En anvendelse av binære søketrær:
Svadageneratoren

Problemet som skal løses

- Les en lengre tekst skrevet i et bestemt språk, f.eks. engelsk eller tysk
- Registrer data som på en eller annen måte beskriver noe som er typisk for dette språket
- Skriv deretter ut en lengre *tilfeldig* tekst (meningsløs *svada*) som ligner på språket i den teksten som er lest

Eksempel: Tysk svada

Input*: Wir reden heute sehr oft von demokratischen Idealen; das heißt nicht in Deutschland, sondern in der anderen Welt wird davon geredet. Denn wir in Deutschland haben ja dieses demokratische Ideal einst zur Genüge kennengelernt; wenn die andere Welt also heute wieder dieses Ideal preist, so können wir darauf zunächst nur erwidern, daß dieses Ideal das deutsche Volk ja mindestens 15 Jahre lang in Reinkultur kennenzulernen Gelegenheit hatte, und wir selbst haben ja nur das Erbe dieser Demokratie angetreten....

Output: Deutschen, sie haben, daß nun in einen auf mich das Rechtig, eine wir erweite Phantom ist es in freiung gepredigt, für die viellein Sie eigenen, meinem größte erwiders komme, die gemeinst wehrlos geboten es ist und einem Haß sich ist und wird. - I wo den Staat, und Angriff. In diesen der eine Person, so Gottes Werte inner Europaganda gab ich nichter der armen vorhatte etwas würde eingen tapfer aber aller. Ich leiderlegt? Haben überwunde über die Gewaltigung nicht überstelle nicht geschreiternahm, das vom konnte, zu vergangesehe...

*: Fra Adolf Hitlers tale i Sportpalast Berlin, 30. januar 1940

Eksempel: Engelsk svada

Input*: Of this I am quite sure, that if we open a quarrel between the past and the present, we shall find that we have lost the future. Therefore, I cannot accept the drawing of any distinctions between members of the present Government. It was formed at a moment of crisis in order to unite all the Parties and all sections of opinion. It has received the almost unanimous support of both Houses of Parliament. Its members are going to stand together, and, subject to the authority of the House of Commons, we are going to govern...

Output: Those to columns, which his been; and the solid, preven-eight and columns, depends our declared free and stand cared scienced not, before been France at see how the Navy is no do. We are yet at the which of the possible losses, from the come dangerous battle intend in their Priment time, together people, that the exploits craft and casting formations which we are size; It have, and for the discussions any day across the part the add great quarter when we are are and trained. Withought ago as two off the sea invasion,...

*: Fra Winston Churchills tale i The House of Commons, London, 18. juni, 1940

Hvorfor løse dette problemet?

- En svadagenerator kan ikke brukes til noe fornuftig, medfører mye programmeringsarbeid for «ingenting»...
- Men:
  - Den er et fint eksempel på smart anvendelse av algoritmer og datastrukturer til å løse et relativt komplekst problem
  - Problemet egner seg godt for bruk av binære søketrær
  - Løsningen gir god forståelse av hvordan søketrær og tilhørende algoritmer fungerer, her er det mye å lære...
  - Det er morsomt (for oss nerder, i det minste) å lage en fungerende løsning

Karakterisering av språk: Tegnsekvenser

- Et språk kan gjenkjennes ved å se på hvilke bokstaver (og andre tegn) som ofte kommer *rett etter hverandre*
- F.eks. vil en tysk tekst typisk inneholde mange forekomster av tegnsekvensene «sch», «eis» og «en »
- En engelsk tekst kan ha mange «the», « wh» og «ing»
- I svadageneratoren *lagrer* vi først alle tegnsekvenser som finnes i input, sammen med frekvensinformasjon
- Vi skriver deretter ut en tilfeldig tekst som *bare* inneholder de tegnsekvensene som vi har lest inn

