代码模版库

目录

1 比赛配置

1.1 代码库校验和

计算每一行忽略空白字符和//注释后内容的 MD5 Hash。使用 `python checksum.py < code.cpp` 打印校验和, 并与代码库侧面的数值比对。

```
c502 import re, sys, hashlib
427e
b41f def digest_line(s):
d74e     return hashlib.md5(re.sub(r'\s|//.*', '', s)).hexdigest()[-4:]
427e
f7db for line in sys.stdin.read().strip().split("\n"):
f335     print digest_line(line), line
```

1.2 Vim 配置文件

```
914c set nocompatible
7db5 set number
57b2 set ruler
9832 set showcmd
e416 set autoindent
7232 set cindent
740c set smartindent
5913 set shiftwidth=4
```

1.3 堆栈外挂

将堆栈指针指向用户空间的缓存中从而获得无限大的堆栈。注意在返回前将堆栈指针恢复，否则将出现 Segmentation Fault。

```
bebe const int STK_SZ = 2000000; // 堆栈的格数
effc char STK[STK_SZ * sizeof(void*)];
4e99 void *STK_BAK;
427e
427e // 32/64位自动检测
7bc9 #if defined(__i386__)
afcf   define SP "%%esp"
ac7a #elif defined(__x86_64__)
a9ea   #define SP "%%rsp"
1937 #endif
427e
3117 int main() {
427e   // 将堆栈指针移动到用户缓存
3750   asm volatile("mov␣" SP ",%0;␣mov␣%1," SP: "=g"(STK_BAK):"g"(STK+sizeof(STK)):)
         ;
427e   // 这时候就可以愉快地递归了!
427e   // 将堆栈指针恢复
6856   asm volatile("mov␣%0," SP::"g"(STK_BAK));
7021   return 0;
95cf }
```

1.4 I/O 外挂

快速整数输入输出，只支持非负整数。

```
6540 #define BUFSIZE 20000000
eb51 char buf[BUFSIZE], *pt = buf;
4e24 #define scan(t) do { \
d4df   int t = 0; \
2760   while (!((*pt) >= '0' && (*pt) <= '9')) pt ++; \
7181   while (((*pt) >= '0' && (*pt) <= '9')) t = t * 10 + (*(pt ++)) - '0'; \
5bd0 } while (0)
427e
3117 int main() {
486b   fread(buf, 1, BUFSIZE, stdin);
b496   scan(N); scan(M); // ...
95cf }
```

