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 vect

```
vector<double> simplex(vector<vector<double> > A, vector<d
    vector<double> c)
{
  int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
  vector<vector<double> > D(n + 2, vector<double>(m + 1, 0
  vector<int> ix(n + m);
  for (int i = 0; i < n + m; ++ i) ix[i] = i;
  for (int i = 0; i < n; ++ i) {
    for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
    D[i][m - 1] = 1;
    D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d; ; ) {
```

```
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      D[r][s] = 1.0 / D[r][s];
      vector<int> speedUp;
      for (int j = 0; j <= m; ++ j) if (j != s) {
        D[r][j] *= -D[r][s];
        if(D[r][j]) {
          speedUp.push_back(j);
        }
      }
      for (int i = 0; i <= n + 1; ++ i) if (i != r) {
        for(int j = 0; j < speedUp.size(); ++ j)
          D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
        D[i][s] *= D[r][s];
      }
    }
    r = -1; s = -1;
    for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j]) {
      if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS))
          s = j;
    }
    if (s < 0) break;
    for (int i = 0; i < n; ++ i) if (D[i][s] < -EPS) {
      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]))
        r = i;
    }
    if (r < 0) return vector<double>(); // no bound
  }
  if (D[n + 1][m] < -EPS) return vector<double>(); // no solution
  vector<double> x(m - 1);
  for (int i = m; i < n + m; ++ i) if (ix[i] < m - 1) x[ix[i]] = D[i
     - m][m];
  return x; // max answer is D[n][m]
}
```

2 Fast Fourier Transform

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

```
const int N = ;
const int P = 786433;
const int G = 10;

int modPow(long long a, int b, int c)
{
  int ret = 1;
  for( ; b; b >>= 1) {
    if (b & 1)
      ret = (long long)ret * a % c;
    a = (long long)a * a % c;
  }
  return ret;
}

void dft(int *x, int on, int n)
{
  int k, id, r, tmp, u, t;
  for(int i = 1, j = n >> 1; i < n - 1; ++ i) {
    if (i < j) swap(x[i], x[j]);
    for(k = n >> 1; j >= k; j -= k, k >>= 1);
    j += k;
  }
  for(int h = 2; h <= n; h <<= 1) {
    r = modPow(G, (P - 1) / h, P);
    if (on < 0) r = modPow(r, P - 2, P);
    int p = h >> 1;
    for(int j = 0; j < n; j += h) {
      int w = 1;
      for(int k = j; k < j + p; k ++) {
```

```
        u = x[k];
        id = k + p;
        t = (long long)w * x[id] % P;
        x[k] = (u + t) % P;
        x[id] = (u - t + P) % P;
        w = (long long)w * r % P;
      }
    }
  }
}

int xa[N], xb[N];

void dft(int *a, int lenA, int *b, int lenB, int *ans, int &lenAns)
{
  for(lenAns = 1; lenAns < lenA + lenB; lenAns <<= 1);
  for(int i = 0; i < lenAns; ++ i) {
    xa[i] = xb[i] = 0;
  }
  for(int i = 0; i < lenA; ++ i) {
    xa[i] = a[i] % P;
  }
  for(int i = 0; i < lenB; ++ i) {
    xb[i] = b[i] % P;
  }

  dft(xa, 1, lenAns);
  dft(xb, 1, lenAns);
  for(int i = 0; i < lenAns; ++ i) {
    xa[i] = (long long)xa[i] * xb[i] % P;
  }
  dft(xa, -1, lenAns);
  int tmp = modPow(lenAns, P - 2, P);
  for(int i = 0; i < lenAns; ++ i) {
    ans[i] = (long long)xa[i]* tmp % P;
  }
}
```

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

```
const int maxn = 100000 + 10;

int n, m, r;
int parent[maxn], label[maxn], cnt, real[maxn];
vector<int> edge[maxn], succ[maxn], pred[maxn];
int semi[maxn], idom[maxn], ancestor[maxn], best[maxn];
vector<int> bucket[maxn];

int realdom[maxn];

void dfs(int u) {
  label[u] = ++cnt; real[cnt] = u;
  for (vector<int>::iterator it = edge[u].begin(); it != e
      (); ++it) {
    int v = *it;
    if (v == parent[u] || label[v] != -1) continue;
    parent[v] = u;
    dfs(v);
  }
}

void link(int v, int w) {
  ancestor[w] = v;
}

void compress(int v) {
  int a = ancestor[v];
  if (ancestor[a] == 0) return;
  compress(a);
  if (semi[best[v]] > semi[best[a]]) best[v] = best[a];
  ancestor[v] = ancestor[a];
```

```
}

int eval(int v) {
  if (ancestor[v] == 0) return v;
  compress(v);
  return best[v];
}

void dominator() { // clear succ & pred & parent[r], let cnt = 0
    first
  cnt = 0;
  for(int i = 1; i <= n; ++ i) {
    succ[i].clear();
    pred[i].clear();
  }
  for (int i = 1; i <= n; ++i) label[i] = -1;
  parent[r] = -1;
  dfs(r); // r is root
  for (int u = 1; u <= n; ++u) {
    for (vector<int>::iterator it = edge[u].begin(); it != edge[u].
        end(); ++it) {
      int v = *it;
      if (label[u] != -1 && label[v] != -1) {
        succ[label[u]].push_back(label[v]);
        pred[label[v]].push_back(label[u]);
      }
    }
  }
  for (int i = 1; i <= n; ++i) {
    semi[i] = best[i] = i;
    idom[i] = ancestor[i] = 0;
    bucket[i].clear();
  }
  for (int w = cnt; w >= 2; --w) {
    int p = label[parent[real[w]]];
    for (vector<int>::iterator it = pred[w].begin(); it !=
        end(); ++it) {
      int v = *it;
      int u = eval(v);
      if (semi[w] > semi[u]) semi[w] = semi[u];
    }
    bucket[semi[w]].push_back(w);
    link(p, w);
    for(int i = 0; i < bucket[p].size(); ++ i) {
      int v = bucket[p][i];
      int u = eval(v);
      idom[v] = (semi[u] < p ? u : p);
    }
    bucket[p].clear();
  }
  for (int w = 2; w <= cnt; ++w) {
    if (idom[w] != semi[w]) idom[w] = idom[idom[w]];
  }
  idom[1] = 0;
  for(int i = 1; i <= n; ++ i) {
    realdom[i] = -1;
  }
  for (int i = 2; i <= cnt; ++i) {
    int u = real[idom[i]], v = real[i];
    // u is immediate dominator of v (i == 1?)
    realdom[v] = u;
  }
}
```

5 vimrc

```
set nu ai ci si mouse=a ts=4 sts=4 sw=4

nmap<C-A> ggVG
vmap<C-C> "+y

nmap<F3> :vs %<.in <CR>
nmap<F4> :!gedit %<CR>
nmap<F5> :!./%< <CR>
nmap<F8> :!./%< < %<.in <CR>
nmap<F9> :make %< <CR>

"nmap<F9> :!export CXXFLAGS=-Wall && make %< <CR>
"autocmd BufNewFile *.cpp 0r ~/temp.cpp
"set hlsearch incseach

"syntax on
"filetype plugin indent on
```