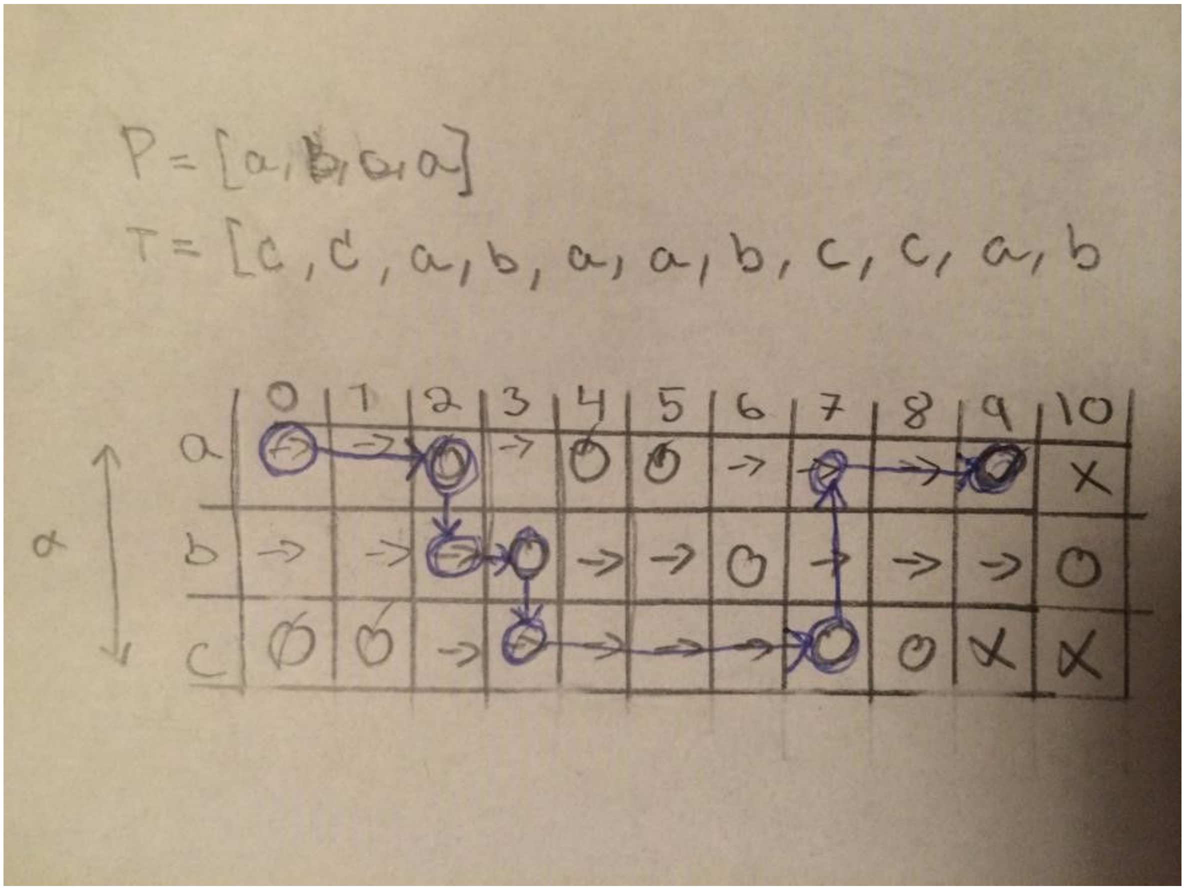
Common Subsequence

For at kunne evaluerer hvorvidt T og P har en subsequence til fælles, må jeg iterativt, i længden af P, lede efter bogstaver der matcher P. Hele P er matched, må det da være en sub squence. Udfordringen ligger derfor i at kunne finde en successor i T, en successor i den forstand at det matcher overens med det næste bogstav i P. Eksempelvis ville en bogstavs specifik successor for 0 indekset $P = a,\ T = [c, b, a]$ være 2(der hvor a er placeret), hvorimod en successor for indeks 2 af $P = b,\ T = [c, b, a, b]$ være 3.

1.1 Give a data structure that answers queries in O(|P|) time and uses little space. Hint: a good solution depends on both the size of the alphabet and the length of T.

Følgende datastruktur:

opbyg α arrays hver af størrelse $\tilde{|T|}$. Hvert af disse arrays indeholder information om ét bogstav's placering i T, således at hvert indeks i arrayet enten er en indikation på bogstavet er placeret på indekset, eller en pointer til hvor det næste er placeret. På den måde ville en query i konstant tid kunne finde bogstav specifikke successors. Figur 1 illustrerer et eksempel.



Plads

Siden et array i str. $|T|$ er bygget over α bogstaver, må pladsen været:

$$\Theta(|T|) * \Theta(\alpha) = \Theta(|T| \; * \; \alpha)$$

Tid

Da en successor tager $\Theta(1)$ tid at evaluere, og det skal gøres $\Theta(|P|)$ gange, må køretiden da blive:

$$\Theta(1) * \Theta(|P|) \; = \; \Theta(|P| \; * \; 1) \; = \; \Theta(|P|)$$

Evaluering

Derfor er både plads, og tid, indenfor det forventede.

1.2 *Give a data structure that uses O(n) space and supports fast queries. The query time should depend on P.*

Følgende datastruktur:

Opbyg α arrays hver af størrelsen på mængden af hvert bogstav i T. Hvert af disse arrays indeholder placeringen på hvert af bogstaverne i T, e.g.

$T = a, a, b, b, a$ ville give

$a = [0, 1, 4]$

$b = [2, 3]$

Hvert af disse Arrays bliver der så bygget et Y-fast trie over, hvilket ville give følgende

Plads

Siden y-fast tries vokser lineært, så har fordelingen af bogstaver i de α forskellige Y-fast tries ikke nogen betydning, hvorfor pladsen altid ville være:

$$\Theta(n)$$

Dertil, for at sænke køretiden i subsequence algoritmen, ville der også være et hashtable der peger til alle de α forskellige Y-fast tries. Dette ville ikke have nogen pladsmæssig betydning, da α aldrig ville kunne overstige n, hvorfor der selv i “worst-case” hvor alle bogstaver kun bliver præsenteret én gang, hvormed $\alpha = n$, ikke ville være større pladsbrug end $\Theta(n + n) = \Theta(n)$.

Tid

Algoritmisk ville det at finde subsequencen ske ved have en “plads” i T som algoritmen er nået til, lad os denote den med j, denne plads ville så blive benyttet til at spørge efter “successor” i det Y-fast trie som angiver pladserne på det bogstav man leder efter. Så i eksemplet med T ovenover, ville en successor til $j = 2$ i det Y-fast trie der indeholder placeringerne på a returnere med $j = 4$ og en $true$.

Validiteten bliver derigennem skabt hvis P, for alle dens elementer, er i stand til at kunne finde successors i de forskellige Y-fast tries P skal benytte.

Tidsmæssigt tager det $\Theta(1)$ at finde det Y-fast trie der repræsentere bogstavet(eftersom vi benytter et hashtable), $\Theta(log\ log\ u)$ at finde successor i et Y-fast trie, hvor u referer til det univers T bitmæssigt kræver for at kunne repræsentere pladserne på alle elementerne i dets string. Hvilket gør at evalueringen for ét bogstav i P tager: $\Theta(log\ log\ u)\ +\ \Theta(1)\ =\ \Theta(log\ log\ u)$, og siden dette gentages $|P|$ gange, må tiden derfor blive:

$$\Theta(|P|\ *\ log\ log\ u)$$

Evaluering

Derfor er både plads, og tid, indenfor det forventede.