1.5 编译优化外挂

开启 GCC 的 O2 编译优化。

```
de4b #pragma GCC optimize("O2")
```

2 组合优化

2.1 上下界最小费用可行流

```
193c /* 支持上下界的最小费用S-T可行流
89b6  * 可兼作高效最小费用最大流实现
9cf5  * template
bff2  *   <class W>     - the weight type
```

```
5a99  * struct MinCostFlow
0cc6  *
33d6  * G.init(         — init graph
f4d6  *    source node id,
815d  *    sink node id,
5053  *    node count,  — number of nodes, 0-based
4113  *    epsilon,     — 0 for integer, 1e-6 for floats
fc78  *    inf,         — the weight upper bound
49d4  * )
0f35  * G.insert(       — insert an edge
00e3  *    u,           — first vertex
915b  *    v,           — second vertex
e6b0  *    lb,          — flow lower bound
7c4f  *    ub,          — flow upper bound
b5bb  *    cost         — unit edge cost (type is W)
49d4  * )
8e08  * G.work() returns the minimum cost.
f2b5  */
e0a5  #include<iostream>
59b9  #include<cstdio>
54ff  #include<algorithm>
09f7  #include<vector>
acb9  #include<queue>
b93d  #include<complex>
421c  using namespace std;
427e
4aa1  template <class W>
4e9f  struct MinCostFlow {
7d06    struct Edge {
5eea      int u, v, cap, nxt;
df79      W cost;
2a9c      Edge &set(int _u,int _v,int _nxt,int _cap,W _cost) {
bdae        u=_u;v=_v;cost=_cost;cap=_cap;nxt=_nxt;
a09f        return *this;
95cf      }
329b    };
427e
bb73    vector<Edge> e;
2bbc    vector<int> vst, head, que;
3049    vector<W> dist;
edec    vector<bool> mark;
2824    int st, en, N, tot, ret;
79c7    W val, INF, EPS;
427e
```

```
d35b  void init(int _s, int _t, int _n, W eps, W inf) {
5985    head.resize(_n);
d804    vst.resize(_n);
5d47    dist.resize(_n);
3f3a    que.resize(_n);
0117    mark.resize(_n);
cb76    st=_s;en=_t;N=_n;EPS=eps;INF=inf;
e965    ret=0;val=0;tot=0;fill(head.begin(),head.end(),-1);
95cf  }
427e
1e3a  int dfs(int v, int cap) {
99eb    if(v == en) {
e949      val += cap * dist[st];
5a51      return cap;
95cf    }
84ad    vst[v]=true;
21d4    int flow=0;
81fb    for(int i=head[v];i!=-1;i=e[i].nxt)
5a4d      if(e[i].cap && ! vst[e[i].v] && abs(dist[e[i].v]+e[i].cost-dist[v])<=EPS)
            {
5c56        int det = dfs(e[i].v, min(cap, e[i].cap));
21fc        if(det) {
3d4b          e[i].cap -= det; e[i^1].cap += det; flow += det;
0a05          if(!(cap-=det))break;
95cf        }
95cf      }
84fb    return flow;
95cf  }
f8b9  int relabel() {
c62d    W det = INF;
a98a    for(int i=0;i<N;++i)
2e8b      if(vst[i])
2591        for(int j=head[i];j!=-1;j=e[j].nxt)
79a8          if(e[j].cap && !vst[e[j].v])
f7cd            det = min(det, dist[e[j].v]+e[j].cost-dist[i]);
6593    if(det==INF) return false;
a98a    for(int i=0;i<N;++i)
488b      if(vst[i])dist[i]+=det;
3361    return true;
95cf  }
e54d  int spfa() {
2d49    queue<int> que;
a053    fill(vst.begin(),vst.end(),-1);
6210    vst[en] = -2;
```

```
3217      fill(dist.begin(),dist.end(),INF); dist[en]=0;
2602      que.push(en);
87f6      fill(mark.begin(),mark.end(),false);
6953      while(!que.empty()) {
3c69        int u = que.front(); que.pop();
4c72        mark[u]=false;
794c        for(int i=head[u];i!=-1;i=e[i].nxt)
07ec          if(e[i^1].cap && dist[u]+e[i^1].cost<dist[e[i].v]) {
c7e4            dist[e[i].v]=dist[u]+e[i^1].cost;vst[e[i].v]=i^1;
52c5            if(!mark[e[i].v]) {
a4dc              mark[e[i].v]=true;
290e              que.push(e[i].v);
95cf            }
95cf          }
95cf      }
95cf    }
e8cd    void insert(int u, int v,int cap, W cost) {
a85c      e.push_back(Edge().set(u,v,head[u],cap,cost)); head[u]=tot++;
b4d8      e.push_back(Edge().set(v,u,head[v],0,-cost)); head[v]=tot++;
95cf    }
3f4c    void insert(int u, int v, int lo, int hi, W cost) {
426c      if(hi > lo)
00f0        insert(u, v, hi-lo, cost);
7fd2      if(lo > 0) {
89ad        insert(st, v, lo, cost);
d921        insert(u, en, lo, 0);
95cf      }
95cf    }
7639    W work() {
81fc      spfa();
a69f      do {
a69f        do {
ef2b          fill(vst.begin(),vst.end(),false);
42c9        } while(dfs(st, 2147483647));
433d      } while(relabel());
3076      return val;
95cf    }
329b  };
```

