# Team notebook

Jala-Peños-i

April 1, 2022

## Contents



# 1 dp

# 2 graph

## 2.1 1 - DFS

```
const int n = 1e6;
vector<int> adj[n + 1];
bool visited[n + 1];

void dfs(int x) {
	if (visited[x]) return;
	visited[x] = true;
	for (int &a : adj[x]) {
		dfs(x);
	}
}
```

## 2.2 2 - BFS

```
2. BFS

vector<int> adj[n + 1];
bool visited[n + 1];
```

```
void bfs() {
	queue<int> q;
	q.push(0); // initial node
	visited[0] = true;
	while(q.size() > 0) {
		int c = q.front();
		q.pop();
		for (int a : adj[c]) {
			if (visited[a]) continue;
			q.push(a);
			visited[a] = true;
		}
	}
}
```

## 2.3 3 - Dijkstra

```
3. Dijkstra

const int inf = 1e9;
vector<pair<int, int>> adj[n];
bool processed[n];
ll distance[n];

void dijkstra() {
	priority_queue<pair<int, int>> q;
	for (int i = 0; i < n; i++) {
		distance[i] = inf;
	}
	distance[start] = 0;
	q.push({0, start});
	while (q.size() > 0) {
		int c = q.top().second;
		q.pop();
		if (processed[c]) continue;
		processed[c] = true;
		for (auto& a : adj[c]) {
			int u = a.first;
			int w = a.second;
			if (distance[c] + w < distance[u]) {
				distance[u] = distance[c] + w;
				q.push({-distance[u], u});
			}
		}
	}
}
```

## 2.4 4 - BellmanFord

```
4. BellmanFord

const int inf = 1e9;
vector<tuple<int, int, int>> edges;
ll distance[n];


void bellmanFord() {
	for (int i = 0; i < n; i++) {
		distance[i] = inf;
	}
	distance[start] = 0;
	for (int i = 0; i < n - 1; i++) {
		//bool changed = false; add one iteration (i < n) to valide
		    negative cicles
		for (auto& edge : edges) {
			int a, b, w;
			tie(a, b, w) = edge;
			if (distance[a] + w < distance[b]) {
				distance[b] = distance[a] + w;
				//changed = true;
			}
		}
	}
}
```

## 2.5 5 - Floyd Warshall

```
5. Floyd Warshall

const int inf = 1e9;
vector<pair<int, int>> adj[n];
```

```
ll distance[n][n];

void floydWarshall() {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			distance[i][j] = inf;
		}
	}
	for (int i = 0; i < n; i++) {
		for (auto p : adj[i]) {
			int b = p.first;
			int w = p.second;
			distance[i][b] = w;
		}
	}
	for (int k = 0; k < n; k++) {
		for (int i = 0; i < n; i++) {
			for (int j = 0; j < n; j++) {
				distance[i][j] = min(distance[i][j],
				    distance[i][k] + distance[k][j]);
			}
		}
	}
}
```

## 2.6 6 - Euler Path and Cycle

```
6. Euler Path and Cycle

// TODO
```

## 2.7 7 - Topological Sort

```
7. Topological Sort

stack<int> topo;
vector<int> adj[n + 1];
bool visited[n + 1];

void dfs(int x) {
	if (visited[x]) return;
	visited[x] = true;
	for (int a : adj[x]) {
		dfs(a);
	}
	topo.push(x);
}
```

## 2.8 8 - Transitive Closure

```
8. Transitive Closure

const int inf = 1e9;
vector<int> adj[n];
ll distance[n][n];

void floydWarshall() {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			distance[i][j] = false;
		}
	}
	for (int i = 0; i < n; i++) {
		for (int b : adj[i]) {
			distance[i][b] = true;
		}
	}
	for (int k = 0; k < n; k++) {
		for (int i = 0; i < n; i++) {
			for (int j = 0; j < n; j++) {
				distance[i][j] |= distance[i][k] & distance[k][j];
			}
		}
	}
}
```

