DM551
Mundtlig eksamen – Præsentationer

Jonas Alexander Havstein Eriksen

Indhold

1 Emne 1: Basic counting problems

Intro og overblik over præsentation

- Tælleproblemer er et kerneproblem i kombinatorik. Det handler grundlæggende om at tælle antllet af objekter, der deler en eller flere egenskaber.
- Jeg vil fokusere på binomialsætningen; anvendelse af og induktionsbevis for.
- Hvis tid: Anveldelseseksempel fra ass1.

Binomialsætningen i relation til tælleproblemer

Et basalt tælleproblem er at bestemmt antallet af r-kombinationer fra en mængde S. Mere præcist ønsker man at finde antallet af delmængder $S' \subseteq S$, hvor $|S| = n$. Antallet af r-kombinationer fra en mængde med n unikke elementer er givet ved binomialkoefficienten

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Så hvis vi fx vil udregne antallet af $3-$kombinationer fra en mængde med 10 elementer:

$$\frac{10!}{3!(10-3)!} = \frac{10!}{3!7!} = \frac{10 \cdot 9 \cdot 8}{6} = \frac{720}{6} = 120$$

Binomialkoefficienter er brugbare fx når det kommer til at finde koefficienter i udvidelsen (expansion) af binomiale udtryk opløftet i en heltalspotens, d.v.s. udtryk på formen

$$(x+y)^n$$

hvor x og y er variabler og $n \in \mathbb{N}$. Hvis vi har en høj eksponent n er det besværligt at gange alle leddene ud for at få en koefficient; her kan vi bruge en basal tælleteknik i form af binomialkoefficienter.

Har vi fx $(x+y)^{25}$ og vil finde koefficienten af $x^{12}y^{13}$ kan vi bruge binomialsætningen i stedet for at gange leddene ud.

Binomialsætningen. Lad x og y være variabler og $n \in \mathbb{N}$. Så gælder

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

I forlængelse af eksemplet oven for ser at eksponenterne til x og y passer når $j = 13$, så koefficienten er

$$\binom{25}{13} = \frac{25!}{13!(25-13)!} = \frac{25!}{13!12!}$$

Bevis for binomialsætningen ved induktion over n.

(Lav først base-case, $n = 0$.
Antag da sætningen for et $n \geq 0$.
Brug induktionsantagelsen til at skrive udtrykket for $(x + y)^{n+1}$.
Brug distributive lov til at få to summer.
Sæt nu $k = j + 1$ i den ene sum og tilpas udtrykket hertil.
Træk grænsetilfældet $j = 0$ ud fra ene sum og $k = n + 1$ fra anden sum.
Reducér og kombiner udtryk til én sum med samme faktorer ganget på. Brug hertil Pascal's identitet.
Put grænsetilfælde tilbage i sum.
Færdig).

Base case. $n = 0$.

$(x + y)^0 = \overbrace{\binom{0}{0}}^{=1} x^{0-0}y^0 = 1 = \sum_{j=0}^{0} \binom{n}{j} x^{n-j}y^j$, så det passer.

Induktionstrin. Antag

$$(x + y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j}y^j$$

for et $n \geq 0$. Da får vi

$$
\begin{aligned}
(x+y)^{n+1} &= (x+y)(x+y)^n && (1)\\
&= (x+y)\cdot \sum_{j=0}^{n} \binom{n}{j} x^{n-j}y^j && (2)\\
&= \sum_{j=0}^{n} \binom{n}{j} x^{n-j+1}y^j + \sum_{j=0}^{n} \binom{n}{j} x^{n-j}y^{j+1} && (3)\\
&= \sum_{j=0}^{n} \binom{n}{j} x^{n-j+1}y^j + \sum_{k=1}^{n+1} \binom{n}{k-1} x^{n+1-k}y^k && (4)\\
&= \overbrace{\underbrace{\binom{n}{0}}_{=1} x^{n-0+1}y^0}^{j=0 \text{ fra sum tv.}} + \overbrace{\underbrace{\binom{n}{(n+1)-1}}_{=1} x^{n+1-(n+1)}y^{n+1}}^{k=n+1 \text{ fra sum th.}} + \sum_{j=1}^{n} \binom{n}{j} x^{n-j+1}y^j + \sum_{k=1}^{n} \binom{n}{k-1} x^{n+1-k}y^k && (5)\\
&= x^{n+1} + y^{n+1} + \sum_{k=1}^{n} \left(\binom{n}{k} + \binom{n}{k-1} \right) x^{n-k+1}y^k && (6)\\
&= x^{n+1} + y^{n+1} + \sum_{k=1}^{n} \binom{n+1}{k} x^{n-k+1}y^k && (7)\\
&= \sum_{k=0}^{n+1} \binom{n+1}{k} x^{n+1-k}y^k && (8)
\end{aligned}
$$

