Dracarys

Standard Code Library

β version

October, 2014

Contents

1 Simplex Algorithm

$\max\{cx|Ax \leq b, x \geq 0\}$
if there is no solution or no bound, return a empty vector

```
1  vector<double> simplex(vector<vector<double> > A, vector<double> b, vector<double> c)
2  {
3      int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
4      vector<vector<double> > D(n + 2, vector<double>(m + 1, 0));
5      vector<int> ix(n + m);
6      for (int i = 0; i < n + m; ++ i) ix[i] = i;
7      for (int i = 0; i < n; ++ i) {
8          for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
9          D[i][m - 1] = 1;
10         D[i][m] = b[i];
11         if (D[r][m] > D[i][m]) r = i;
12     }
13     for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
14     D[n + 1][m - 1] = -1;
15     for (double d; ; ) {
16         if (r < n) {
17             int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
18             D[r][s] = 1.0 / D[r][s];
19             vector<int> speedUp;
20             for (int j = 0; j <= m; ++ j) if (j != s) {
21                 D[r][j] *= -D[r][s];
22                 if(D[r][j]) {
23                     speedUp.push_back(j);
24                 }
25             }
26             for (int i = 0; i <= n + 1; ++ i) if (i != r) {
27                 for(int j = 0; j < speedUp.size(); ++ j)
28                     D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
29                 D[i][s] *= D[r][s];
30             }
31         }
32         r = -1; s = -1;
33         for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j]) {
34             if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)) s = j;
35         }
36         if (s < 0) break;
37         for (int i = 0; i < n; ++ i) if (D[i][s] < -EPS) {
38             if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS || (d <
                   EPS && ix[r + m] > ix[i + m]))
39                 r = i;
40         }
41         if (r < 0) return vector<double>(); // no bound
42     }
43     if (D[n + 1][m] < -EPS) return vector<double>(); // no solution
44     vector<double> x(m - 1);
```

```
45      for (int i = m; i < n + m; ++ i) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
46      return x; // max answer is D[n][m]
47  }
```

2 Fast Fourier Transform

$P = C * 2^k + 1$
G is primitive root of P

```
1  const int N = ;
2  const int P = 786433;
3  const int G = 10;
4
5  int modPow(long long a, int b, int c)
6  {
7      int ret = 1;
8      for( ; b; b >>= 1) {
9          if (b & 1)
10             ret = (long long)ret * a % c;
11         a = (long long)a * a % c;
12     }
13     return ret;
14 }
15
16 void dft(int *x, int on, int n)
17 {
18     int k, id, r, tmp, u, t;
19     for(int i = 1, j = n >> 1; i < n - 1; ++ i) {
20         if (i < j) swap(x[i], x[j]);
21         for(k = n >> 1; j >= k; j -= k, k >>= 1);
22         j += k;
23     }
24     for(int h = 2; h <= n; h <<= 1) {
25         r = modPow(G, (P - 1) / h, P);
26         if (on < 0) r = modPow(r, P - 2, P);
27         int p = h >> 1;
28         for(int j = 0; j < n; j += h) {
29             int w = 1;
30             for(int k = j; k < j + p; k ++) {
31                 u = x[k];
32                 id = k + p;
33                 t = (long long)w * x[id] % P;
34                 x[k] = (u + t) % P;
35                 x[id] = (u - t + P) % P;
36                 w = (long long)w * r % P;
37             }
38         }
39     }
40 }
41
42 int xa[N], xb[N];
43
44 void dft(int *a, int lenA, int *b, int lenB, int *ans, int &lenAns)
45 {
```

```
46      for(lenAns = 1; lenAns < lenA + lenB; lenAns <<= 1);
47      for(int i = 0; i < lenAns; ++ i) {
48          xa[i] = xb[i] = 0;
49      }
50      for(int i = 0; i < lenA; ++ i) {
51          xa[i] = a[i] % P;
52      }
53      for(int i = 0; i < lenB; ++ i) {
54          xb[i] = b[i] % P;
55      }
56
57      dft(xa, 1, lenAns);
58      dft(xb, 1, lenAns);
59      for(int i = 0; i < lenAns; ++ i) {
60          xa[i] = (long long)xa[i] * xb[i] % P;
61      }
62      dft(xa, -1, lenAns);
63      int tmp = modPow(lenAns, P - 2, P);
64      for(int i = 0; i < lenAns; ++ i) {
65          ans[i] = (long long)xa[i]* tmp % P;
66      }
67  }
```