Innlesing av *n*-sekvenser

- Teksten leses tegn for tegn
- All *påfølgende* whitespace, inkludert linjeskift, erstattes med en enkel `' '` (space)
- Alle sekvenser av *n* tegn som kommer rett etter hverandre (såkalte *n*-sekvenser), skal leses og lagres
- Typiske verdier for *n* er $2 \leq n \leq 6$
- Likheten mellom input og output blir større hvis vi bruker lengre *n*-sekvenser
- Lagrer alle sekvenser, også de som inneholder skilletegn, tall, spesialtegn og space i tillegg til bokstaver

n-sekvenser: Eksempel med *n* = 3

- Input:

  «Wir reden heute sehr oft von demokratischen Idealen »

- 3-sekvenser:

```
«Wir» «ir » «r r» « re» «red» «ede» «den» «en »
«n h» « he» «heu» «eut» «ute» «te » «e s» « se»
«seh» «ehr» «hr » «r o» « of» «oft» «ft » «t v»
« vo» «von» «on » «n d» « de» «dem» «emo» «mok»
«okr» «kra» «rat» «ati» «tis» «isc» «sch» «che»
«hen» «en » «n I» « Id» «Ide» «dea» «eal» «ale»
«len» «en »
```

- Java-kode for innlesing: `sekvensLeser.java`

Lagring av *n*-sekvenser

- Bruker et (stort) binært søketre til å registrere *alle ulike n*-sekvenser som finnes i teksten
- Treet skal ha én node for *hver ulik n*-sekvens
- Søketreet skal være sortert sortert stigende på *tegnverdi* (dvs. alfabetisk for bokstaver)
- Når vi senere skal skrive ut svada, trekker vi hele tiden tegnsekvenser fra dette søketreet

Lagring av frekvensinformasjon

- For hver av de ulike *n*-sekvensene som forekommer i teksten, må vi lagre data som sier noe om hvor ofte sekvensen forekommer
- Enkleste løsning:
  - Bare lagre antall ganger hver sekvens forekommer, sammen med selve sekvensen i søketreet
- Ulempe:
  - Gjør det «fiklete» å generere svada som både har samme frekvenser av *n*-sekvenser *og* ligner på originalteksten

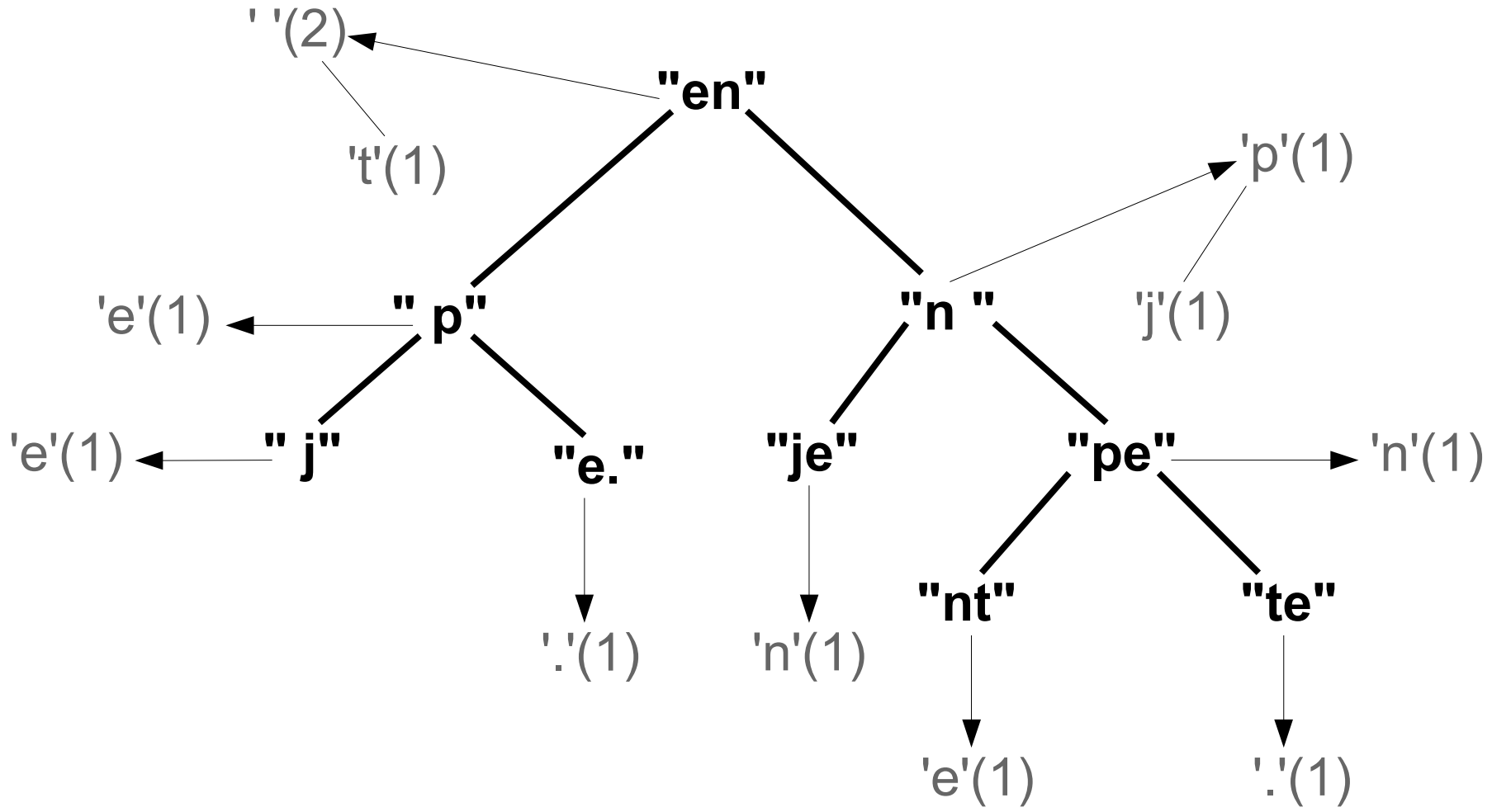
Smartere løsning: Lagring av etterfølgertegn til hver *n*-sekvens

