

maxflow

```
struct MaxFlow{
    int n;
    vector<vector<int>>>adj;
    vector<vector<ll>>>cp;

    MaxFlow(int n):n(n){
        adj=vector<vector<int>>>(n);
        cp=vector(n,vector(n,0LL));
    }

    // connect a and b with capacity c
    void add(int a,int b,int c){
        if(cp[a][b]==0){
            adj[a].push_back(b);
            adj[b].push_back(a);
        }
        cp[a][b]+=c;
    }

    ll bfs(int s,int t,vector<int>&p){
        fill(p.begin(),p.end(),-1);
        p[s]=-2;
        queue<pair<int,int>>q;
        q.push({s,1e9});
        while(!q.empty()){
            int c=q.front().first;
            ll f=q.front().second;
            q.pop();
            for(int x:adj[c]){
                if(p[x]==-1&&cp[c][x]){
                    p[x]=c;
                    ll r=min(f,cp[c][x]);
                    if(x==t)
                        return r;
                    q.push({x,r});
                }
            }
        }

        return 0;
    }

    // call only once
    ll maxflow(int s,int t){
        ll f=0,r;
        vector<int>pr(n);
        while((r=bfs(s,t,pr))){
            f+=r;
            int cu=t;
            while(cu!=s){
                int p=pr[cu];
                cp[p][cu]-=r;
                cp[cu][p]+=r;
                cu=p;
            }
        }
        return f;
    }
};
```

euclid

```
// ax+by=gcd
int gcd(int a,int b,int&x,int&y){
    x=1;y=0;
    int x1=0,y1=1,a1=a,b1=b;
    while(b1){
        int q=a1/b1;
        tie(x,x1)=make_tuple(x1,x-q*x1);
        tie(y,y1)=make_tuple(y1,y-q*y1);
        tie(a1,b1)=make_tuple(b1,a1-q*b1);
    }
    return a1;
}
```

z_algorithm

```
//      z[i] pove, da se z[i] znakovl zacensi z i-tim ujemajo s prvimi z[i] znaki stringa s.
vector<int> z_function(string s) {
    int n = s.size();
    vector<int> z(n);
    int l = 0, r = 0;
    for(int i = 1; i < n; i++) {
        if(i < r) {
            z[i] = min(r - i, z[i - l]);
        }
        while(i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if(i + z[i] > r) {
            l = i;
            r = i + z[i];
        }
    }
    return z;
}
```

modulo_int

```
const int PC=1e7;
typedef long long ll;
const int MOD=1e9+7;
struct M{
    ll val=0;
    static ll inv[PC+1];
    static ll calc_inv(ll i){if(i<=PC)return inv[i];return(MOD-MOD/i)*calc_inv(MOD%i)%MOD;}
    static void precomp(){inv[1]=1;for(ll i=2;i<=PC;++i){inv[i]=(MOD-MOD/i)*inv[MOD%i]%MOD;if(inv[i]<0)inv[i]+=MOD;}}
    M(ll x){val=(x%MOD+MOD)%MOD;}
    M()=default;
    template<class T>M operator+(T bt){M b(bt);M r(val);r.val+=b.val;if(r.val>=MOD)r.val-=MOD;return r;}
    template<class T>M operator-(T bt){M b(bt);M r(val);r.val+=MOD-b.val;if(r.val>=MOD)r.val-=MOD;return r;}
    template<class T>M operator*(T bt){M b(bt);M r(val);r.val*=b.val;r.val%=MOD;return r;}
    template<class T>M&operator+=(T bt){M b(bt);*this=*this+b;return*this;}
    template<class T>M&operator-=(T bt){M b(bt);*this=*this-b;return*this;}
    template<class T>M&operator*=(T bt){M b(bt);*this=*this*b;return*this;}
    M pow(ll e){M r(1);M a(val);while(e){if(e%2)r=r*a;a=a*a/e/=2;return r;}
    template<class T>M operator/(T bt){ M b(bt); return M(val*M::calc_inv(b.val)); }
    template<class T>M&operator/=(T bt){M b(bt);*this*=M::calc_inv(b.val);return*this;}
};
ostream&operator<<(ostream&s,M m){s<<m.val;return s;}
istream&operator>>(istream&s,M&m){s>>m.val;return s;}

// Add `M::precomp();` to the start of main
```

