Team Notebook

May 18, 2023

Contents

1 DS

1.1 dsu

```
int getpar(int v){
 return (par[v] ? par[v] = getpar(par[v]) : v);
}

void merge(int u ,int v){
 u = getpar(u) , v = getpar(v);
 if(u == v)return;
 par[u]=v;
}
```

1.2 fenwick

```
void add(int pos,int x){
    for(;pos<maxn;pos+=pos&(-pos))
        fen[pos]+=x;
}
int get(int pos){
    int ans = 0;
    for(;pos;pos-=pos&(-pos))
        ans+=fen[pos];
    return(ans);
}
```

1.3 pbds

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;

template <class T> using Tree = tree<T, null_type, less<T>,
     rb_tree_tag,tree_order_statistics_node_update>;

struct oset{ // just don't use with numbers <= 0
    int maxn;
    vector < int > fen;
    oset(int n):
        maxn(n+100),
        fen(maxn){}

    void add(int x , int pos){
        for( ; pos < maxn ; pos += pos & -pos)
            fen[pos] += x;
    }
    int get(int pos){
        int sum = 0 ;
        for( ; pos ; pos -= pos & -pos)
            sum += fen[pos];
        return(sum);
    }

    void insert(int x , int cnt = 1){
        add(cnt , x);
    }
    void erase(int x , int cnt = 1){
        add(-cnt , x);
    }

    int find_by_order(int k){ // k-th element
        int sum = 0 , pos = 0;
        for(int i = log2(maxn) ; i >= 0 ; i --)
            if(pos + (1 << i) < maxn and sum + fen[pos + (1
                 << i)] < k)
                pos += (1 << i),
                sum += fen[pos];
        return(pos + 1);
    }
    int order_of_key(int key){ // number of elements <= key
        return(get(key));
    }
};
```

1.4 segment

```
#define lc (v<<1)
#define rc (lc|1)
#define mid ((l+r)>>1)
struct segment{
    int seg[maxn<<2], lazy[maxn<<2];
    void build(int v = 1, int l = 1, int r = maxn){
        if(r - l == 1){
            seg[v] = a[rst[l]];
            return;
        }
        build(lc, l, mid);
        build(rc, mid, r);
        seg[v] = seg[lc] + seg[rc];
    }
    void shift(int v, int l, int r){
        if(!lazy[v])return;
        seg[v] += lazy[v];
        if(r - l == 1){
            lazy[v] = 0;
            return;
        }
        lazy[lc] += lazy[v];
        lazy[rc] += lazy[v];
        lazy[v] = 0;
    }
    void update(int L, int R, int val, int v = 1, int l = 1,
         int r = maxn){
        if(r <= L or R <= l)
            return;
        shift(v, l, r);
        if(L <= l and r <= R){
            lazy[v] += val;
            shift(v, l, r);
            return;
        }
        update(L, R, val, lc, l, mid);
        update(L, R, val, rc, mid, r);
        seg[v] = seg[lc] + seg[rc];
    }
    int query(int L, int R, int v = 1, int l = 1, int r =
         maxn){
        if(r <= L or R <= l)
            return 0;
        shift(v, l, r);
        if(L <= l and r <= R){
            return seg[v];
        }
        return query(L, R, lc, l, mid) + query(L, R, rc, mid,
              r);
    }
};
```

2 Graphs

2.1 2-sat

```
struct sat{ //v = 2*v , ~v = 2*v + 1 ==> ~v = v^1
 int n, c;
 vector < vector < int > > in , out;
 vector < int > col , topo;
 sat(int N):
  n(N) , c(0) , in(2*n + 5) , out(2*n + 5) , col(2*n + 5){}
 bool operator [] (int x) { return(col[2*x] > col[2*x + 1]);
       }
 void add_e(int v , int u){in[u].pb(v) , out[v].pb(u);}
```