- For *hver n*-sekvens oppretter vi et nytt (mindre) søketre til å registrere *alle* enkelt-tegn som kom rett *etter* denne sekvensen. Det «lille» søketreet er sortert på tegnverdi.
- Hver node i søketreet lagrer selve etterfølgertegnet og *antall ganger* dette tegnet forekom etter denne sekvensen
- Ved utskrift av neste tegn velger vi alltid et av tegnene som ligger lagret i etterfølgertreet til *n*-sekvensen som sist ble skrevet ut
- Vil da bare kunne skrive ut *n*-sekvenser som finnes i den opprinnelige teksten

Eksempel: *n* = 2, input: "en pen jente.."

2-sekvenser:

"en" "n " " p" "pe" "en" "n " " j" "je" "en" "nt" "te" "e."



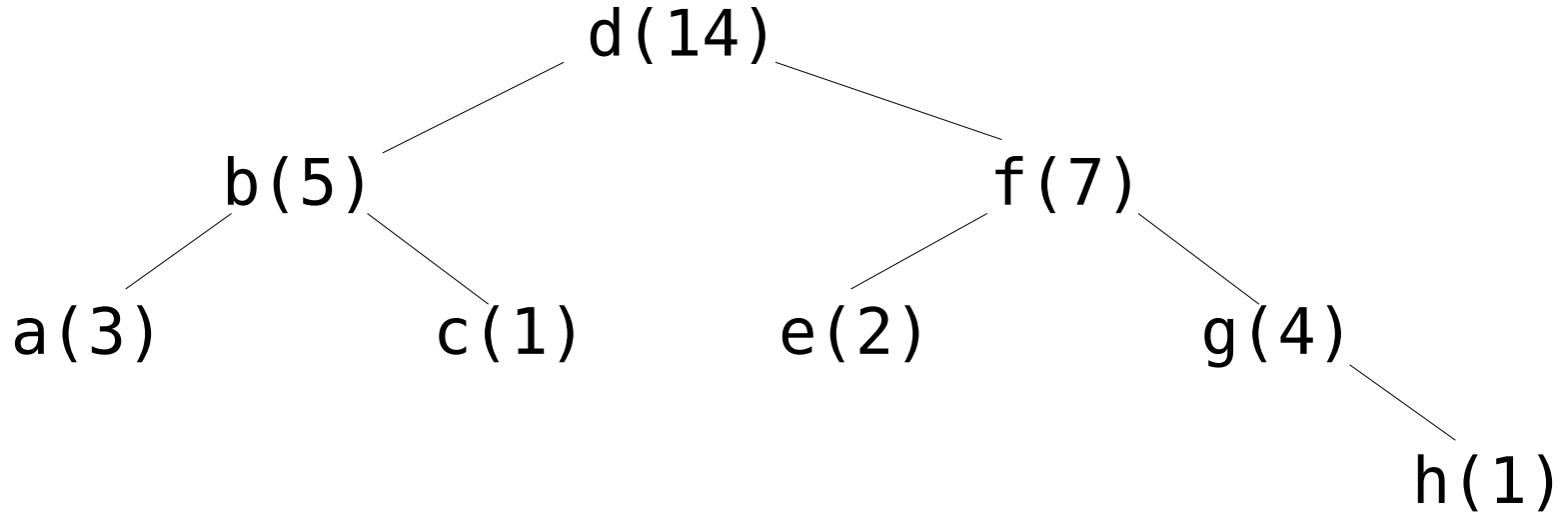
En enda smartere løsning

