
                pair<int, int> ij = solve(B, -curB, curA - A);
                assert(B * ij.first + A == curB * ij.second + curA);
                int newA = (B * ij.first + A);
                B = lcm(B, curB);
                A = newA % B;
                if (i + 1 == a.size()) return A;
        }
}

int main() {
        vector<int> a = {0, 3, 3};
        vector<int> b = {3, 6, 9};
        cout << solveSystem(a, b) << endl;
        return 0;
}
```

# 2 Combinatorial optimization

# 3 Geometry

# 4 Numerical algorithms

## 4.1 Simplex Algorithm

```
typedef long double ld;

const ld EPS = 1e-8;

struct LPSolver {
    static vector<ld> simplex(vector<vector<ld>> a) {
        int n = (int) a.size() - 1;
        int m = (int) a[0].size() - 1;
        vector<int> left(n + 1);
        vector<int> up(m + 1);
        iota(left.begin(), left.end(), m);
        iota(up.begin(), up.end(), 0);
        auto pivot = [&](int x, int y) {
            swap(left[x], up[y]);
            ld k = a[x][y];
            a[x][y] = 1;
            vector<int> pos;
            for (int j = 0; j <= m; j++) {
                a[x][j] /= k;
                if (fabs(a[x][j]) > EPS) {
                    pos.push_back(j);
                }
            }
            for (int i = 0; i <= n; i++) {
                if (fabs(a[i][y]) < EPS || i == x) {
                    continue;
                }
                k = a[i][y];
                a[i][y] = 0;
                for (int j : pos) {
                    a[i][j] -= k * a[x][j];
                }
            }
        };
        while (1) {
            int x = -1;
            for (int i = 1; i <= n; i++) {
                if (a[i][0] < -EPS && (x == -1 || a[i][0] < a[x][0])) {
                    x = i;
                }
            }
            if (x == -1) {
                break;
            }
            int y = -1;
```

```
        for (int j = 1; j <= m; j++) {
            if (a[x][j] < -EPS && (y == -1 || a[x][j] < a[x][y])) {
                y = j;
            }
        }
        if (y == -1) {
            return vector<ld>(); // infeasible
        }
        pivot(x, y);
    }
    while (1) {
        int y = -1;
        for (int j = 1; j <= m; j++) {
            if (a[0][j] > EPS && (y == -1 || a[0][j] > a[0][y])) {
                y = j;
            }
        }
        if (y == -1) {
            break;
        }
        int x = -1;
        for (int i = 1; i <= n; i++) {
            if (a[i][y] > EPS && (x == -1 || a[i][0] / a[i][y] < a[x][0] / a[x][y])) {
                x = i;
            }
        }
        if (x == -1) {
            return vector<ld>(); // unbounded
            }
            pivot(x, y);
        }
        vector<ld> ans(m + 1);
        for (int i = 1; i <= n; i++) {
            if (left[i] <= m) {
                ans[left[i]] = a[i][0];
            }
        }
        ans[0] = -a[0][0];
        return ans;
    }
};
```

# 5 Graph algorithms

# 6 Data structures

# 7 Miscellaneous