

```
1  /*
2   | Trie |
3   Desc: Multiple string matching in O(max(s.size())) for all operations
4   Source: KawakiMeido
5   State: VERY Untested and old code lmao
6 */
7
8 struct Trie{
9
10    struct Node{
11        Node* child[2];
12        int cnt;
13        int l,r;
14        Node(){
15            child[0] = child[1] = NULL;
16            cnt = 0;
17        }
18    };
19
20    Node* Root;
21    int cnt;
22    Trie(){
23        Root = new Node();
24        cnt = 0;
25    }
26
27    void Init(){
28        clr(Root);
29    }
30    void clr(Node* cur){
31        if (cur->child[0] != NULL){
32            clr(cur->child[0]);
33            cur->child[0] = NULL;
34        }
35        if (cur->child[1] != NULL){
36            clr(cur->child[1]);
37            cur->child[1] = NULL;
38        }
39        if (cur != Root) delete cur;
40    }
41
42    void Add(int x, int pos){
43        Node* cur = Root;
44        for (int i=29; i ≥ 0; i--){
45            int idx = ((x>>i)&1);
46            if (cur->child[idx] == NULL) cur->child[idx] = new Node();
47            cur = cur->child[idx];
48            cur->cnt++;
49        }
50    }
51
52    void Print(){
53        cout << "Root: " << Root->cnt << endl;
54        for (int i=0; i<29; i++){
55            cout << "Child " << i << ": ";
56            for (int j=0; j<2; j++)
57                cout << cur->child[i]->cnt << " ";
58            cout << endl;
59        }
60    }
61
62    int Search(string s){
63        Node* cur = Root;
64        for (int i=0; i<s.size(); i++){
65            int idx = ((s[i]>>i)&1);
66            if (cur->child[idx] == NULL) return -1;
67            cur = cur->child[idx];
68        }
69        return cur->cnt;
70    }
71
72    int Delete(string s){
73        Node* cur = Root;
74        for (int i=0; i<s.size(); i++){
75            int idx = ((s[i]>>i)&1);
76            if (cur->child[idx] == NULL) return -1;
77            cur = cur->child[idx];
78        }
79        if (cur->cnt == 0) delete cur;
80        return cur->cnt;
81    }
82
83    int GetCount(string s){
84        Node* cur = Root;
85        for (int i=0; i<s.size(); i++){
86            int idx = ((s[i]>>i)&1);
87            if (cur->child[idx] == NULL) return -1;
88            cur = cur->child[idx];
89        }
90        return cur->cnt;
91    }
92
93    void PrintCount(string s){
94        Node* cur = Root;
95        for (int i=0; i<s.size(); i++){
96            int idx = ((s[i]>>i)&1);
97            if (cur->child[idx] == NULL) return -1;
98            cur = cur->child[idx];
99        }
100       cout << cur->cnt << endl;
101    }
102 }
```

```
50     }
51     int Get(int x, Node* cur, int lg){
52         int res = 0;
53         if (cur->cnt ≤ 0) return INF;
54         for (int i=lg; i ≥ 0; i--){
55             int idx = ((x>>i)&1);
56             if (cur->child[idx] == NULL){
57                 res = res+(1<<i);
58                 cur = cur->child[(idx+1)%2];
59             }
60             else{
61                 cur = cur->child[idx];
62             }
63         }
64         return res;
65     }
66 };
67
68 Trie TR;
```