- Løsningen på forrige side lagrer antall forekomster av hvert tegn i etterfølgertreet til en *n*-sekvens
- Det viser seg at det blir enklere og raskere å trekke tilfeldige etterfølgertegn hvis vi i stedet, for hver node i de «små» søketrærne, lagrer:
  - Antall forekomster av tegn som finnes i *hele subtreet* som en etterfølgernode er rot i
  - Kommer tilbake til hvordan vi faktisk gjør den tilfeldige trekkingen om litt...

Eksempel på etterfølgertre

- Etterfølgertegnene registreres i denne rekkefølgen i et tre som initielt er tomt: d b a c f e g h a a d e g g

- Resultat:



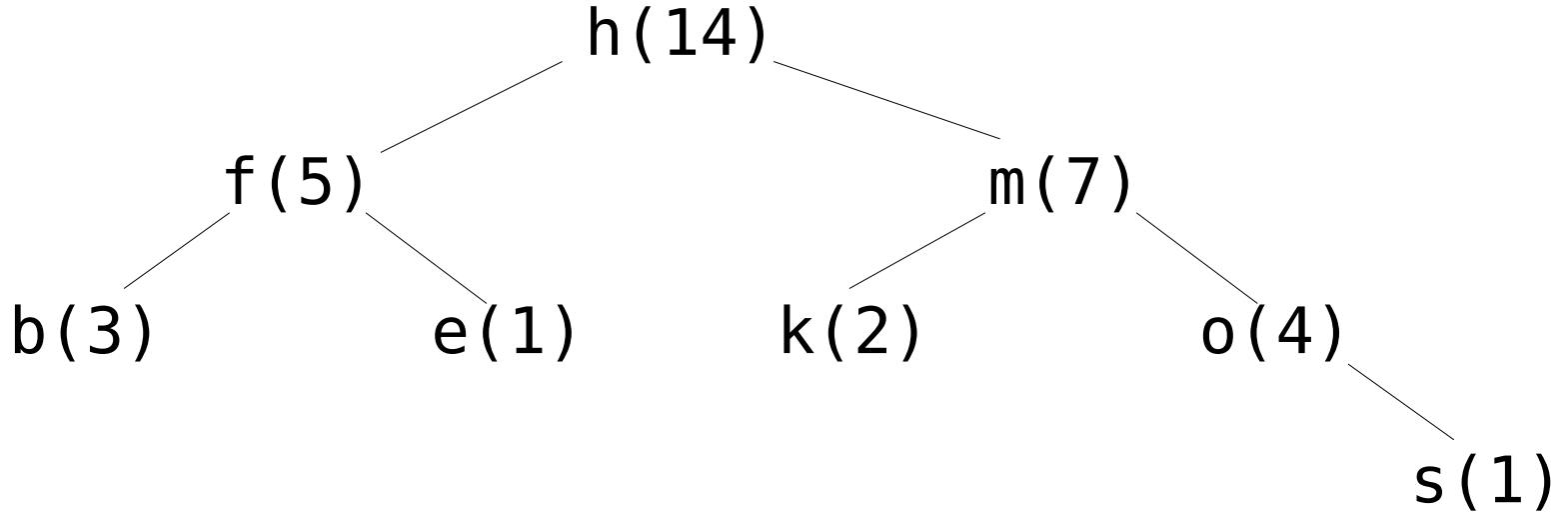
- Merk at antall forekomster av et tegn er lik antallet lagret i noden minus antallene lagret i høyre og venstre subtre