2.2 线性规划 (单纯形)

```
427e  // Simplex for linear programming
427e  // min. A(0, 0:n-1)
      // s.t. A(1:m, 0:n-1) x <= A(1:m, n)                                427e
                                                                          427e
      #define lp for(;;)                                                  4fbb
      #define repf(i,a,b)                                                 70b7
      #define ft(i,a,b) for (int i=(a);i<=(b);++i)                        053e
      #define rep(i,n) for (int i=0;i<(n);++i)                            3977
      #define rtn return                                                  f1cf
      #define pb push_back                                                0938
      #define mp make_pair                                                d471
      #define sz(x) (int((x).size()))                                     bdad
      typedef double db;                                                  f7dc
      typedef vector<int> vi;                                             76b3
      db inf=1e+10;                                                       e8d0
      db eps=1e-10;                                                       8cfd
      inline int sgn(const db& x){rtn (x>+eps)-(x<-eps);}                 f4db
                                                                          427e
      const int MAXN=500;                                                 ce3a
      const int MAXM=501;                                                 68bc
      int n,m;                                                            35b8
      db A[MAXM+1][MAXN+1],X[MAXN];                                       d126
      int basis[MAXM+1],out[MAXN+1];                                      dead
                                                                          427e
      void pivot(int a,int b) {                                           83d4
        ft(i,0,m) if (i!=a&&sgn(A[i][b])) ft(j,0,n)                       ada3
          if (j!=b) A[i][j]-=A[a][j]*A[i][b]/A[a][b];                     1e9c
        ft(j,0,n) if (j!=b) A[a][j]/=A[a][b];                             75b7
        ft(i,0,m) if (i!=a) A[i][b]/=-A[a][b];                            1080
        A[a][b]=1/A[a][b];                                                6814
        swap(basis[a],out[b]);                                            531b
      }                                                                   95cf
      db simplex() {                                                      6340
        rep(j,n) A[0][j]=-A[0][j];                                        1c55
        ft(i,0,m) basis[i]=-i;                                            c429
        ft(j,0,n) out[j]=j;                                               6889
        lp {                                                              f08e
          int ii=1,jj=0;                                                  a898
          ft(i,1,m) if (mp(A[i][n],basis[i])<mp(A[ii][n],basis[ii])) ii=i;  22bd
          if (A[ii][n]>=0) break;                                         5129
          rep(j,n) if (A[ii][j]<A[ii][jj]) jj=j;                          f5ca
          if (A[ii][jj]>=0) rtn -inf;                                     e2e5
          pivot(ii,jj);                                                   8eb2
        }                                                                 95cf
        lp {                                                              f08e
          int ii=1,jj=0;                                                  a898
```

```
be8e      rep(j,n) if (mp(A[0][j],out[j])<mp(A[0][jj],out[jj])) jj=j;
55c8      if (A[0][jj]>=0) break;
5cdf      ft(i,1,m)
f105        if (A[i][jj]>0&&(A[ii][jj]<=0||mp(A[i][n]/A[i][jj],basis[i])
8599          <mp(A[ii][n]/A[ii][jj],basis[ii])))
0fc0          ii=i;
7bbe      if (A[ii][jj]<=0) rtn +inf;
8eb2      pivot(ii,jj);
95cf    }
16f0    rep(j,n) X[j]=0;
9394    ft(i,1,m) if (basis[i]>=0) X[basis[i]]=A[i][n];
329f    rtn A[0][n];
95cf  }
```

3 Watashi 代码库 (备用)

3.1 $O(n \log n) - O(1)$ RMQ

```
9581  #include <climits>    // CHAR_BIT
54ff  #include <algorithm>  // copy
427e
421c  using namespace std;
427e
b7ec  template<typename T>
1f3e  struct RMQ {
5c83    int n;
bd3a    vector<T> e;
f687    vector<vector<int> > rmq;
427e
de2b    static const int INT_BIT = sizeof(4) * CHAR_BIT;
eeb3    static inline int LG2(int i) { return INT_BIT - 1 - __builtin_clz(i); }
449d    static inline int BIN(int i) { return 1 << i; }
427e
6413    int cmp(int l, int r) const {
cdf9      return e[l] <= e[r] ? l : r;
95cf    }
427e
01c5    void init(int n, const T e[]) {
b985      this->n = n;
40f2      vector<T>(e, e + n).swap(this->e);
427e
dcba      int m = 1;
f312      while (BIN(m) <= n) {
1114        ++m;
95cf      }
1730      vector<vector<int> >(m, vector<int>(n)).swap(rmq);
427e
6c2f      for (int i = 0; i < n; ++i) {
ac66        rmq[0][i] = i;
95cf      }
b0ef      for (int i = 0; BIN(i + 1) <= n; ++i) {
6941        for (int j = 0; j + BIN(i + 1) <= n; ++j) {
d8a7          rmq[i + 1][j] = cmp(rmq[i][j], rmq[i][j + BIN(i)]);
95cf        }
95cf      }
95cf    }
427e
7689    int index(int l, int r) const {
083c      int b = LG2(r - l);
db43      return cmp(rmq[b][l], rmq[b][r - (1 << b)]);
95cf    }
427e
6e14    T value(int l, int r) const {
72b5      return e[index(l, r)];
95cf    }
329b  };
```