## 2.9 9 - Kruskal

```
9. Kruskal
```

```
vector<tuple<int, int, int> edges;


void kruskal() {
	ll res = 0;
	sort(edges.begin(), edges.end());// sorted by weight
	for (auto edge : edges) {
		int a, b, w;
		tie(w, a, b) = edge;
		if (find(a) != find(b)) {
			group(a, b);
			res += w;
		}
	}
}
```

## 2.10 A - Union Find

```
10. Union Find

int link[n];
int score[n];

void find(int a) {
	if (link[a] == a) return a;
	return link[a] = find(link[a]);
}

void group(int a, int b) {
	int pa = find(a);
	int pb = find(b);
	if (pa != pb) {
		if (score[pa] > score[pb]) {
			link[pb] = pa;
		} else if (score[pa] < score[pb]) {
			link[pa] = pb;
		} else {
			score[pa]++;
			link[pb] = pa;
		}
	}
}

void init() {
	for (int i = 0; i < n; i++) {
		link[i] = i;
		score[i] = 0;
	}
}
```

## 2.11 B - SCC

```
B - SCC

/// Complexity: O(|N|)
/// Tested: https://tinyurl.com/y8ujj3ws
int scc(int n) {
  vector<int> dfn(n+1), low(n+1), in_stack(n+1);
  stack<int> st;
  int tag = 0;
  function<void(int, int&)> dfs = [&](int u, int &t) {
    dfn[u] = low[u] = ++t;
    st.push(u);
    in_stack[u] = true;
    for(auto &v : g[u]) {
      if(!dfn[v]) {
        dfs(v, t);
        low[u] = min(low[u], low[v]);
      } else if(in_stack[v])
        low[u] = min(low[u], dfn[v]);
    }
    if (low[u] == dfn[u]) {
      int v;
      do {
        v = st.top(); st.pop();
//        id[v] = tag;
        in_stack[v] = false;
      } while (v != u);
      tag++;
    }
  };
  for(int u = 1, t; u <= n; ++u) {
    if(!dfn[u]) dfs(u, t = 0);
  }
```

```
  return tag;
}
```

# 3 math

## 3.1 Extended Euclides

```
// It finds X and Y in equation:
// a * X + b * Y = gcd(a, b)

int x, y;

int euclid(int a, int b) {
	if (b == 0) {
		x = 1;
		y = 0;
		return a;
	}
	int aux = x;
	x = y;
	y = aux - a/b*y;
	return euclid(b, a % b);
}
```

## 3.2 Greatest Common Divisor

```
// Alternative: __gcd(a, b);
// O(log(max(a, b)))

ll gcd(ll a, ll b) {
	return b == 0 ? a : gcd(b, a % b);
}
```

## 3.3 Lowest Common Multiple

```
// O(log(max(a, b)))
int lcm(int a, int b) {
	return a/gcd(a, b) * b;
}
```

## 3.4 Modular Aritmethics

Modular Aritmethics.cpp

```
ll sum(ll a, ll b) {
    ll c = a + b;
    if (c >= m) c -=m;
    return c;
}

ll sub(ll a, ll b) {
    ll c = a - b;
    if (c < 0) c += m;
    return c;
}

ll mul(__int128 a, __int128 b) {
    return (a * b) % m;
}

ll modexp(ll a, ll n) {
    if (n == 0) return 1;
    ll p = modexp(a, n / 2);
    ll res = mul(p, p);
    if (n & 1) {
        res = mul(res, a);
    }
    return res;
}

// O(sqrt n)
ll phi(ll n) {
    ll ans = n;
    for (int p = 2; p <= n/p; ++p) {
        if (n % p == 0) ans -= ans / p;
        while (n % p == 0) n /= p;
    }
    if (n > 1) ans -= ans / n;
```

```
    return ans;
}

ll x, y;
/// O(log(max(a, b)))
ll euclid(ll a, ll b) {
    if(b == 0) { x = 1; y = 0; return a; }
    ll d = euclid(b, a%b);
    ll aux = x;
    x = y;
    y = aux - a/b*y;
    return d;
}

ll invmod(ll a) {
    ll d = euclid(a, m);
    if (d > 1) return -1;
    return (x % m + m) % m;
}

ll divv(ll a, ll b) {
    ll inv = invmod(b);
    if (inv == -1) return -1;
    ll res = mul(a, inv);
    return res;
}


// a * (b^{euler(m) - 1})
// for primes: a * b ^ (P - 2)
ll divv2(ll a, ll b) {
    if (__gcd(b, m) != 1) return -1;
    ll ex = modexp(b, euler - 1);
    ll res = mul(a, ex);
    return res;
}
```

