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;
    }
};
```