```
void add(int v , int u){add_e(u^1 , v) , add_e(v^1 , u);}
void sfd(int v){
 col[v] = c;
 for(auto u : in[v]) if(!col[u])
  sfd(u);
}
void dfs(int v){
 col[v] = 1;
 for(auto u : out[v]) if(!col[u])
  dfs(u);
 topo.pb(v);
}
bool validate(){
 for(int i = 1 ; i <= 2*n+1 ; i ++ ) if(!col[i]) dfs(i);
 reverse(topo.begin() , topo.end());
 fill(col.begin() , col.end() , 0 );
 for(auto v : topo)
  if(!col[v])
   ++c , sfd(v);
 for(int i = 1 ; i <= n ; i ++) if(col[i * 2] == col[i * 2
     + 1])return(0);
return(1);
}
};
```

2.2 Dinic

```
#include <bits/stdc++.h>

using namespace std;

struct Dinic {
    #define MAXN 100010
    int n = 0, m = 0, turn = 0;
    vector < int > a, b, h, mark, pos, adj[MAXN];
    vector < int64_t > c, d;
    queue < int > q;
    void add_edge(int u, int v, int64_t w = 1) {
        u--, v--;
        adj[u].push_back(m);
        adj[v].push_back(m);
        a.push_back(u);
        b.push_back(v);
        c.push_back(w);
        m++;
        n = max(n, max(u, v) + 1);
    }
    void bfs(int v) {
        mark[v] = turn;
        int l = 0, r = 0;
        pos[r++] = v;
        h[v] = 0;
        while (l < r) {
            int v = pos[l++];
            for (int w: adj[v]) {
                if (a[w] == v and mark[b[w]] ^ turn and c[w] -
                     d[w] > 0) {
                    mark[b[w]] = turn, h[b[w]] = h[v] + 1;
                    pos[r++] = b[w];
                }
                if (b[w] == v and mark[a[w]] ^ turn and d[w] >
                     0) {
                    mark[a[w]] = turn, h[a[w]] = h[v] + 1;
                    pos[r++] = a[w];
                }
            }
        }
    }
    int64_t pump(int v, int source, int sink, int64_t cap =
         (1LL << 62)) {
        int64_t ans = 0;
        if (v == sink)
            return cap;
        if (v == source)
            turn++, bfs(v), fill(pos.begin(), pos.end(), 0);
        mark[v] = turn;
        for (; pos[v] < int(adj[v].size()); pos[v]++) {
            int w = adj[v][pos[v]];
            if (a[w] == v) {
                if (c[w] - d[w] == 0) continue;
                if (h[b[w]] ^ (h[v] + 1)) continue;
                int64_t res = pump(b[w], source, sink, min(cap
                     , c[w] - d[w]));
                ans += res;
                cap -= res;
                d[w] += res;
            }
            if (b[w] == v) {
                if (d[w] == 0) continue;
                if (h[a[w]] ^ (h[v] + 1)) continue;
                int64_t res = pump(a[w], source, sink, min(cap
                     , d[w]));
                ans += res;
                cap -= res;
                d[w] -= res;
            }
        }
        return ans;
    }
    int64_t solve(int source, int sink) {
        source--, sink--;
        int64_t ans = 0;
        d.resize(m), fill(d.begin(), d.end(), 0);
        mark.resize(n), fill(mark.begin(), mark.end(), 0);
        h.resize(n);
        pos.resize(n);
        while (int64_t pumped = pump(source, source, sink))
            ans += pumped;
        return ans;
    }
};

int n, m;

Dinic crap;

int32_t main() {
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    cin >> n >> m;
    for (int i = 1, a, b, c; i <= m; i++)
        cin >> a >> b >> c, crap.add_edge(a, b, c);
    cout << crap.solve(1, n);
    return (0);
}
```

2.3 Hopcroft

```
#include <bits/stdc++.h>

using namespace std;