3.2 $O(n \log n) - O(\log n)$ LCA

```
59b9  #include <cstdio>
09f7  #include <vector>
54ff  #include <algorithm>
427e
421c  using namespace std;
427e
3bd6  const int MAXM = 16;
b0bb  const int MAXN = 1 << MAXM;
427e
427e  // LCA
6dd4  struct LCA {
6910    vector<int> e[MAXN];
6058    int d[MAXN], p[MAXN][MAXM];
427e
de24    void dfs_(int v, int f) {
c4a8      p[v][0] = f;
```

```
67b8      for (int i = 1; i < MAXM; ++i) {
47b3        p[v][i] = p[p[v][i - 1]][i - 1];
95cf      }
6bea      for (int i = 0; i < (int)e[v].size(); ++i) {
2389        int w = e[v][i];
2732        if (w != f) {
1e13          d[w] = d[v] + 1;
3245          dfs_(w, v);
95cf        }
95cf      }
95cf    }
427e
6074    int up_(int v, int m) {
3b30      for (int i = 0; i < MAXM; ++i) {
1fe0        if (m & (1 << i)) {
bbaa          v = p[v][i];
95cf        }
95cf      }
aa78      return v;
95cf    }
427e
8119    int lca(int a, int b) {
e7ce      if (d[a] > d[b]) {
4012        swap(a, b);
95cf      }
66a3      b = up_(b, d[b] - d[a]);
59db      if (a == b) {
5ffd        return a;
8e2e      } else {
08b5        for (int i = MAXM - 1; i >= 0; --i) {
7ecb          if (p[a][i] != p[b][i]) {
434f            a = p[a][i];
f29b            b = p[b][i];
95cf          }
95cf        }
d21f        return p[a][0];
95cf      }
95cf    }
427e
d34f    void init(int n) {
6c2f      for (int i = 0; i < n; ++i) {
c9ff        e[i].clear();
95cf      }
95cf    }
```

```
427e
77e3    void add(int a, int b) {
486c      e[a].push_back(b);
800c      e[b].push_back(a);
95cf    }
427e
2114    void build() {
3dbe      d[0] = 0;
e5db      dfs_(0, 0);
95cf    }
a83c  } lca;
```

3.3 树状数组

```
09f7  #include <vector>
427e
421c  using namespace std;
427e
ad8a  template<typename T = int>
5f7d  struct BIT {
ab5c    vector<T> a;
427e
d34f    void init(int n) {
7f29      vector<T>(n + 1).swap(a);
95cf    }
427e
853b    void add(int i, T v) {
13ab      for (int j = i + 1; j < (int)a.size(); j = (j | (j - 1)) + 1) {
a45a        a[j] += v;
95cf      }
95cf    }
427e
427e    // [0, i)
9992    T sum(int i) const {
951c      T ret = T();
b76f      for (int j = i; j > 0; j = j & (j - 1)) {
2a23        ret += a[j];
95cf      }
ee0f      return ret;
95cf    }
427e
7517    T get(int i) const {
da02      return sum(i + 1) - sum(i);
```

```
95cf    }
427e
bb52    void set(int i, T v) {
7c32      add(i, v - get(i));
95cf    }
329b  };
```

3.4 并查集

```
09f7  #include <vector>
427e
421c  using namespace std;
427e
c793  struct DisjointSet {
7521    vector<int> p;
427e
d34f    void init(int n) {
9156      p.resize(n);
6c2f      for (int i = 0; i < n; ++i) {
a208        p[i] = i;
95cf      }
95cf    }
427e
8b08    int getp(int i) {
a584      return i == p[i] ? i : (p[i] = getp(p[i]));
95cf    }
427e
7fac    bool setp(int i, int j) {
8b35      i = getp(i);
e8c6      j = getp(j);
df09      p[i] = j;
29ee      return i != j;
95cf    }
329b  };
```

3.5 轻重权树剖分

```
59b9  #include <cstdio>
09f7  #include <vector>
54ff  #include <algorithm>
427e
421c  using namespace std;
427e
3bd6  const int MAXM = 16;
b0bb  const int MAXN = 1 << MAXM;
427e
427e  // Heavy-Light Decomposition
b49e  struct TreeDecomposition {
1769    vector<int> e[MAXN], c[MAXN];
4971    int s[MAXN];    // subtree size
9524    int p[MAXN];    // parent id
448c    int r[MAXN];    // chain root id
a6ab    int t[MAXN];    // timestamp, index used in segtree
d16b    int ts;
427e
de24    void dfs_(int v, int f) {
e350      p[v] = f;
5172      s[v] = 1;
6bea      for (int i = 0; i < (int)e[v].size(); ++i) {
2389        int w = e[v][i];
2732        if (w != f) {
3245          dfs_(w, v);
ea7a          s[v] += s[w];
95cf        }
95cf      }
95cf    }
427e
a22f    void decomp_(int v, int f, int k) {
fd9c      t[v] = ts++;
048a      c[k].push_back(v);
eb44      r[v] = k;
427e
e6d1      int x = 0, y = -1;
6bea      for (int i = 0; i < (int)e[v].size(); ++i) {
2389        int w = e[v][i];
2732        if (w != f) {
08e1          if (s[w] > x) {
3089            x = s[w];
35ea            y = w;
95cf          }
95cf        }
95cf      }
bee4      if (y != -1) {
f904        decomp_(y, v, k);
95cf      }
```

```
427e
6bea      for (int i = 0; i < (int)e[v].size(); ++i) {
2389        int w = e[v][i];
6e71        if (w != f && w != y) {
9768          decomp_(w, v, w);
95cf        }
95cf      }
95cf    }
427e
d34f    void init(int n) {
6c2f      for (int i = 0; i < n; ++i) {
c9ff        e[i].clear();
95cf      }
95cf    }
427e
77e3    void add(int a, int b) {
486c      e[a].push_back(b);
800c      e[b].push_back(a);
95cf    }
427e
2114    void build() {  // !!
d7ef      ts = 0;
e5db      dfs_(0, 0);
917b      decomp_(0, 0, 0);
95cf    }
8dca  } hld;
```

