Coding-Library

August 14, 2023

Contents

1. Graph

1..1 0-1 BFS

```
vector<int> d(n, INF);
d[s] = 0;
deque<int> q;
q.push_front(s);
while (!q.empty()) {
    int v = q.front();
    q.pop_front();
    for (auto edge : adj[v]) {
        int u = edge.first;
        int w = edge.second;
        if (d[v] + w < d[u]) {
            d[u] = d[v] + w;
            if (w == 1)
                q.push_back(u);
            else
                q.push_front(u);
        }
    }
}
```

1..2 DSU

```
struct dsu {
    vt<int> par, sz;
    explicit dsu(int n)
    {
        par.assign(n+1, 0);
        iota(all(par), 0);
        sz.assign(n+1, 1);

    }
    int get_par(int x) {
        if (par[x] == x) return x;
        return par[x] = get_par(par[x]);
    }
    bool join(int a, int b) {
        a = get_par(a), b = get_par(b);
        if (a == b) return false;
        if (sz[a] > sz[b]) swap(a, b);
        par[a] = par[b];
        sz[b] += sz[a];
        return true;
    }
};
```

1..3 Dijkstra

```
vector<int> dist; // answer -> dist[destination]
vector<vector<pair<int,int>>> g; // {cost, to}

void djikstra(int src = 1) {
    dist.resize(g.size(), INFINITY);
    priority_queue<pair<int,int>, vector<pair<int,int>>,
         greater<pair<int,int>>> pq;
    pq.push({dist[src] = 0, src});
    while(!pq.empty()) {
        auto curr = pq.top(); pq.pop();
        for(auto to : g[curr.second])
            if(dist[to.second] > curr.first + to.first)
                pq.push({dist[to.second] = curr.first + to.
                     first, to.second});
    }
}
```

2. Number theory

2..1 Euler's totient

```
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;
        }
    }
}
```

2..2 Fast Power

```
ll power(ll b, ll p, ll mod) {
    ll res = 1;
    b = b % mod;
    if (b == 0) return 0;
    while (p)
    {
        if (p & 1)
            res = (res * b) % mod;
        p >>= 1;
        b = (b * b) % mod;
    }
    return res;
}
```

2..3 Sieve

```
int n;
vector<bool> is_prime(n+1, true);
is_prime[0] = is_prime[1] = false;
for (int i = 2; i * i <= n; i++) {
    if (is_prime[i]) {
        for (int j = i * i; j <= n; j += i)
            is_prime[j] = false;
    }
}
```

2..4 nCr with Mod Inverse

```
ll modInverse(ll n, ll mod)
{
    return power(n, mod - 2, mod);
}

ll nCr(ll n, ll r, ll mod)
{
    if (n < r)
        return 0;
    if (r == 0)
        return 1;
    ll fac[n + 1];
    fac[0] = 1;
    for (int i = 1; i <= n; i++)
        fac[i] = (fac[i - 1] * i) % mod;
    return (fac[n] * modInverse(fac[r], mod) % mod
            * modInverse(fac[n - r], mod) % mod)
        % mod;
}
```

2..5 nPr

```
int nPr(int n, int r)
{
 ll ans = 1;
 for (int i = 0; i < (n - r); i++)
  ans *= (n - i);
}
```