Metode for innsetting i etterfølgertreet

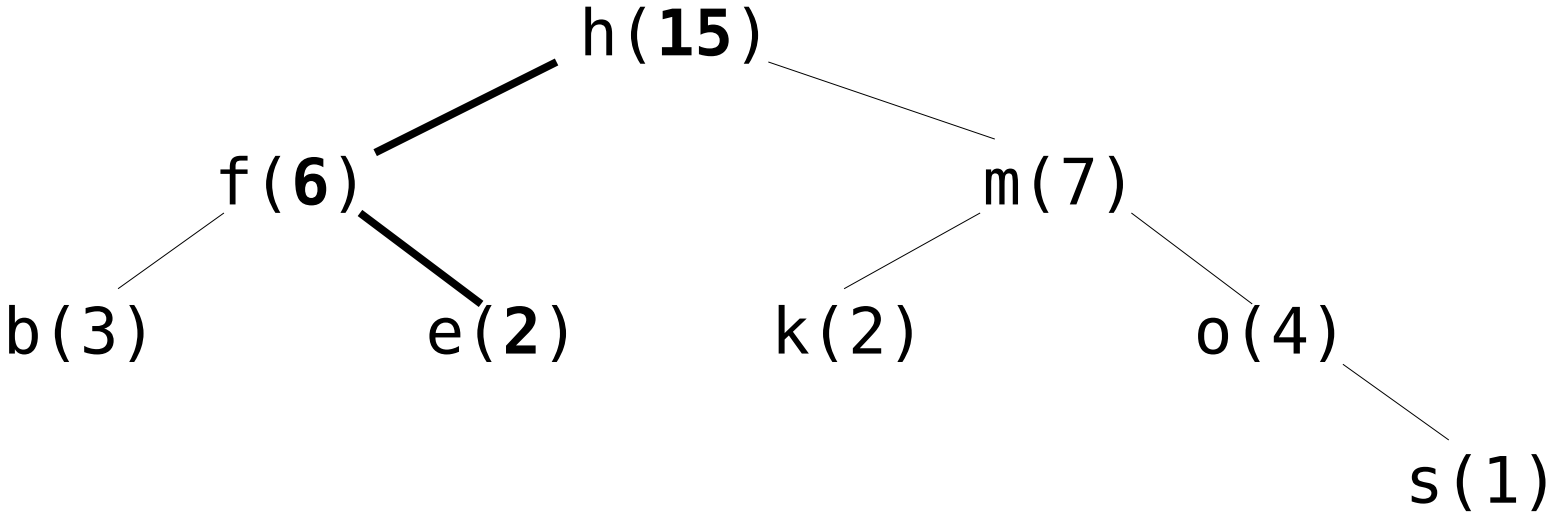
- Registrering av ny forekomst av et etterfølgertegn:
  - Hvis tegnet finnes i treet fra før, økes bare antall forekomster av dette tegnet med 1
  - Hvis tegnet ikke finnes fra før, settes det inn som en bladnode med den vanlige algoritmen for innsetting i binært søketre
- I tillegg må vi oppdatere antall forekomster av tegn i alle de andre nodene som berøres av innsettingen:
  - Gjøres enkelt ved å legge til 1 i alle nodene vi er innom på søkeveien ned til noden som lagrer dette etterfølgertegnet

