

Reiknirit Ahos og Corasicks (1975)

Bergur Snorrason

6. apríl 2022

- Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .

- ▶ Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .
- ▶ Látum $|s|$ tákna lengd strengsins s og $|p| = |p_1| + \dots + |p_n|$.

- ▶ Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .
- ▶ Látum $|s|$ tákna lengd strengsins s og $|p| = |p_1| + \dots + |p_n|$.
- ▶ Við viljum finna alla hlutstrengi s sem eru í listanum p .

- ▶ Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .
- ▶ Látum $|s|$ tákna lengd strengsins s og $|p| = |p_1| + \dots + |p_n|$.
- ▶ Við viljum finna alla hlutstrengi s sem eru í listanum p .
- ▶ Við getum notað reiknirit Knuths, Morrisar og Pratt's n sinnum.

- ▶ Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .
- ▶ Látum $|s|$ tákna lengd strengsins s og $|p| = |p_1| + \dots + |p_n|$.
- ▶ Við viljum finna alla hlutstrengi s sem eru í listanum p .
- ▶ Við getum notað reiknirit Knuths, Morrisar og Pratts n sinnum.
- ▶ Þessi aðferð hefur tímaflækjuna $\mathcal{O}(n \cdot |s| + |p|)$.

- ▶ Gerum ráð fyrir að við séum með stafróð Σ , streng s og list af n strengjum p , þar sem j -ti strengurinn kallast p_j .
- ▶ Látum $|s|$ tákna lengd strengsins s og $|p| = |p_1| + \dots + |p_n|$.
- ▶ Við viljum finna alla hlutstrengi s sem eru í listanum p .
- ▶ Við getum notað reiknirit Knuths, Morrisar og Pratt's n sinnum.
- ▶ Þessi aðferð hefur tímaflækjuna $\mathcal{O}(n \cdot |s| + |p|)$.
- ▶ Reiknirit Ahos og Corasicks bætir þetta.

- Byrjum á að setja all strengina í p inn í forgangstré T .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnútarnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnútarnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnútarnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnútarnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.
- ▶ Annars þurfum við að fara aftur í hnút w þannig að strengurinn sem svarar til w er bakstrengur strengsins sem svarar til v .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnúturnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.
- ▶ Annars þurfum við að fara aftur í hnút w þannig að strengurinn sem svarar til w er bakstrengur strengsins sem svarar til v .
- ▶ Við viljum hafa strenginn w sem lengstan (með öðrum orðum viljum við fara sem styst aftur).

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnúturnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.
- ▶ Annars þurfum við að fara aftur í hnút w þannig að strengurinn sem svarar til w er bakstrengur strengsins sem svarar til v .
- ▶ Við viljum hafa strenginn w sem lengstan (með öðrum orðum viljum við fara sem styst aftur).
- ▶ Við köllum þessa leggi *basktrengsleggi* (e. *suffix links*).

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnúturnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.
- ▶ Annars þurfum við að fara aftur í hnút w þannig að strengurinn sem svarar til w er bakstrengur strengsins sem svarar til v .
- ▶ Við viljum hafa strenginn w sem lengstan (með öðrum orðum viljum við fara sem styst aftur).
- ▶ Við köllum þessa leggi *basktrengsleggi* (e. *suffix links*).
- ▶ Takið eftir að þeir eru í raun óháðir bókstafnum c .

- ▶ Byrjum á að setja all strengina í p inn í forgangstré T .
- ▶ Við viljum síðan búa til stöðuvél úr T .
- ▶ Hnútarnir í trénu eru stöðurnar, en okkur vantar að finna færslur fyrir hverja stöðu og bókstaf í Σ .
- ▶ Gerum ráð fyrir að við séum í hnút v í T og viljum með fyrir staf c í Σ .
- ▶ Ef það er leggur úr v merktur með c þá ferðumst við eftir honum.
- ▶ Annars þurfum við að fara aftur í hnút w þannig að strengurinn sem svarar til w er bakstrengur strengsins sem svarar til v .
- ▶ Við viljum hafa strenginn w sem lengstan (með öðrum orðum viljum við fara sem styst aftur).
- ▶ Við köllum þessa leggi *bakstrengsleggi* (e. *suffix links*).
- ▶ Takið eftir að þeir eru í raun óháðir bókstafnum c .
- ▶ Við látum bakstrengslegg rótarinn benda á sjálfa sig.