## 3.5 primes

```
// O(sqrt(n))
bool isPrime(int x) {
    for (int d = 2; d * d <= x; d++) {
        if (x % d == 0)
            return false;
    }
    return true;
}

// O(nloglogn)
// sieve[X] == 0 if it is prime
int const N = 1e6;
bool sieve[N + 1];
vector<int> primes;

void calculate() {
    for (int p = 2; p <= N; p++) {
        if (sieve[p]) continue;
        primes.PB(p);
        for (ll i = 1ll*p*p; i <= N; i += p)
            sieve[i] = true;
    }
}


// For 64-bit integers
// O((ln n)^2)
// 32 bits bases: 2, 3, 5, 7.
// 64 bits bases: 2 ... 37

using u64 = uint64_t;
using u128 = __uint128_t;

u64 binpower(u64 base, u64 e, u64 mod) {
    u64 result = 1;
    base %= mod;
    while (e) {
        if (e & 1)
            result = (u128)result * base % mod;
        base = (u128)base * base % mod;
        e >>= 1;
    }
    return result;
}

bool check_composite(u64 n, u64 a, u64 d, int s) {
    u64 x = binpower(a, d, n);
```

```
    if (x == 1 || x == n - 1)
        return false;
    for (int r = 1; r < s; r++) {
        x = (u128)x * x % n;
        if (x == n - 1)
            return false;
    }
    return true;
}

bool MillerRabin(u64 n) {
    if (n < 2)
        return false;

    int r = 0;
    u64 d = n - 1;
    while ((d & 1) == 0) {
        d >>= 1;
        r++;
    }

    for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}) {
        if (n == a)
            return true;
        if (check_composite(n, a, d, r))
            return false;
    }
    return true;
}
```

# 4 query

## 4.1 1 - Segment Tree

1. Segment Tree

```
const int N = 1e6 + 1;

int tree[N * 4 + 4];
int nums[N + 1];

void build(int i, int l, int r) {
        if (l == r) {
                tree[i] = nums[r];
        } else {
                int mid = (l + r) / 2;
                build(i * 2 + 1, l, mid);
                build(i * 2 + 2, mid + 1, r);
                tree[i] = tree[i * 2 + 1] + tree[i * 2 + 2];
                // tree[i] = compare(tree[i * 2 + 1], tree[i * 2 + 2]);
        }
}

void update(int i, int l, int r, int pos, int diff) {
        if (l <= pos && pos <= r) {
                if (l == r) { // leaf
                        tree[i] += diff;
                } else { // node
                        int mid = (l + r) / 2;
                        update(i * 2 + 1, l, mid, pos, diff);
                        updaet(i * 2 + 2, mid + 1, r, pos, diff);
                        tree[i] = tree[i * 2 + 1] + tree[i * 2 + 2];
                        // tree[i] = compare(...)
                }
        }
}

int query(int i, int sl, int sr, int l, int r) {
        if (l <= sl && sr <= r) { // overlap
                return tree[i];
        } else if (sr < l || r < sl) { // no overlap
                return 0;
        } else { // partially over lap
                int mid = (sl + sr) / 2;
                return query(i * 2 + 1, sl, mid, l, r) + query(i * 2 + 2, mid +
                    1, sr, l, r);
                // return compare(a, b);
        }
}
```