3 Geometry Extended

get a s->t plane satisfied $ax + by + c \geq 0$
return -1, 0, 1

```
int getHalfPlane(LD a, LD b, LD c, Point &s, Point &t)
{
    if (sign(a)) {
        s.y = 0;
        s.x = -c / a;
    } else if (sign(b)) {
        s.x = 0;
        s.y = -c / b;
    } else {
        if (sign(c) < 0) return -1;
        return 1;
    }
    t = s + Point(b, -a);
    return 0;
}
```

get a line s-t satisfied $ax + by + c = 0, a^2 + b^2 \neq 0$

```
void getLine(LD a, LD b, LD c, Point &s, Point &t)
{
    if (sign(a)) {
        s.y = 0;
        s.x = -c / a;
    } else {
        s.x = 0;
        s.y = -c / b;
    }
    t = s + Point(b, -a);
}
```

4 Dominator Tree

edge ,n ,r , 1-based,realdom dominator tree father, realdom -1

```
1  const int maxn = 100000 + 10;
2
3  int n, m, r;
4  int parent[maxn], label[maxn], cnt, real[maxn];
5  vector<int> edge[maxn], succ[maxn], pred[maxn];
6  int semi[maxn], idom[maxn], ancestor[maxn], best[maxn];
7  vector<int> bucket[maxn];
8
9  int realdom[maxn];
10
11 void dfs(int u) {
12     label[u] = ++cnt; real[cnt] = u;
13     for (vector<int>::iterator it = edge[u].begin(); it != edge[u].end(); ++it) {
14         int v = *it;
15         if (v == parent[u] || label[v] != -1) continue;
16         parent[v] = u;
17         dfs(v);
18     }
19 }
20
21 void link(int v, int w) {
22     ancestor[w] = v;
23 }
24
25 void compress(int v) {
26     int a = ancestor[v];
27     if (ancestor[a] == 0) return;
28     compress(a);
29     if (semi[best[v]] > semi[best[a]]) best[v] = best[a];
30     ancestor[v] = ancestor[a];
31 }
32
33 int eval(int v) {
34     if (ancestor[v] == 0) return v;
35     compress(v);
36     return best[v];
37 }
38
39 void dominator() { // clear succ & pred & parent[r], let cnt = 0 first
40     cnt = 0;
41     for(int i = 1; i <= n; ++ i) {
42         succ[i].clear();
43         pred[i].clear();
44     }
45     for (int i = 1; i <= n; ++i) label[i] = -1;
46     parent[r] = -1;
```

```
47      dfs(r); // r is root
48      for (int u = 1; u <= n; ++u) {
49          for (vector<int>::iterator it = edge[u].begin(); it != edge[u].end(); ++it) {
50              int v = *it;
51              if (label[u] != -1 && label[v] != -1) {
52                  succ[label[u]].push_back(label[v]);
53                  pred[label[v]].push_back(label[u]);
54              }
55          }
56      }
57      for (int i = 1; i <= n; ++i) {
58          semi[i] = best[i] = i;
59          idom[i] = ancestor[i] = 0;
60          bucket[i].clear();
61      }
62      for (int w = cnt; w >= 2; --w) {
63          int p = label[parent[real[w]]];
64          for (vector<int>::iterator it = pred[w].begin(); it != pred[w].end(); ++it) {
65              int v = *it;
66              int u = eval(v);
67              if (semi[w] > semi[u]) semi[w] = semi[u];
68          }
69          bucket[semi[w]].push_back(w);
70          link(p, w);
71          for(int i = 0; i < bucket[p].size(); ++ i) {
72              int v = bucket[p][i];
73              int u = eval(v);
74              idom[v] = (semi[u] < p ? u : p);
75          }
76          bucket[p].clear();
77      }
78      for (int w = 2; w <= cnt; ++w) {
79          if (idom[w] != semi[w]) idom[w] = idom[idom[w]];
80      }
81      idom[1] = 0;
82      for(int i = 1; i <= n; ++ i) {
83          realdom[i] = -1;
84      }
85      for (int i = 2; i <= cnt; ++i) {
86          int u = real[idom[i]], v = real[i];
87          // u is immediate dominator of v (i == 1?)
88          realdom[v] = u;
89      }
90  }
```

5 vimrc

```
1  set nu ai ci si mouse=a ts=4 sts=4 sw=4
2
3  nmap<C-A> ggVG
4  vmap<C-C> "+y
5
6  nmap<F3> : vs %<.in <CR>
7  nmap<F4> : !gedit % <CR>
8  nmap<F5> : !./%< <CR>
9  nmap<F8> : !./%< < %<.in <CR>
10 nmap<F9> : make %< <CR>
11
12 "nmap<F9> :!export CXXFLAGS=-Wall && make %< <CR>
13 "autocmd BufNewFile *.cpp 0r ~/temp.cpp
14 "set hlsearch incseach
15
16 "syntax on
17 "filetype plugin indent on
```