- ▶ Hvernig finnum við alla bakstrengsleggina?

- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.

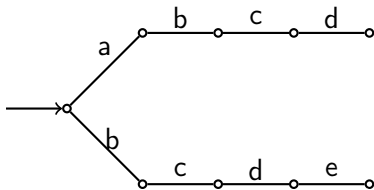
- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.
- ▶ Látum $f(w, c)$ tákna færslu úr stöðu w með staf c og $g(w)$ tákna bakstrengslegg w .

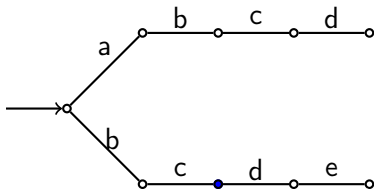
- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.
- ▶ Látum $f(w, c)$ tákna færslu úr stöðu w með staf c og $g(w)$ tákna bakstrengslegg w .
- ▶ Gerum einnig ráð fyrir að foreldri v sé p og $f(p, a) = v$.

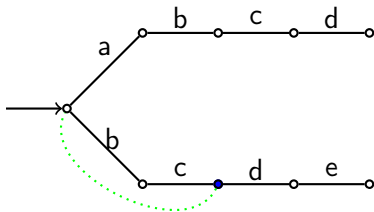
- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.
- ▶ Látum $f(w, c)$ tákna færslu úr stöðu w með staf c og $g(w)$ tákna bakstrengslegg w .
- ▶ Gerum einnig ráð fyrir að foreldri v sé p og $f(p, a) = v$.
- ▶ Við sjáum þá að $g(v) = f(g(p), a)$.

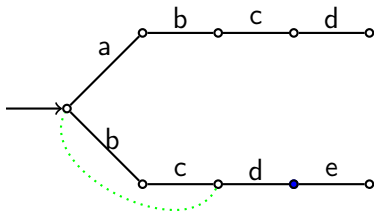
- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.
- ▶ Látum $f(w, c)$ tákna færslu úr stöðu w með staf c og $g(w)$ tákna bakstrengslegg w .
- ▶ Gerum einnig ráð fyrir að foreldri v sé p og $f(p, a) = v$.
- ▶ Við sjáum þá að $g(v) = f(g(p), a)$.
- ▶ Með öðrum orðum förum við upp í foreldrið, ferðumst eftir bakstrenglegg þaðan og færum okkur í stöðuvélinni eftir a .

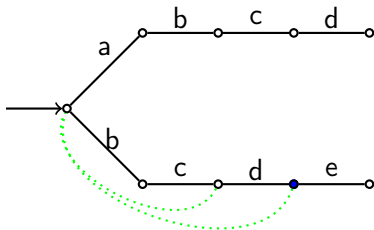
- ▶ Hvernig finnum við alla bakstrengsleggina?
- ▶ Við notum kvika bestun.
- ▶ Látum $f(w, c)$ tákna færslu úr stöðu w með staf c og $g(w)$ tákna bakstrengslegg w .
- ▶ Gerum einnig ráð fyrir að foreldri v sé p og $f(p, a) = v$.
- ▶ Við sjáum þá að $g(v) = f(g(p), a)$.
- ▶ Með öðrum orðum förum við upp í foreldrið, ferðumst eftir bakstrenglegg þaðan og færum okkur í stöðuvélinni eftir a .
- ▶ Við höfum þá rakningarformúlu sem við getum notað til að reikna bakstrengshlekkina.