3.6 强连通分量

```
8207  #include <stack>
09f7  #include <vector>
54ff  #include <algorithm>
427e
421c  using namespace std;
427e
9356  struct SCCTarjan {
5c83    int n;
9d4c    vector<vector<int> > e;
427e
fbd2    vector<int> id;
e01d    vector<vector<int> > scc;
427e
d34f    void init(int n) {
b985      this->n = n;
4883      vector<vector<int> >(n).swap(e);
a1b9      id.resize(n);
94d6      dfn.resize(n);
021f      low.resize(n);
95cf    }
427e
77e3    void add(int a, int b) {
486c      e[a].push_back(b);
95cf    }
427e
5728    vector<int> dfn, low;
724e    int timestamp;
9cad    stack<int> s;
427e
3dd3    void dfs(int v) {
d2b0      dfn[v] = timestamp++;
daec      low[v] = dfn[v];
d819      s.push(v);
b9fe      for (vector<int>::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
650e        if (dfn[*w] == -1) {
0402          dfs(*w);
181c          low[v] = min(low[v], low[*w]);
e5cc        } else if (dfn[*w] != -2) {
f480          low[v] = min(low[v], dfn[*w]);
95cf        }
95cf      }
427e
01f5      if (low[v] == dfn[v]) {
8631        vector<int> t;
a69f        do {
5973          int w = s.top();
c2f4          s.pop();
11f8          id[w] = (int)scc.size();
7783          t.push_back(w);
b819          dfn[w] = -2;
e8fd        } while (t.back() != v);
6cf2        scc.push_back(t);
95cf      }
95cf    }
427e
3cca    int gao() {
efaa      scc.clear();
4810      stack<int>().swap(s);
```

```
    timestamp = 0;
    fill(dfn.begin(), dfn.end(), -1);
    for (int i = 0; i < n; ++i) {
      if (dfn[i] == -1) {
        dfs(i);
      }
    }
    return (int)scc.size();
  }
};
```

3.7 双连通分量

```
#include <stack>
#include <vector>
#include <utility>
#include <algorithm>

using namespace std;

// TODO: cannot handle duplicate edges
struct Tarjan {
  int n;
  vector<vector<int> > e;

  vector<int> cut;
  vector<pair<int, int> > bridge;
  vector<vector<pair<int, int> > > bcc;

  void init(int n) {
    this->n = n;
    e.clear();
    e.resize(n);
    dfn.resize(n);
    low.resize(n);
  }

  void add(int a, int b) {
    // assert(find(e[a].begin(), e[a].end(), b) == e[a].end());
    e[a].push_back(b);
    e[b].push_back(a);
  }

vector<int> dfn, low;
int timestamp;
stack<pair<int, int> > s;

void dfs(int v, int p) {
  int part = p == -1 ? 0 : 1;
  dfn[v] = low[v] = timestamp++;
  for (vector<int>::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
    pair<int, int> f = make_pair(min(v, *w), max(v, *w));
    if (dfn[*w] == -1) {
      s.push(f);
      dfs(*w, v);
      low[v] = min(low[v], low[*w]);
      if (dfn[v] <= low[*w]) {
        // articulation point
        if (++part == 2) {
          cut.push_back(v);
        }
        // articulation edge
        if (dfn[v] < low[*w]) {
          bridge.push_back(f);
        }
        // biconnected component (2-vertex-connected)
        vector<pair<int, int> > t;
        do {
          t.push_back(s.top());
          s.pop();
        } while (t.back() != f);
        bcc.push_back(t);
      }
    } else if (*w != p && dfn[*w] < dfn[v]) {
      s.push(f);
      low[v] = min(low[v], dfn[*w]);
    }
  }
}

void gao() {
  cut.clear();
  bridge.clear();
  bcc.clear();

  timestamp = 0;
```

```
ef66      stack<pair<int, int> >().swap(s);
d38c      fill(dfn.begin(), dfn.end(), -1);
427e
6c2f      for (int i = 0; i < n; ++i) {
e5ea        if (dfn[i] == -1) {
e343          dfs(i, -1);
95cf        }
95cf      }
95cf    }
329b  };
427e
427e
0f18  struct BridgeBlockTree {
43ad    Tarjan<MAXN> bcc;
76c3    DisjointSet<MAXN> ds;
6910    vector<int> e[MAXN];
427e
d34f    void init(int n) {
5e72      bcc.init(n);
37f8      ds.init(n);
95cf    }
427e
77e3    void add(int a, int b) {
1cf2      bcc.add(a, b);
95cf    }
427e
eb55    void gao() {
e6ec      bcc.gao();
87a1      for (const auto& i: bcc.bcc) {
75dc        if (i.size() > 1) {
3273          for (const auto& j: i) {
e387            ds.setp(j.first, j.second);
95cf          }
95cf        }
95cf      }
cdda      for (const auto& i: bcc.bridge) {
28e1        int a = ds.getp(i.first);
9ba6        int b = ds.getp(i.second);
486c        e[a].push_back(b);
800c        e[b].push_back(a);
95cf      }
95cf    }
427e
c12a    int id(int v) {
deff      return ds.getp(v);
95cf    }
329b  };
```