# 5 string

## 5.1 1 - KMP

1. KMP.cpp

```cpp
string t;
string p;
int lps[p.size()]; // set it with 0's

void init() {
	int j = 0;
	for (int i = 0; i < p.size(); i++) {
		while (j && p[i] != p[j]) j = lps[j - 1];
		if (p[i] == p[j]) j++;
		lps[i] = j;
	}
}

void kmp() {
	int n = t.size();
	int m = p.size();
	int j = 0;
	for (int i = 0; i < n; i++) {
		while (j & p[j] != s[i]) j = lps[j - 1];
		if (p[j] == s[i]) j++;
		if (j == m) {
			//process
			j = pf[j - 1];
		}
	}
}
```

## 5.2 2 Rabin Karp

2. Rabin Karp.java

```java
public class C {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
    String t;
    String p;
    long prime = 257;
    long mod = (int)1e9 + 7;// mod is prime and mod * mod < LONG_MAX_VALUE

    private void solve() throws IOException {
        StringTokenizer st = new StringTokenizer(br.readLine());
        t = st.nextToken();
        p = st.nextToken();
        int n = t.length();
        int m = p.length();

        if (n < m) {
            return;
        }

        long patterHash = 0;
        for (int i = 0; i < m; i++) {
            int charValue = p.charAt(i) - 'a' + 1;
            patterHash = multiply(patterHash, prime, mod);
            patterHash = add(patterHash, charValue, mod);
        }

        long currentHash = 0;
        for (int i = 0; i < m; i++) {
            int charValue = t.charAt(i) - 'a' + 1;
            currentHash = multiply(currentHash, prime, mod);
            currentHash = add(currentHash, charValue, mod);
        }

        long maxPower = 1;
        for (int i = 0; i < m - 1; i++) {
            maxPower = multiply(maxPower, prime, mod);
        }

        for (int i = 0; i <= n - m; i++) {
            if (currentHash == patterHash) {
                System.out.print("Same hash found starting in index " + i + "
                    the substring is: ");
                System.out.println(t.substring(i, i + m));
            }
            if (i + m < n) {
                int head = t.charAt(i) - 'a' + 1;
                int next = t.charAt(i + m) - 'a' + 1;
```

```
                currentHash = subtract(currentHash, multiply(maxPower, head,
                    mod), mod);
                currentHash = multiply(currentHash, prime, mod);
                currentHash = add(currentHash, next, mod);
            }
        }
    }

    public long add(long a, long b, long mod) {
        return ((a % mod) + (b % mod))%mod;
    }

    public long subtract(long a, long b, long mod) {
        long result = ((a % mod) - (b % mod))%mod;
        if (result < 0) {
            result += mod;
        }
        return result;
    }

    public long multiply(long a, long b, long mod) {
        return ((a % mod) * (b % mod)) % mod;
    }

    public static void main(String[] args) throws IOException {
        new C().solve();
    }
}
```

# 6 tree

## 6.1 1 K-th Parent

1. K-th Parent.cpp

```
class TreeAncestor {
    int LOG = 20;
    int up[50000][20];
public:
    TreeAncestor(int n, vector<int>& parent) {
        memset(up, -1, 50000 * LOG * 4);
        for (int i = 0; i < n; i++) {
            up[i][0] = parent[i];
        }
        for (int k = 1; k < LOG; k++) {
            for (int i = 0; i < n; i++) {
                if (up[i][k-1] != -1)
                    up[i][k] = up[up[i][k-1]][k-1];
            }
        }
    }

    int getKthAncestor(int node, int k) {
        for (int i = 0; i < LOG; i++) {
            if (k & 1<<i) {
                node = up[node][i];
            }
            if (node == -1) return -1;
        }
        return node;
    }
};
```