#define endl '\n'

struct hopcroft{ //0 based
    int n , m; // size of each side
    int ans;
    vector < int > mu , mv; // u is matched with mu[u] and v
         with mv[v], -1 if not matched
    vector < vector < int > > adj;
    vector < int > layer;
    hopcroft(int n, int m):
        n(n) , m(m), ans(0),
        mu(n , -1) , mv(m , -1),
        adj(n) , layer(n){}
    void add_edge(int u, int v){
        adj[u].push_back(v);
        if(mu[u] == -1 and mv[v] == -1)
            ans ++ , mu[u] = v , mv[v] = u;
```

```
    }
    void bfs(){
        queue <int> q;
        for(int u = 0; u < n; u ++){
            if(mu[u] == -1) q.push(u), layer[u] = 0;
            else layer[u] = -1;
        }
        while(!q.empty()){
            int u = q.front(); q.pop();
            for(auto v: adj[u]) if(mv[v] != -1 and layer[mv[v
                 ]] == -1){
                layer[mv[v]] = layer[u] + 1;
                q.push(mv[v]);
            }
        }
    }
    bool dfs(int u){
        for(auto v: adj[u]) if(mv[v] == -1){
            mu[u] = v, mv[v] = u;
            return(1);
        }
        for(auto v: adj[u]) if(layer[mv[v]] == layer[u] + 1
             and dfs(mv[v])){
            mu[u] = v, mv[v] = u;
            return(1);
        }
        return(0);
    }
    int solve(){ // O( sqrt(V) * E )
        while(true){
            bfs();
            int augment = 0;
            for(int u = 0; u < n; u ++)
                if(mu[u] == -1)
                    augment += dfs(u);
            if(!augment)
                break;
            ans += augment;
        }
        return(ans);
    }
};

int l , r;
int m;

int32_t main(){
    ios::sync_with_stdio(false);cin.tie(0);
    cin >> l >> r >> m;
    hopcroft g(l , r);
    while(m -- ){
        int u , v;
        cin >> u >> v;
        g.add_edge(u , v);
    }
    cout << g.solve() << endl;
    for(int i = 0 ; i < l ; i ++){
        if(g.mu[i] != -1)
            cout << i << ' ' << g.mu[i] << endl;
    }
    return(0);
}
```

2.4 cen

```
void plant(int v , int par = 0){
 sz[v] = 1;
 for(auto u : adj[v]) if(u != par and !hide[u])
  plant(u , v) , sz[v] += sz[u];
}
int cen(int v , int n , int par = 0 , bool found = 0){
 while(!found){
  found = 1;
  for(auto u : adj[v]) if(u!=par and !hide[u] and sz[u] * 2
       > n){
   par = v , v = u , found = 0;
   break;
  }
 }
 return(v);
}
void add(int v , int par , int c){
 if(hide[v])return;
 for(auto u : adj[v])
  if(u!=par)
   add(u , v , c);
}
void rem(int v , int par , int c){
 if(hide[v])return;
 for(auto u : adj[v])
  if(u!=par)
   rem(u , v , c);
}
void calc(int v , int par){
 if(hide[v])return;
 for(auto u : adj[v])if(u!=par)
  calc(u , v);
}
void calc(int v){
 for(auto u : adj[v])
  add(u , v , a[v]);
 for(auto u : adj[v])
  rem(u , v , a[v]) , calc(u , v) , add(u , v , a[v]);
 for(auto u : adj[v])
  rem(u , v , a[v]);
}
void solve(int v){
 plant(v);
 v = cen(v , sz[v]);
 hide[v] = 1;
 calc(v);
 for(auto u : adj[v])
  if(!hide[u])
   solve(u);
}
```

2.5 dijkstra

```
void djk(int source){
    ms(dist, 63);
    dist[source] = 0;
    pq.push({-dist[source], source});
    while(pq.size()){
        auto [d, v] = pq.top();
        pq.pop();
        if(mark[v])continue;
        mark[v] = 1;
        for(auto [u, w] : adj[v]){
            if(dist[u] > dist[v] + w)
                dist[u] = dist[v] + w, pq.push({-dist[u], u});
        }
    }
}
```

2.6 hld