3.8 二分图匹配

```
427e  // maximum matchings in bipartite graphs
427e  // maximum cardinality bipartite matching
427e  // O(|V||E|), generally fast
427e
09f7  #include <vector>
2349  #include <string>
54ff  #include <algorithm>
427e
421c  using namespace std;
427e
84ee  struct Hungarian {
fbf6    int nx, ny;
9ec6    vector<int> mx, my;
9d4c    vector<vector<int> > e;
427e
8324    void init(int nx, int ny) {
c1d1      this->nx = nx;
f9c1      this->ny = ny;
8789      mx.resize(nx);
50d0      my.resize(ny);
c568      e.clear();
25b9      e.resize(nx);
1023      mark.resize(nx);
95cf    }
427e
77e3    void add(int a, int b) {
486c      e[a].push_back(b);
95cf    }
427e
427e    // vector<bool> is evil!!!
50e0    basic_string<bool> mark;
427e
0c2b    bool augment(int i) {
207c      if (!mark[i]) {
dae4        mark[i] = true;
40c5        for (vector<int>::const_iterator j = e[i].begin(); j != e[i].end(); ++j) {
4d94          if (my[*j] == -1 || augment(my[*j])) {
```

```
159a            mx[i] = *j;
c291            my[*j] = i;
3361            return true;
95cf          }
95cf        }
95cf      }
438e      return false;
95cf    }
427e
3cca    int gao() {
5b57      int ret = 0;
103b      fill(mx.begin(), mx.end(), -1);
319d      fill(my.begin(), my.end(), -1);
d0cb      for (int i = 0; i < nx; ++i) {
87f6        fill(mark.begin(), mark.end(), false);
7c0c        if (augment(i)) {
8bb8          ++ret;
95cf        }
95cf      }
ee0f      return ret;
95cf    }
329b  };
```

3.9 最小费用最大流

```
acb9  #include <queue>
3160  #include <limits>
09f7  #include <vector>
59b9  #include <cstdio>
54ff  #include <algorithm>
427e
421c  using namespace std;
427e
9889  template<int MAXN, typename T = int, typename S = T>
e90e  struct MinCostMaxFlow {
7b7f    struct NegativeCostCircuitExistsException {
329b    };
427e
7d06    struct Edge {
3b67      int v;
4f94      T c;
3345      S w;
5b0e      int b;
7f53      Edge(int v, T c, S w, int b) : v(v), c(c), w(w), b(b) { }
329b  };
427e
b80f  int n, source, sink;
360a  vector<Edge> e[MAXN];
427e
41c4  void init(int n, int source, int sink) {
b985    this->n = n;
a099    this->source = source;
8a8f    this->sink = sink;
6c2f    for (int i = 0; i < n; ++i) {
c9ff      e[i].clear();
95cf    }
95cf  }
427e
46f2  void addEdge(int a, int b, T c, S w) {
7f83    e[a].push_back(Edge(b, c, w, e[b].size()));
0c8e    e[b].push_back(Edge(a, 0, -w, e[a].size() - 1));  // TODO
95cf  }
427e
cd81  bool mark[MAXN];
d6ad  T maxc[MAXN];
97c2  S minw[MAXN];
5e7d  int dist[MAXN];
e4d7  Edge* prev[MAXN];
427e
70a8  bool _spfa() {
93d2    queue<int> q;
fef6    fill(mark, mark + n, false);
eb72    fill(maxc, maxc + n, 0);
b992    fill(minw, minw + n, numeric_limits<S>::max());
d6ae    fill(dist, dist + n, 0);
5d6d    fill(prev, prev + n, (Edge*)NULL);
ce15    mark[source] = true;
ef7c    maxc[source] = numeric_limits<S>::max();
5ee3    minw[source] = 0;
427e
bd63    q.push(source);
cc78    while (!q.empty()) {
e23b      int cur = q.front();
ad9c      mark[cur] = false;
15dd      q.pop();
45d3      for (typename vector<Edge>::iterator it = e[cur].begin(); it != e[cur].end
            (); ++it) {
```

```
c425         T c = min(maxc[cur], it->c);
fd54         if (c == 0) {
b333           continue;
95cf         }
427e
757c         int v = it->v;
227a         S w = minw[cur] + it->w;
2767         if (minw[v] > w || (minw[v] == w && maxc[v] < c)) { // TODO
e8de           maxc[v] = c;
dbc3           minw[v] = w;
a779           dist[v] = dist[cur] + 1;
16f9           if (dist[v] >= n) {
438e             return false;
95cf           }
05f0           prev[v] = &*it;
8c3b           if (!mark[v]) {
065b             mark[v] = true;
eab7             q.push(v);
95cf           }
95cf         }
95cf       }
95cf     }
3361     return true;
95cf   }
427e
93fe   pair<T, S> gao() {
29a5     T sumc = 0;
9e20     S sumw = 0;
1026     while (true) {
9653       if (!_spfa()) {
c891         throw NegativeCostCircuitExistsException();
8767       } else if (maxc[sink] == 0) {
6173         break;
8e2e       } else {
3ee3         T c = maxc[sink];
109e         sumc += c;
27e0         sumw += c * minw[sink];
427e
6fb2         int cur = sink;
a9d1         while (cur != source) {
8a2b           Edge* e1 = prev[cur];
0cc2           e1->c -= c;
aad1           Edge* e2 = &e[e1->v][e1->b];
3d0a           e2->c += c;
84d7           cur = e2->v;
95cf         }
95cf       }
95cf     }
d460     return make_pair(sumc, sumw);
95cf   }
329b };
```