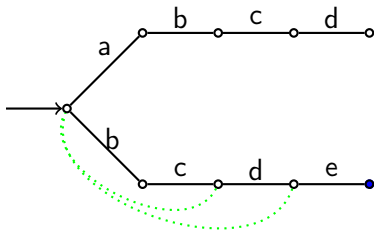


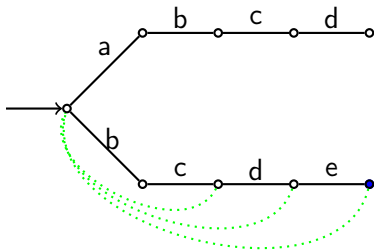


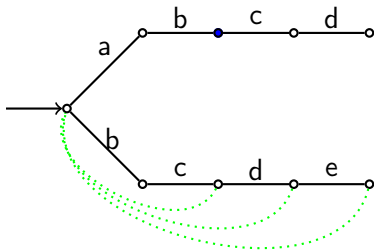


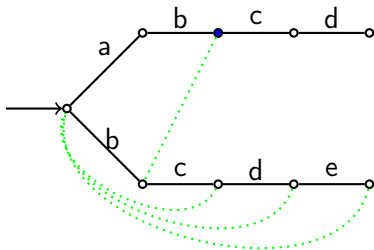


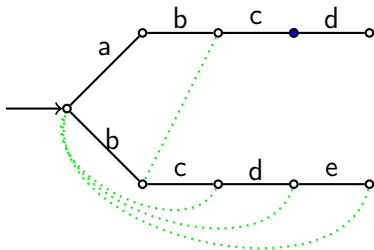


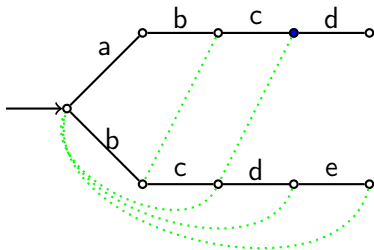


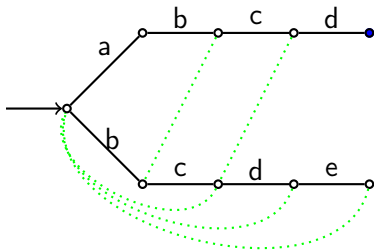


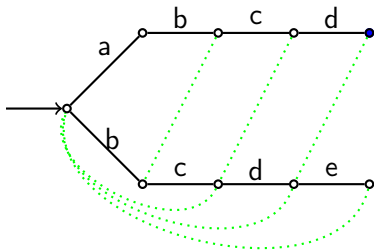








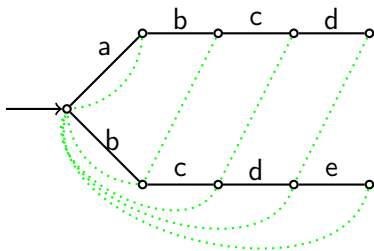




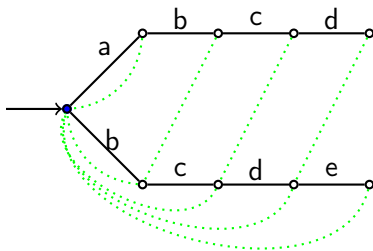
- ▶ Í T merkjum við lokastöður þar sem strengir enda.

- ▶ Í T merkjum við lokastöður þar sem strengir enda.
- ▶ Við ferðumst svo í gegnum strenginn s og hliðrum stöðvélinni miðað við stafina í s .