```
void dfs_sz(int v = 0) {
    sz[v] = 1;
    for(auto &u: g[v]) {
        dfs_sz(u);
        sz[v] += sz[u];
        if(sz[u] > sz[g[v][0]]) {
            swap(u, g[v][0]);
        }
    }
}
```

```
void dfs_hld(int v = 0) {
    if(!head[v])head[v] = v;
    if(g[v].size())
        head[g[v][0]] = head[v];
    in[v] = ++t;
    for(auto u: g[v]) {
        dfs_hld(u);
    }
    out[v] = t+1;
}
```

2.7 scc

```
int n , m , cnt = 1 ;
vector < int > adj[maxn] , radj[maxn] , order;
int mark[maxn] , c[maxn];

void sfd(int v){
    c[v] = cur;
    for (auto u : radj[v])
        if(!c[u])
            sfd(u);
}

void dfs(int v){
    mark[v] = 1;
    for (auto u : adj[v])
        if(!mark[u])
            dfs(u);
    order.pb(v);
}

int32_t main(){
    for (int i = 1 ; i <= n ; i ++)
        if(!mark[i])
            dfs(i);
    reverse(order.begin() , order.end());
    for (int i = 0 ; i < n ; i ++)
        if(!c[order[i]])
            ++cnt, sfd(order[i]);
    return(0);
}
```

2.8 tur

```
int pointer[MAXN];
vector<pii> adj[MAXN];
bool mark[MAXN];

void tour(int v){
    while(pointer[v] < (int)adj[v].size()){
        if(!mark[adj[v][pointer[v]].S]){
            mark[adj[v][pointer[v]].S] = 1;
            tour(adj[v][pointer[v]].F);
        }
        pointer[v]++;
    }
    ans.push_back(v);
}
```

3 Misc

3.1 LIS

```
int LIS(vector <int> &vec){
    multiset <int> st;
    for(int x : vec){
        st.insert(x);
        auto it = st.lower_bound(x);
        it++;
        if(it != st.end())
            st.erase(it);
    }
    return st.size();
}
```

3.2 dp$_{d}ivide$

```
int m, n;
vector<long long> dp_before(n), dp_cur(n);

long long C(int i, int j);

// compute dp_cur[l], ... dp_cur[r] (inclusive)
void compute(int l, int r, int optl, int optr) {
    if (l > r)
        return;

    int mid = (l + r) >> 1;
    pair<long long, int> best = {LLONG_MAX, -1};

    for (int k = optl; k <= min(mid, optr); k++) {
        best = min(best, {(k ? dp_before[k - 1] : 0) + C(k,
            mid), k});
    }

    dp_cur[mid] = best.first;
    int opt = best.second;

    compute(l, mid - 1, optl, opt);
    compute(mid + 1, r, opt, optr);
}

int solve() {
    for (int i = 0; i < n; i++)
        dp_before[i] = C(0, i);

    for (int i = 1; i < m; i++) {
        compute(0, n - 1, 0, n - 1);
        dp_before = dp_cur;
    }

    return dp_before[n - 1];
}
```

3.3 mat

```
struct Mat{
    int m[K][K];
    Mat(int diag = -1){
        ms(m , 0);
        if(diag==0)for(int i = 0 ; i < K ; i ++)m[i][i] = 1;
        if(diag>0)for(int i = 0 ; i < diag ; i ++)m[i][i
            +1]=1;
    }
    Mat operator* (const Mat &b) const{
        Mat c = Mat();
        for(int i = 0 ; i < K ; i ++)
            for(int k = 0 ; k < K ; k ++)
                for(int j = 0 ; j < K ; j ++)
                    c.m[i][j] = (ll(c.m[i][j]) + ll(m[i][k]) *
                        b.m[k][j])%mod;
        return(c);
    }
};
Mat pw(Mat a, ll b){Mat res(0);while(b){if(b&1){res=(a*res)
    ;}a=(a*a);b>>=1;}return(res);}
```

3.4 phi