Eksempel: Setter inn ny forekomst av 'e'

- Før:

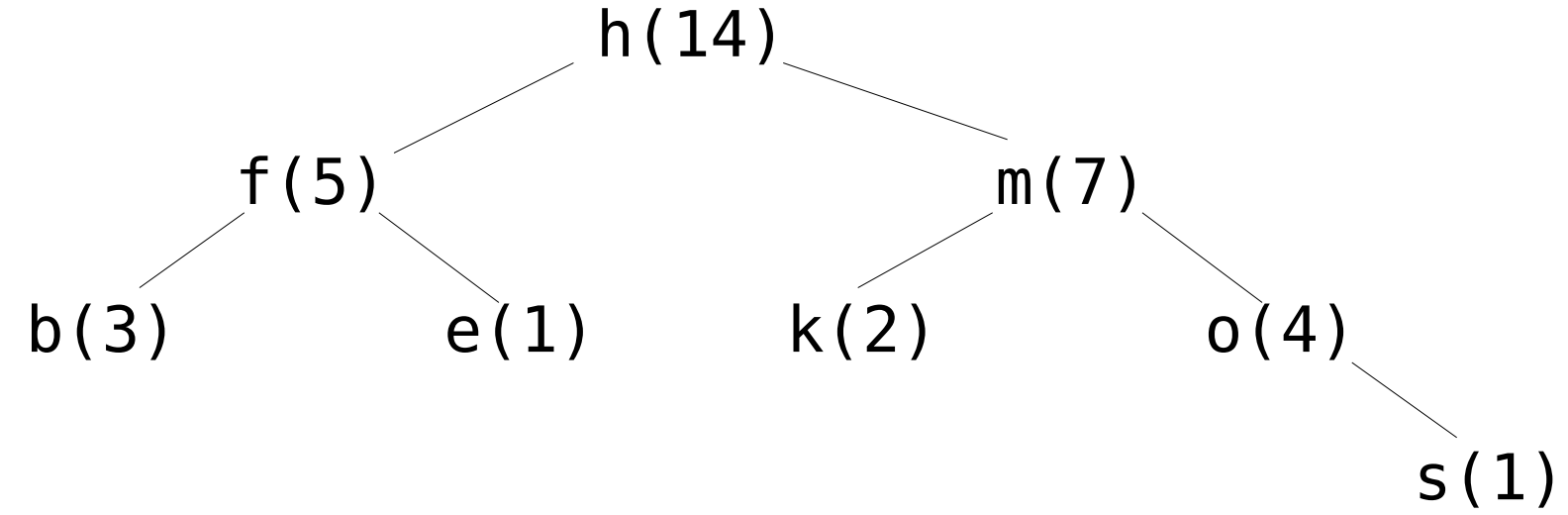


- Etter:

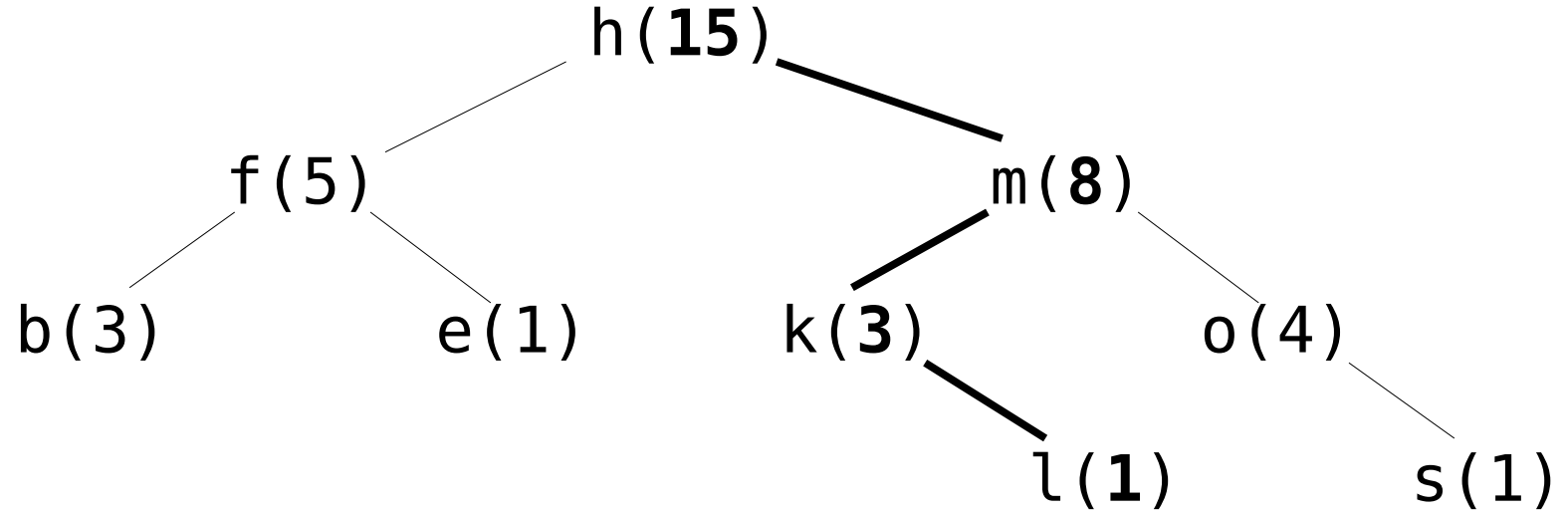


Eksempel: Setter inn ny forekomst av 'l'

- Før:



- Etter:



Etterfølgertre: Implementasjon

- Bruker en egen klasse for et etterfølgertre
- Klassen inneholder en indre klasse for nodene i treet, som lagrer:
  - Selve etterfølgertegnet
  - Totalt antall forekomster i subtreet hvor noden er rot
  - Pekere/referanser til venste og høyre barn
- Metoden for å registrere en ny forekomst av et etterfølgertegn blir nesten lik standardmetoden for innsetting i binært søketre
- Java-kode: `etterfolgerRegister.java`

Registrering av *n*-sekvenser i det «store» sekvenstreet

- Under innlesning registrerer vi hele tiden hver *n*-sekvens som leses *sammen* med sekvensens etterfølgertegn i teksten
- Hvis en lest sekvens ikke finnes fra før, settes den inn som en bladnode med den vanlige algoritmen for innsetting i binært søketre
- Etter at vi har funnet/opprettet sekvensen i det «store» søketreet, registrerer vi en ny forekomst av det leste etterfølgertegnet for denne sekvensen