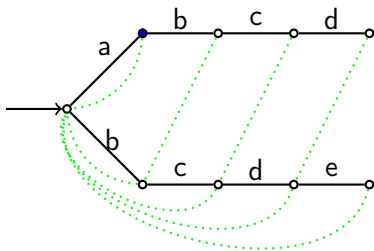
„abcdcdeaaabcdeabcxab”



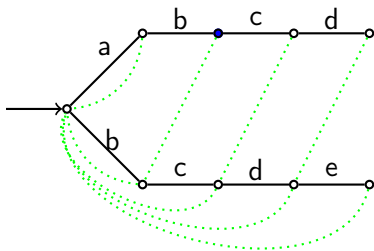
„abcdcdeaaabcdeabcxab”



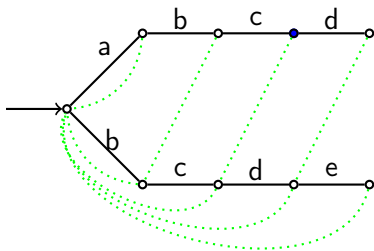
„bcdcd eaaabcdeabcxab”



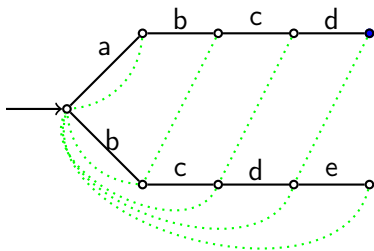
„cdcdeaaaabcdeabcxab”



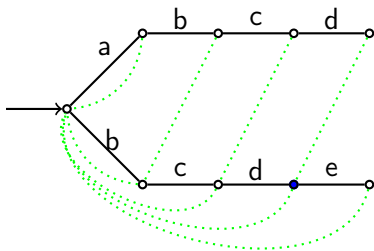
„dcdeaaaabcdeabcxab”



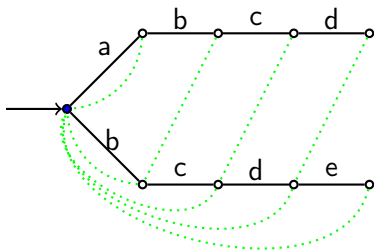
„cdeaaabcdeabcxab”



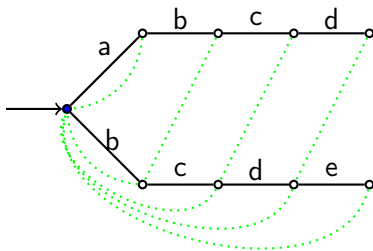
„cdeaaabcdeabcxab”



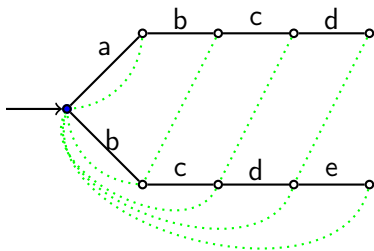
„cdeaaaabcdeabcxab”



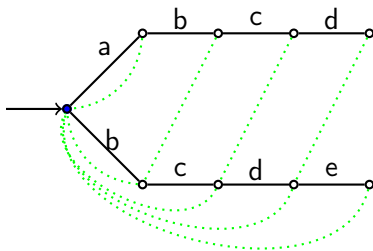
„deaaaabcdeabcxab”



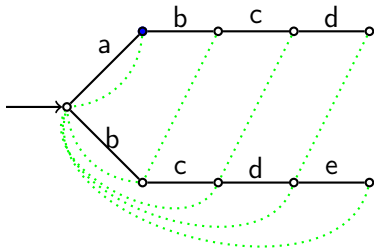
„eaaabcdeabcxab”



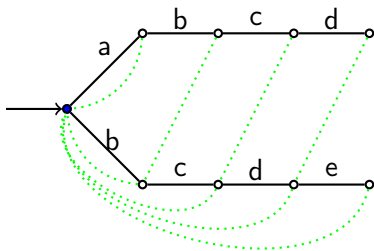
„aaabcdeabcxab”