```
int phi(int n) {
    int result = n;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            while (n % i == 0)
                n /= i;
            result -= result / i;
        }
    }
    if (n > 1)
        result -= result / n;
    return result;
}

void phi_1_to_n(int n) {
    vector<int> phi(n + 1);
    for (int i = 0; i <= n; i++)
        phi[i] = i;

    for (int i = 2; i <= n; i++) {
        if (phi[i] == i) {
            for (int j = i; j <= n; j += i)
                phi[j] -= phi[j] / i;
        }
    }
}

struct Congruence {
    long long a, m;
};

long long chinese_remainder_theorem(vector<Congruence> const
     & congruences) {
    long long M = 1;
    for (auto const& congruence : congruences) {
        M *= congruence.m;
    }

    long long solution = 0;
    for (auto const& congruence : congruences) {
        long long a_i = congruence.a;
        long long M_i = M / congruence.m;
        long long N_i = mod_inv(M_i, congruence.m);
        solution = (solution + a_i * M_i % M * N_i) % M;
    }
    return solution;
}
```

```
for (int s=m; ; s=(s-1)&m) {
 ... you can use s ...
 if (s==0) break;
}
```

3.5 time

```
chrono::steady_clock::time_point begin = chrono::
    steady_clock::now();
chrono::steady_clock::time_point end = chrono::steady_clock
    ::now();
chrono::duration_cast<std::chrono::milliseconds>(end - begin
    ).count();

auto start_time = chrono::high_resolution_clock::now();
auto end_time = chrono::high_resolution_clock::now();
auto elapsed_time = chrono::duration_cast<chrono::
    milliseconds>(end_time - start_time);
std::cout << "Elapsed time: " << elapsed_time.count() << "
    ms\n";
```

4 Strings

4.1 Aho

```
#define SIGMA 26

int nxt[SIGMA][MAXN] , f[MAXN] , ext[MAXN] , sz = 0;
bool endpoint[MAXN];

int add(string &s){
 int cur = 0;
 for(char c : s){
  if(!nxt[c - 'a'][cur])nxt[c - 'a'][cur] = ++sz;
  cur = nxt[c - 'a'][cur];
 }
 endpoint[cur] = 1;
 return cur;
}

void build(){//q is a queue
 for(int i = 0 ; i < SIGMA ; i ++)if(nxt[i][0])q.push(nxt[i
     ][0]);
 while(q.size()){
  int v = q.front();
  q.pop();
  if(endpoint[f[v]])ext[v] = f[v];
  else ext[v] = ext[f[v]];
  for(int i = 0 ; i < SIGMA ; i ++)
   if(nxt[i][v])f[nxt[i][v]] = nxt[i][f[v]] , q.push(nxt[i][
       v]);
   else nxt[i][v] = nxt[i][f[v]];
 }
}
```

4.2 Strtable

