Contents

1 Data Structure

1.1 BIT

```cpp
#define lowbit(k) (k & -k)
void add(vector<int> &tr, int id, int val) {
    for (; id <= n; id += lowbit(id)) {
        tr[id] += val;
    }
}
int sum(vector<int> &tr, int id) {
    int ret = 0;
    for (; id >= 1; id -= lowbit(id)) {
        ret += tr[id];
    }
    return ret;
}
```

1.2 Segment tree

```cpp
int dfs(int lef, int rig){
    if(lef + 2 == rig){
        if(num[lef] > num[rig-1]){
            return lef;
        }
        else{
            return rig-1;
        }
    }
    int mid = (lef + rig)/2;
    int p1 = dfs(lef, mid);
    int p2 = dfs(mid, rig);
    if(num[p1] > num[p2]){
        return p1;
    }
    else{
        return p2;
    }
}
```

1.3 Trie

```cpp
const int MAXL = ; // 自己填
const int MAXC = ;
struct Trie {
    int nex[MAXL][MAXC];
    int len[MAXL];
    int sz;
    void init() {
        memset(nex, 0, sizeof(nex));
        memset(len, 0, sizeof(len));
        sz = 0;
    }
    void insert(const string &str) {
        int p = 0;
        for (char c : str) {
            int id = c - 'a';
            if (!nex[p][id]) {
                nex[p][id] = ++sz;
            }
            p = nex[p][id];
        }
        len[p] = str.length();
    }
    vector<int> find(const string &str, int i) {
    int p = 0;
    vector<int> ans;
    for (; i < str.length(); i++) {
        int id = str[i] - 'a';
        if (!nex[p][id]) {
            return ans;
        }
        p = nex[p][id];
        if (len[p]) {
            ans.pb(len[p]);
        }
    }
    return ans;
    }
};
```

2 Dynamic Programming

2.1 Josephus

```cpp
int josephus (int n, int k) {
    // 有 n 個人圍成一圈，每 k 個一次
    return n > 1 ? (josephus(n-1 , k) + k) % n : 0;
}
// 回傳最後一人的編號, 0 index
```

2.2 LCS

```cpp
int LCS(string s1, string s2) {
    int n1 = s1.size(), n2 = s2.size();
    int dp[n1 + 1][n2 + 1];
    memset(dp, 0, sizeof(dp));
    // dp[i][j] = s1的前 i 個字元和 s2 的前 j 個字元
    for (int i = 1; i <= n1; i++) {
        for (int j = 1; j <= n2; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            }
            else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j
                    - 1]);
            }
        }
    }
    return dp[n1][n2];
}
```

2.3 LIS

```cpp
int LIS(vector<int> &a) {
    vector<int> s;
    for (int i = 0; i < a.size(); i++) {
        if (s.empty() || s.back() < a[i]) {
            s.push_back(a[i]);
        }
        else {
            *lower_bound(s.begin(), s.end(), a[i]) =
                a[i];
        }
    }
    return s.size();
}
```