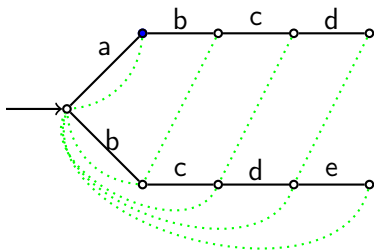
„aabcdeabcxab”



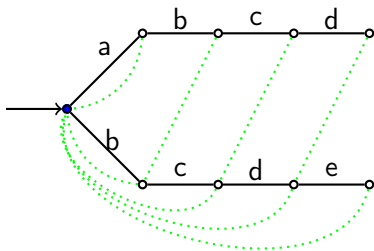
„aabcdeabcxab”



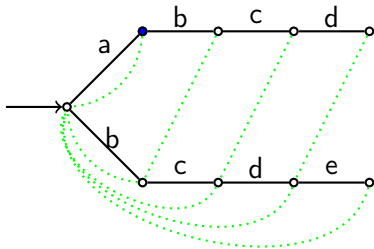
„abcdeabcxab”



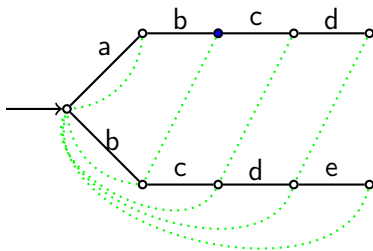
„abcdeabcxab”



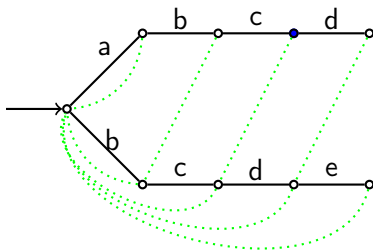
„bcdeabcxab”



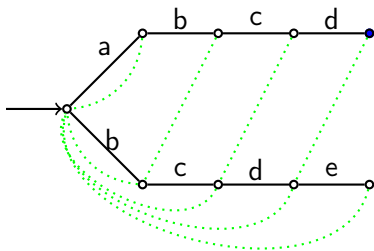
„cdeabcxab”



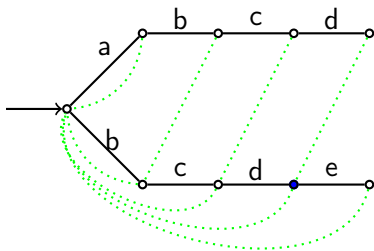
„deabcxab”



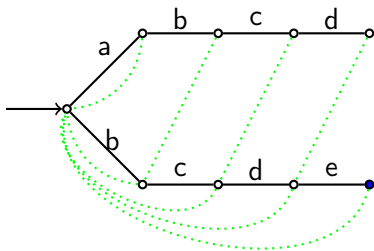
„eabcxab”



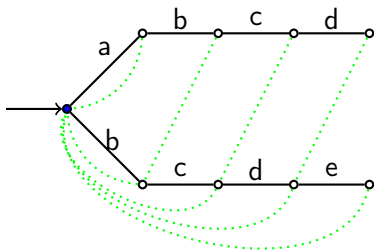
„eabcxab”



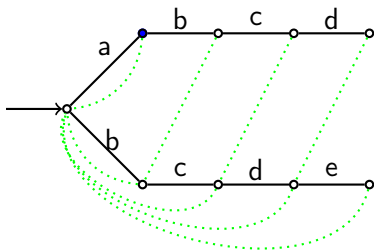
„abcxab”



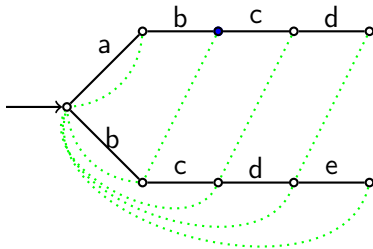
„abcxab”



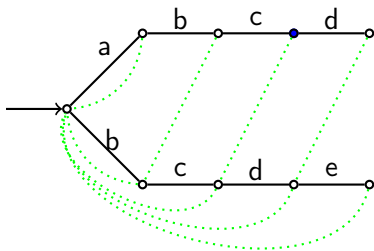
„bcxab”