```
struct strtable{
 #define MAXN 500010
 #define LG 20

 int rnk[LG][MAXN], n;
 int tmp[MAXN];
 int LST[MAXN], NXT[MAXN];
 int lg[MAXN];
 void build(string s){
  n = s.size();
  for(int i = 0 ; i < n ; i ++)
   tmp[i] = s[i] - 'a';
  sort(tmp , tmp + n);
  int sz = unique(tmp , tmp + n) - tmp;
  for(int i = 0 ; i < sz ; i ++)
   LST[tmp[i]] = i;
  for(int i = 0 ; i < n ; i ++)
   rnk[0][i] = LST[s[i] - 'a'];
  for(int j = 1 ; (1 << j) - 1 < n ; j ++){
   for(int i = 0 ; i + (1 << (j-1)) - 1 < n ; i ++)LST[i] =
       -1;
   for(int i = n - (1 << j) ; ~i ; i --)
    NXT[i] = LST[rnk[j - 1][i + (1 << (j - 1))]] , LST[rnk[j
        - 1][i + (1 << (j - 1))]] = i;
   int pos = 0;
   for(int i = 0 ; i + (1 << (j-1)) - 1 < n ; i ++)
    for(int k = LST[i] ; ~k ; k = NXT[k])
     tmp[pos++] = k;
   for(int i = 0 ; i + (1 << (j-1)) - 1 < n ; i ++)LST[i] =
       -1;
   for(int i = n - (1 << j) ; ~i ; i --)
    NXT[i] = LST[rnk[j - 1][tmp[i]]] , LST[rnk[j - 1][tmp[i
        ]]] = i;
   pos = 0;
   for(int i = 0 ; i + (1 << (j-1)) - 1 < n ; pos += (LST[i]
        > -1) , i ++)
    for(int k = LST[i] ; ~k ; k = NXT[k])
     rnk[j][tmp[k]] = pos ,
```

```
    pos = ((~NXT[k]) ? ((rnk[j - 1][tmp[k] + (1 << (j - 1))]
          ^ rnk[j - 1][tmp[NXT[k]] + (1 << (j - 1))]) ? pos
        + 1 : pos) : pos);
 }
 for(int i = 2 ; i <= n ; i ++)
  lg[i] = lg[i >> 1] + 1;
}
pair < int , int > get(int l , int r){
 return pair < int , int > (rnk[lg[r - l]][l] , rnk[lg[r -
     l]][r - (1 << lg[r - l]) + 1]);
}
bool cmp(int l, int r, int L, int R){
 int sz = min(r - l , R - L);
 if(get(l , l + sz) == get(L , L + sz))
  return (r - l) < (R - L);
 return get(l , l + sz) < get(L , L + sz);
}
int Lcp(int l , int r , int L , int R){
 int ans = 0;
 for(int i = 0 ; i < n ; i ++){
  for(int j = LG ; ~j ; j --)if(l + (1 << j) -1 <= r and L
       + (1 << j) - 1 <= R){
   if(rnk[j][l] == rnk[j][L]){
    ans += (1 << j);
    l += (1 << j);
    L += (1 << j);
   }
  }
 }
 return ans;
}
};

int sa[MAXN];
strtable *st;
bool SAcmp(int i , int j){
 return st->cmp(i , st->n - 1 , j , st->n - 1);
}
void SA(strtable *ST){
 st = ST;
 for(int i = 0 ; i < st->n ; i ++) sa[i] = i;
 sort(sa , sa + st->n , SAcmp);
}

int lcp[MAXN];
void LCP(strtable *st){
 for(int i = 1 ; i < st->n ; i ++){
  int u = sa[i - 1], v = sa[i];
  for(int j = LG ; ~j ; j --)if(u + (1 << j) - 1 < st->n and
       v + (1 << j) - 1 < st->n){
   if(st->rnk[j][u] == st->rnk[j][v]){
    lcp[i] += (1 << j);
    u += (1 << j);
    v += (1 << j);
   }
  }
 }
}
```

4.3 SuffixArray

```
int sa[maxn];
int rk[maxn];
int tp[maxn];
int cnt[maxn];
int lcp[maxn];