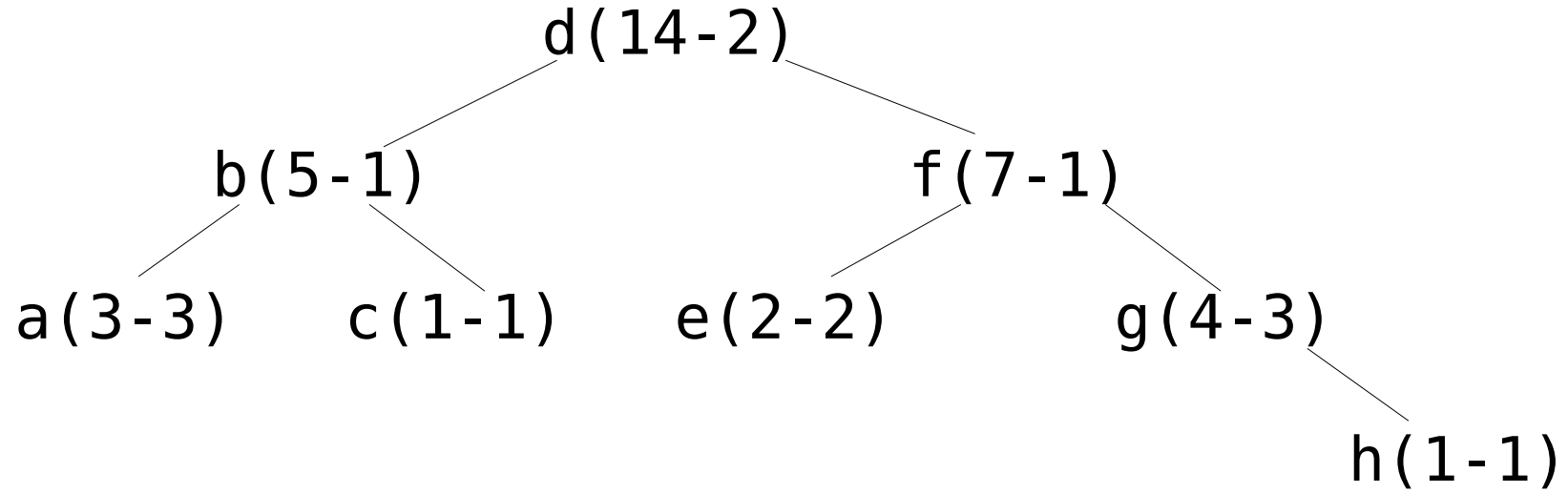
Sekvenstre: Implementasjon

- Bruker en egen klasse for sekvenstreet
- Klassen inneholder en indre klasse for nodene i treet, som lagrer:
  - Selve n-sekvensen
  - En peker til roten i det «lille» søketreet med registrerte etterfølgertegn til denne sekvensen
  - Pekere til venste og høyre barn i sekvenstreet
- Metoden for å registrere ny forekomst av sekvens blir nesten lik standardmetoden for innsetting i et søketre
- Java-kode: `sekvensRegister.java`

Utskrift av tilfeldig svada

- Begynner med å skrive ut den første *n*-sekvensen som ble lest fra input
- Skriver deretter ut ett og ett tegn på denne måten:
  - Søk i det «store» sekvenstreet og finn den siste utskrevne sekvensen
  - Velg tilfeldig et av tegnene som er registrert som etterfølger til denne siste utskrevne *n*-sekvensen, og skriv ut dette tegnet
  - Vi har da skrevet ut en ny *n*-sekvens, og er klar for å velge neste tegn fra *denne* sekvensens etterfølgertre
- Trekkingen av etterfølgertegn gjøres slik at tegnene forekommer med samme frekvenser som i input

Eksempel: Tilfeldig trekking av etterfølgertegn



- Trekk et tilfeldig tall *r*, $1 \leq r \leq 14$:

  Hvis $1 \leq r \leq 5$, velg (rekursivt) et tegn fra venstre subtre

  Hvis $5 < r \leq (14 - 7) = 7$, velg tegnet i roten, d

  Hvis $(14 - 7) = 7 < r \leq 14$, velg (rekursivt) et tegn fra høyre subtre

- Fordeling: a: 1 2 3, b: 4, c: 5, d: 6 7, e: 8 9, f: 10,
  g: 11 12 13, h: 14

Implementasjon av utskrift

- Tilfeldig trekning av etterfølgertegn:
  - Metoden `trekk` i `etterfolgerRegister.java`
- Søk etter n-sekvens og neste tegn som skal skrives ut:
  - Metoden `trekkEtterfolger` i `sekvensRegister.java`
- Formatering av utskriften:
  - Metoden `skrivSvada` i hovedprogrammet `svada.java`

Tekstfiler som kan brukes til testing

- Adolf Hitlers tale i Sportpalast Berlin, 30. januar 1940
- Winston Churchills tale i The House of Commons, London, 18. juni, 1940
- Kong Harald Vs nyttårstale i 2008
- Merk:
  - I disse tre filene er nest siste avsnitt i talen repetert på slutten av filen, for å unngå at vi skriver ut en *n*-sekvens som ikke finnes registrert
  - Filene inneholder spesialtegn som kan gi problemer ved utskrift fra Java-programmet – tegnsettet i output kan evt. transformeres før utskrift