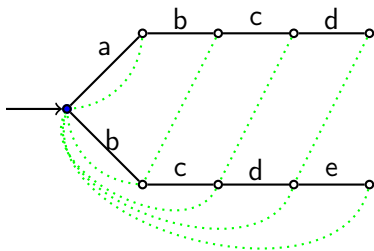
„cxab”



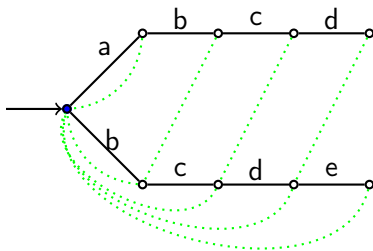
„xab”



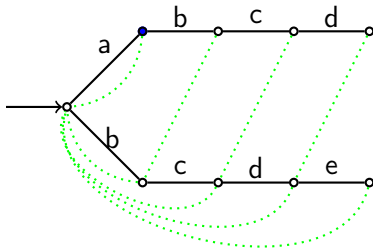
„xab“



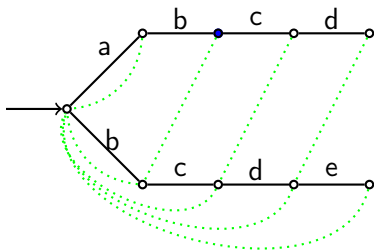
„ab”



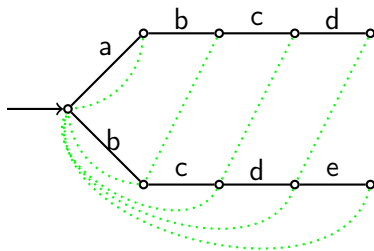
„b”



”
”



”
”

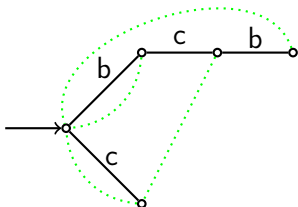


- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .

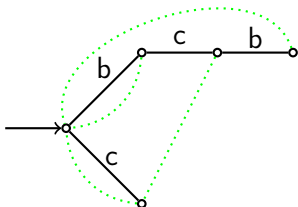
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?

- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.

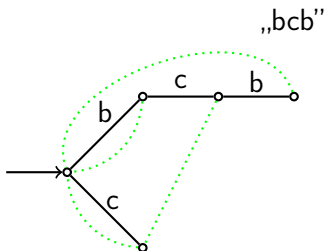
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



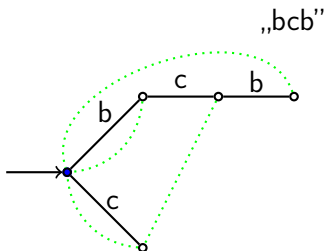
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



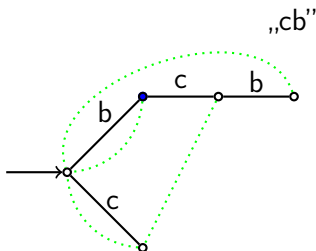
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



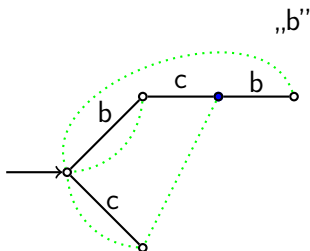
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



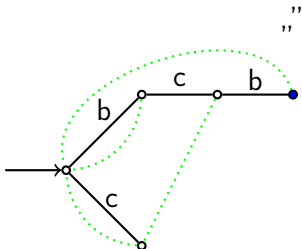
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



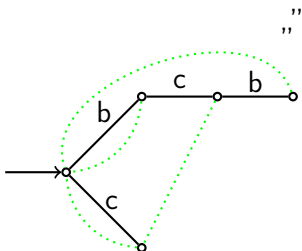
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



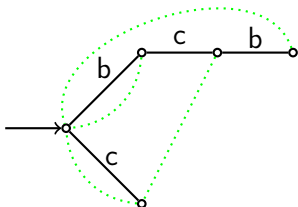
- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.