void SA(string &s){
 int A = 'z' , p = 0 , n = s.size();
 if(n == 1){
  sa[0] = rk[0] = 0;
  return;
 }
 for(int i = 0 ; i < n ; i ++)
  sa[i] = i , rk[i] = s[i];
 for(int j = 1 ; p < n - 1 ; j<<=1){
  p = 0;
  int k = (j>>1);
  for(int i = n - k ; i < n ; i ++)
   tp[p ++ ] = i;
  for(int i = 0 ; i < n ; i ++)
   if(sa[i] >= k)
    tp[p ++] = sa[i] - k;
  for(int i = 0 ; i <= A ; i ++)
   cnt[i] = 0;
  for(int i = 0 ; i < n ; i ++)
   cnt[rk[i]] ++;
  for(int i = 1 ; i <= A ; i ++)
   cnt[i] += cnt[i - 1];
  for(int i = n - 1 ; i >= 0 ; i --)
   sa[--cnt[rk[tp[i]]]] = tp[i];
  swap(rk , tp);
  rk[sa[0]] = p = 0;
  for(int i = 1 ; i < n ; i ++)
   p += (tp[sa[i - 1]]!=tp[sa[i]] || sa[i - 1] + k >= n ||
    tp[sa[i - 1]+k] != tp[sa[i] + k]) , rk[sa[i]] = p;
  A = p;
 }
}
```

4.4 Zalgo

```
int zlcp[MAXN]; //zlcp[i] == lcp(s[i , .... , n-1] , s[0 ,
     ... , n-1])
void ZAlgo(strtable *st){
 for(int i = 0 ; i < st-> n ; i ++){
  int u = i , v = 0;
  for(int j = LG ; ~j ; j --)if(u + (1 << j) -1 < st->n and
       v + (1 << j) - 1 < st->n){
   if(st->rnk[j][u] == st->rnk[j][v]){
    zlcp[i] += (1 << j);
    u += (1 << j);
    v += (1 << j);
   }
  }
 }
}
```

4.5 kmp

```
//1 indexed
vector < int > kmp(string s){
    int i = -1 ;
    vector < int > f(s.size() + 1);
    f[0] = -1;
    for (int j = 0 ; j < s.size() ; j ++){
        while(s[j]!=s[i] and i!=-1)
            i = f[i];
        f[j + 1] = ++i;
    }
    return(f);
}
//0 indexed
vector < int > kmp(string s){
 int n = s.size();
    vector < int > f(n);
    for(int i = 1; i < n ; i ++){
```

```
  int j = f[i-1];
  while(j and s[i]!=s[j])
   j = f[j-1];
  j+=(s[i]==s[j]) , f[i] = j;
 }
   return(f);
}
```

5 Templates

5.1 generator

5.1.1 tester

```
#!/bin/bash
echo "" > main.txt
echo "" > naive.txt
g++ -std=c++17 -o main main.cpp
g++ -std=c++17 -o naive naive.cpp
g++ -std=c++17 -o gen gen.cpp
((i = 1))
while diff main.txt naive.txt -Bb
do
  echo $i
  ((i++))
  ./gen > test.txt
  ./main < test.txt > main.txt
  ./naive < test.txt > naive.txt
done
```

5.2 temp

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef long double ld;
typedef pair<int , int> pii;

mt19937 rng(chrono::steady_clock::now().time_since_epoch().
    count());

const int maxn = 3e6;
const ll mod = 1e9+7;

#define pb push_back
#define endl '\n'
#define dokme(x) cout << x , exit(0)
#define ms(x , y) memset(x , y , sizeof x)
ll pw(ll a, ll b, ll md = mod){ll res = 1;while(b){if(b&1){
    res=(a*res)%md;}a=(a*a)%md;b>>=1;}return(res);}



int32_t main(){
 cin.tie(0)->sync_with_stdio(0);

 return(0);
}
```

5.3 vimrc

```
filetype detect
set nocompatible
set exrc
set mouse=a
set tabstop=4
set shiftwidth=4

set autoindent
set smartindent
set cindent

set showcmd
set number
set autowrite
set autoread

set nowrap
colorscheme elflord

set keymodel=startsel,stopsel

map <F9> :<C-U>!g++ -O2 -std=c++11 -Wall -Wextra %:r.cpp -o
    %:r<CR>
map <F5> :<C-U>!./%:r<CR>
```