3.10 AhoCorasick 自动机

```
acb9 #include <queue>
54ff #include <algorithm>
427e
421c using namespace std;
427e
3de7 struct AhoCorasick {
36aa   static const int NONE = 0;
35e5   static const int MAXN = 1024;
2eee   static const int CHARSET = 26;
427e
4022   int end;
35a5   int tag[MAXN];
9143   int fail[MAXN];
f098   int trie[MAXN][CHARSET];
427e
5d53   void init() {
84fb     tag[0] = NONE;
572c     fill(trie[0], trie[0] + CHARSET, -1);
feb8     end = 1;
95cf   }
427e
a3fb   int add(int m, const int* s) {
539d     int p = 0;
5b80     for (int i = 0; i < m; ++i) {
e72e       if (trie[p][*s] == -1) {
6656         tag[end] = NONE;
0081         fill(trie[end], trie[end] + CHARSET, -1);
bb4e         trie[p][*s] = end++;
95cf       }
a009       p = trie[p][*s];
47f8       ++s;
95cf     }
e149     return p;
```

```
95cf    }
427e
2114    void build() {  // !!
dfc8      queue<int> bfs;
a7a6      fail[0] = 0;
3873      for (int i = 0; i < CHARSET; ++i) {
131c        if (trie[0][i] != -1) {
9b4d          fail[trie[0][i]] = 0;
79f5          bfs.push(trie[0][i]);
8e2e        } else {
6c43          trie[0][i] = 0;
95cf        }
95cf      }
88bb      while (!bfs.empty()) {
e42e        int p = bfs.front();
38ff        tag[p] |= tag[fail[p]];
1a76        bfs.pop();
3873        for (int i = 0; i < CHARSET; ++i) {
9f81          if (trie[p][i] != -1) {
076e            fail[trie[p][i]] = trie[fail[p]][i];
659d            bfs.push(trie[p][i]);
8e2e          } else {
7720            trie[p][i] = trie[fail[p]][i];
95cf          }
95cf        }
95cf      }
95cf    }
0244  } ac;
```

