Reiknirit Ahos og Corasicks (1975)

Bergur Snorrason

4. apríl 2022

- TODO

```
25 #define ALPHABET 128
26 #define MAXN 1000000
27 typedef struct { int v, n; } listnode;
28 typedef struct { int t[ALPHABET], g[ALPHABET], l, e, p, c, d; } trienode;
29 typedef struct { int s, r, l; trienode m[MAXN]; listnode w[MAXN];} trie;
30 int val(char c) { return c; }
31 int list_node(trie *t, int v, int n)
32 {
33     t->w[t->l].v = v, t->w[t->l].n = n;
34     return t->l++;
35 }
36 int trie_node(trie *t, int p, int c)
37 {
38     int i;
39     for (i = 0; i < ALPHABET; i++)
40         t->m[t->s].t[i] = t->m[t->s].g[i] = -1;
41     t->m[t->s].l = -1, t->m[t->s].e = -1, t->m[t->s].p = p,
42         t->m[t->s].c = c, t->m[t->s].d = -1;
43     return t->s++;
44 }
45 void trie_init(trie *t) { t->s = t->l = 0, t->r = trie_node(t, -1, -1); }
46
47 void trie_insert(trie *t, char *s, int x)
48 {
49     int h;
50     for (h = t->r; *s; h = t->m[h].t[val(*s++)])
51         if (t->m[h].t[val(*s)] == -1)
52             t->m[h].t[val(*s)] = trie_node(t, h, val(*s));
53     t->m[h].l = list_node(t, x, t->m[h].l);
54 }
55
56 int trie_step(trie*, int, int);
57 int trie_suffix(trie *t, int h)
58 { // dp-lookup hjálparfall fyrir suffix link
59     if (t->m[h].d != -1) return t->m[h].d;
60     if (h == t->r || t->m[h].p == t->r) return t->m[h].d = t->r;
61     return t->m[h].d = trie_step(t, trie_suffix(t, t->m[h].p), t->m[h].c);
62 }
```