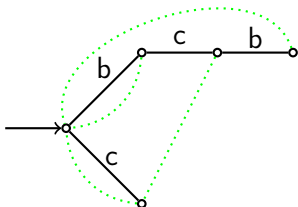


- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



- ▶ Til að koma í vega fyrir að tímaflækjan verði of slæm þá notum við aftur kvika bestun.

- ▶ Ljóst er að alltaf þegar við erum í lokstöðu þá erum við með hlutstreng í s sem er í p .
- ▶ En eru þetta einu slíku hlutstrengirnir?
- ▶ Nei, við þurfum líka, í hverju skrefi, að athuga hvort við getum komist í lokastöðu ef við ferðumst eftir bakstrengsleggjum.



- ▶ Til að koma í vega fyrir að tímaflækjan verði of slæm þá notum við aftur kvika bestun.
- ▶ Við bætum í raun leggjum inn í tréð, sem við köllum *lokaleggi* (e. *exit link*).

- ▶ Í útfærslunni munum við notast við þrjú hjálparföll.

- ▶ Í útfærslunni munum við notast við þrjú hjálparföll.
- ▶ Fyrsta heitir `trie_step(...)` og er það notað til að ferðast um stöðuvélina.

- ▶ Í útfærslunni munum við notast við þrjú hjálparföll.
- ▶ Fyrsta heitir `trie_step(...)` og er það notað til að ferðast um stöðuvélina.
- ▶ Annað heitir `trie_step(...)` og er það notað til að finna bakstrengsleggi.

- ▶ Í útfærslunni munum við notast við þrjú hjálparföll.
- ▶ Fyrsta heitir `trie_step(...)` og er það notað til að ferðast um stöðuvélina.
- ▶ Annað heitir `trie_step(...)` og er það notað til að finna bakstrengsleggi.
- ▶ Þriðja heitir `trie_exit(...)` og er það notað til að finna lokaleggina.

- ▶ Í útfærslunni munum við notast við þrjú hjálparföll.
- ▶ Fyrsta heitir `trie_step(...)` og er það notað til að ferðast um stöðuvélina.
- ▶ Annað heitir `trie_step(...)` og er það notað til að finna bakstrengsleggi.
- ▶ Þriðja heitir `trie_exit(...)` og er það notað til að finna lokaleggina.
- ▶ Öll þessi föll eru endurkvæm og notast við minnun.

```

24 #define ALPHABET 128
25 #define MAXN 1000000
26 typedef struct { int v, n; } listnode;
27 typedef struct { int t[ALPHABET], l, e, p, c, d; } trienode;
28 typedef struct { int s, r, l; trienode m[MAXN]; listnode w[MAXN]; } trie;
29 int val(char c) { return c; }
30 int list_node(trie *t, int v, int n)
31 {
32     t->w[t->l].v = v, t->w[t->l].n = n;
33     return t->l++;
34 }
35 int trie_node(trie *t, int p, int c)
36 {
37     int i;
38     for (i = 0; i < ALPHABET; i++) t->m[t->s].t[i] = -1;
39     t->m[t->s].l = -1, t->m[t->s].e = -1, t->m[t->s].p = p,
40     t->m[t->s].c = c, t->m[t->s].d = -1;
41     return t->s++;
42 }
43 void trie_init(trie *t) { t->s = t->l = 0, t->r = trie_node(t, -1, -1); }
44
45 void trie_insert(trie *t, char *s, int x)
46 {
47     int h;
48     for (h = t->r; *s; h = t->m[h].t[val(*s++)])
49         if (t->m[h].t[val(*s)] == -1)
50             t->m[h].t[val(*s)] = trie_node(t, h, val(*s));
51     t->m[h].l = list_node(t, x, t->m[h].l);
52 }