3.11 后缀数组

```
09f7  #include <vector>
0947  #include <utility>
54ff  #include <algorithm>
421c  using namespace std;
427e
010b  struct SuffixArray {
8a07    vector<int> sa, rank, height;
427e
b7ec    template<typename T>
e220    void init(int n, const T a[]) {
b104      sa.resize(n);
0be0      rank.resize(n);
427e
caf4      vector<pair<T, int> > assoc(n);
6c2f      for (int i = 0; i < n; ++i) {
7d91        assoc[i] = make_pair(a[i], i);
95cf      }
7c59      sort(assoc.begin(), assoc.end());
6c2f      for (int i = 0; i < n; ++i) {
079a        sa[i] = assoc[i].second;
bc80        if (i == 0 || assoc[i].first != assoc[i - 1].first) {
ff4f          rank[sa[i]] = i;
8e2e        } else {
dc5c          rank[sa[i]] = rank[sa[i - 1]];
95cf        }
95cf      }
427e
90d1      vector<int> tmp(n), cnt(n);
1d87      vector<pair<int, int> > suffix(n);
9a38      for (int m = 1; m < n; m <<= 1) {
427e        // snd
5b80        for (int i = 0; i < m; ++i) {
0ece          tmp[i] = n - m + i;
95cf        }
5c61        for (int i = 0, j = m; i < n; ++i) {
dcfb          if (sa[i] >= m) {
0041            tmp[j++] = sa[i] - m;
95cf          }
95cf        }
427e        // fst
c9ea        fill(cnt.begin(), cnt.end(), 0);
6c2f        for (int i = 0; i < n; ++i) {
e3e0          ++cnt[rank[i]];
95cf        }
6da6        partial_sum(cnt.begin(), cnt.end(), cnt.begin());
1894        for (int i = n - 1; i >= 0; --i) {
8376          sa[--cnt[rank[tmp[i]]]] = tmp[i];
95cf        }
427e        //
6c2f        for (int i = 0; i < n; ++i) {
cc5d          suffix[i] = make_pair(rank[i],
a959            i + m < n ? rank[i + m] : numeric_limits<int>::min());
95cf        }
6c2f        for (int i = 0; i < n; ++i) {
e1ef          if (i == 0 || suffix[sa[i]] != suffix[sa[i - 1]]) {
ff4f            rank[sa[i]] = i;
```

```
8e2e         } else {
dc5c           rank[sa[i]] = rank[sa[i - 1]];
95cf         }
95cf       }
95cf     }
427e
3ce6     height.resize(n);
ef3c     for (int i = 0, z = 0; i < n; ++i) {
d157       if (rank[i] == 0) {
39a0         height[0] = z = 0;
8e2e       } else {
691f         int x = i, y = sa[rank[i] - 1];
b5d4         z = max(0, z - 1);
af70         while (x + z < n && y + z < n && a[x + z] == a[y + z]) {
aba4           ++z;
95cf         }
109c         height[rank[i]] = z;
95cf       }
95cf     }
95cf   }
329b };
```

3.12 LU 分解

```
6e2d const int MAXN = 128;
c726 const double EPS = 1e-10;
427e
1806 void LU(int n, double a[MAXN][MAXN], int r[MAXN], int c[MAXN]) {
6c2f   for (int i = 0; i < n; ++i) {
178d     r[i] = c[i] = i;
95cf   }
cde9   for (int k = 0; k < n; ++k) {
56c5     int ii = k, jj = k;
1bf2     for (int i = k; i < n; ++i) {
4636       for (int j = k; j < n; ++j) {
b564         if (fabs(a[i][j]) > fabs(a[ii][jj])) {
0fc0           ii = i;
f3bf           jj = j;
95cf         }
95cf       }
95cf     }
9478     swap(r[k], r[ii]);
e7f0     swap(c[k], c[jj]);
6c2f     for (int i = 0; i < n; ++i) {
6fea       swap(a[i][k], a[i][jj]);
95cf     }
44fd     for (int j = 0; j < n; ++j) {
50eb       swap(a[k][j], a[ii][j]);
95cf     }
b775     if (fabs(a[k][k]) < EPS) {
b333       continue;
95cf     }
c190     for (int i = k + 1; i < n; ++i) {
a9f2       a[i][k] = a[i][k] / a[k][k];
3bee       for (int j = k + 1; j < n; ++j) {
56e2         a[i][j] -= a[i][k] * a[k][j];
95cf       }
95cf     }
95cf   }
95cf }
427e
c4d2 void solve(int n, double a[MAXN][MAXN], int r[MAXN], int c[MAXN], double b[MAXN
       ]) {
5bc8   static double x[MAXN];
6c2f   for (int i = 0; i < n; ++i) {
2c68     x[i] = b[r[i]];
95cf   }
6c2f   for (int i = 0; i < n; ++i) {
1226     for (int j = 0; j < i; ++j) {
99d9       x[i] -= a[i][j] * x[j];
95cf     }
95cf   }
1894   for (int i = n - 1; i >= 0; --i) {
34a7     for (int j = n - 1; j > i; --j) {
99d9       x[i] -= a[i][j] * x[j];
95cf     }
4459     if (fabs(a[i][i]) >= EPS) {
7360       x[i] /= a[i][i];
95cf     } // else assert(fabs(x[i]) < EPS);
95cf   }
6c2f   for (int i = 0; i < n; ++i) {
2a61     b[c[i]] = x[i];
95cf   }
95cf }
427e
427e // LU(n - 1, a, r, c);
427e // solve(n - 1, a, r, c, b);
```