segtree

```
template<typename T>
struct SegTree{
    int n;
    vector<T>t;
    T (*m)(T,T);

    // n = size, d = default value, m = merge function
    SegTree(int n,T d,T(*m)(T,T)):n(n),m(m){
        t=vector<T>(2*n,d);
        for(int i=n-1;i>0;i--){
            t[i]=m(t[2*i],t[2*i+1]);
        }

        // point update
        void upd(int i,T x){
            i+=n;
            t[i]=x;
            while(i>1){
                i/=2;
                t[i]=m(t[2*i],t[2*i+1]);
            }
        }

        // range query
        T get(int l,int r){
            l+=n;
            r+=n;
            T rs=t[l];
            l++;
            while(l<r){
                if(l%2){
                    rs=m(rs,t[l]);
                    l++;
                }
                if(r%2){
                    r--;
                    rs=m(rs,t[r]);
                }
                l/=2;
                r/=2;
            }
            return rs;
        }
    };
};
```

suffix_array

```
vector<int> sort_cyclic_shifts(string const& s) {
    int n = s.size();
    const int alphabet = 256;
    vector<int> p(n), c(n), cnt(max(alphabet, n), 0);
    for (int i = 0; i < n; i++)
        cnt[s[i]]++;
    for (int i = 1; i < alphabet; i++)
        cnt[i] += cnt[i-1];
    for (int i = 0; i < n; i++)
        p[--cnt[s[i]]] = i;
    c[p[0]] = 0;
    int classes = 1;
    for (int i = 1; i < n; i++) {
        if (s[p[i]] != s[p[i-1]])
            classes++;
        c[p[i]] = classes - 1;
    }
    vector<int> pn(n), cn(n);
    for (int h = 0; (1 << h) < n; ++h) {
        for (int i = 0; i < n; i++) {
            pn[i] = p[i] - (1 << h);
            if (pn[i] < 0)
                pn[i] += n;
        }
        fill(cnt.begin(), cnt.begin() + classes, 0);
        for (int i = 0; i < n; i++)
            cnt[c[pn[i]]]++;
        for (int i = 1; i < classes; i++)
            cnt[i] += cnt[i-1];
        for (int i = n-1; i >= 0; i--)
            p[--cnt[c[pn[i]]]] = pn[i];
        cn[p[0]] = 0;
        classes = 1;
        for (int i = 1; i < n; i++) {
            pair<int, int> cur = {c[p[i]], c[(p[i] + (1 << h)) % n]};
            pair<int, int> prev = {c[p[i-1]], c[(p[i-1] + (1 << h)) % n]};
            if (cur != prev)
                ++classes;
            cn[p[i]] = classes - 1;
        }
        c.swap(cn);
    }
    return p;
}

vector<int> suffix_array_construction(string s) {
    s += "$";
    vector<int> sorted_shifts = sort_cyclic_shifts(s);
    sorted_shifts.erase(sorted_shifts.begin());
    return sorted_shifts;
}

int compare(int i, int j, int l, int k) {
    pair<int, int> a = {c[k][i], c[k][(i+l-(1 << k))%n]};
    pair<int, int> b = {c[k][j], c[k][(j+l-(1 << k))%n]};
    return a == b ? 0 : a < b ? -1 : 1;
}

int lcp(int i, int j) {
    int ans = 0;
    for (int k = log_n; k >= 0; k--) {
        if (c[k][i % n] == c[k][j % n]) {
            ans += 1 << k;
            i += 1 << k;
            j += 1 << k;
        }
    }
    return ans;
}

vector<int> lcp_construction(string const& s, vector<int> const& p) {
    int n = s.size();
    vector<int> rank(n, 0);
    for (int i = 0; i < n; i++)
        rank[p[i]] = i;

    int k = 0;
    vector<int> lcp(n-1, 0);
    for (int i = 0; i < n; i++) {
        if (rank[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = p[rank[i] + 1];
        while (i + k < n && j + k < n && s[i+k] == s[j+k])
            k++;
        lcp[rank[i]] = k;
        if (k)
            k--;
    }
    return lcp;
}
```