```



```

54 int trie_step(trie*, int, int);
55 int trie_suffix(trie *t, int h)
56 { // dp-lookup hjálparfall fyrir suffix link
57     if (t->m[h].d != -1) return t->m[h].d;
58     if (h == t->r || t->m[h].p == t->r) return t->m[h].d = t->r;
59     return t->m[h].d = trie_step(t, trie_suffix(t, t->m[h].p), t->m[h].c);
60 }
61
62 int trie_step(trie *t, int h, int c)
63 { // dp-lookup hjálparfall til að ítra
64     if (t->m[h].t[c] != -1) return t->m[h].t[c];
65     return t->m[h].t[c] = h == t->r ? t->r :
66         trie_step(t, trie_suffix(t, h), c);
67 }
68
69 int trie_exit(trie *t, int h)
70 { // dp-lookup hjálparfall fyrir exit link
71     if (t->m[h].e != -1) return t->m[h].e;
72     if (h == 0 || t->m[h].l != -1) return t->m[h].e = h;
73     return t->m[h].e = trie_exit(t, trie_suffix(t, h));
74 }

```

```

76 trie t; // global svo pað keyri locally.
77 int aho_corasick(char *s, char **p, int m)
78 {
79     trie_init(&t);
80     int h, i, j, k, w, l[m];
81     for (i = 0; i < m; i++) l[i] = strlen(p[i]);
82     for (i = 0; i < m; i++) trie_insert(&t, p[i], i);
83     printf(" ");
84     for (i = 0; i < strlen(s); i++) printf("%0d", i%10); printf("\n");
85     printf("searching in %s\n", s);
86     for (i = 0, j = 0, h = t.r; ; j++)
87     {
88         k = trie_exit(&t, h);
89         while (t.m[k].l != -1)
90         {
91             for (w = t.m[k].l; w != -1; w = t.w[w].n)
92             {
93                 printf(" '%s' found at index %0d\n",
94                     p[t.w[w].v], j - l[t.w[w].v]);
95             }
96             k = trie_exit(&t, trie_suffix(&t, k));
97         }
98         h = trie_step(&t, h, val(*s));
99         if (*s++ == '\0') break;
100     }
101     return i;
102 }

```

- Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(\quad)$.

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p| + k)$.

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p| + k)$.
- ▶ Ef við höfum bara áhuga á að finna töluna k getum við breytt `trie_exit(...)` þannig að það reikni fjölda lokastaða á leiðinni að rót eftir bakstrengsleggjum.

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p| + k)$.
- ▶ Ef við höfum bara áhuga á að finna töluna k getum við breytt `trie_exit(...)` þannig að það reikni fjölda lokastaða á leiðinni að rót eftir bakstrengsleggjum.
- ▶ Þá verður tímaflækjan $\mathcal{O}(\quad)$.

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p| + k)$.
- ▶ Ef við höfum bara áhuga á að finna töluna k getum við breytt `trie_exit(...)` þannig að það reikni fjölda lokastaða á leiðinni að rót eftir bakstrengsleggjum.
- ▶ Þá verður tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p|)$.

- ▶ Gerum ráð fyrir að strengirnar í p komi fyrir k sinnum í s .
- ▶ Þá er tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p| + k)$.
- ▶ Ef við höfum bara áhuga á að finna töluna k getum við breytt `trie_exit(...)` þannig að það reikni fjölda lokastaða á leiðinni að rót eftir bakstrengsleggjum.
- ▶ Þá verður tímaflækjan $\mathcal{O}(|s| + |\Sigma| \cdot |p|)$.
- ▶ Takið eftir að, ef stafrófið er takmarkað, þá er seinni tímaflækjan línuleg.