(1) Skriver udtrykket ud
(2) Jf. induktionsantagelse.
(3) Jf. den distributive lov for multiplikation. Dette får vi ved at gange hele summen ind i parentesen $(x+y)$; af samme grund inkrementeres eksponenten for x med 1 i venstre udtryk og eksponenten for y med én i højre udtryk.
(4) Lad nu $k = j+1$ så $j = k-1$. Justerer indekser i hø. sum til k.
(5) Vi har et problem i og med at summen tv. summer fra 0 til n og summen tv. summer fra 1 til $n+1$. Så vi trækker $j=0$ ud fra ve. sum og $k = n+1$ ud fra hø. sum.
(6) Vi bemærker at faktoren $x^{n-j+1}y^j$ og $x^{n-k+1}y^k$ er fælles for de to summer fordi der summeres over samme interval, nemlig 1 til n. Derfor disse fælles faktorer trækkes uden for en parentes og de to summer kan kombineres. Derved fås den samlede sum. Bemærk at fordi der summeres fra 1 til n i begge summer er det ligegyldigt, hvad indeksvariablen kaldes; her har vi bare kaldt den for k.
(7) Bruger vi Pascal's identitet (Thm. 6.4.2): $\binom{k}{k-1} + \binom{n}{k} = \binom{n+1}{k}$.
(8) Inkorporerer de to grænsetilfælde som vi fik i (5) og (6). Disse er henholdsvis resultatet for $k=0$ og $k = n+1$, så summens rækkevidde udvides fra 1 til n til fra 0 til $n+1$.

□

Eksempel på anvendelse (fra ass1), hvis tid til overs

Vi ønsker at finde koefficienten til x^{95} i det binomiale udtryk $(x+3)^{189}$. We note that $(x+3)^{189}$ is a binomial and hence we can use the Binomial Theorem to find the coefficient of x^{95}.

V.h.a. binomialsætningen får vi

$$(x+3)^{189} = \sum_{j=0}^{189} \binom{189}{j} \cdot 1^{189-j} \cdot 3^j$$

Herfra ser vi at vi får x^{95} når $j = 189 - 95 = 94$. Derfor er koefficienten til x^{95}

$$\binom{189}{94} \cdot 1^{95} \cdot 3^{94} = \binom{189}{94} \cdot 3^{94}$$

2 Emne 2: Inclusion-exclusion with applications

Intro og overblik over præsentation

- Princippet om inklusion-eksklusion handler om at tælle antallet af elementer i foreningsmængden af to eller flere mængder. Nyttigt princip fordi overcounting nemt forekommer.
- Jeg fik først præsentere princippet for inklusion-eksklusion (Thm. 8.5.1 i Rosen) og dernæst bevise dette.
- Som anvendelseseksempel vil jeg vise, hvordan man kan bruge princippet til at tælle antallet af surjektive (onto) funktioner mellem to mængder.

Princippet for inklusion-eksklusion

Lad A og B være to endelige mængder. Så er

$$|A \cup B| = |A| + |B| - |A \cap B|$$

hvilket giver intuitivt god mening (understøt med Venn-diagram!). Det er altså vigtigt at fratrække størrelsen af foreningsmængde, fordi ellers tælles alle elementer heri 2 gange $\Rightarrow$ overcount.

Men hvad så når vi ønsker at finde $|A_1 \cup A_2 \cup \cdots \cup A_n|$, hvor $A_1, A_2, \ldots, A_n$ er endelige mængder? Her kan vi bruge en generalisering af ovenstående observationer omkring $|A \cup B|$.

Princippet for inklusion-eksklusion (Thm. 8.5.1). Lad $A_1, A_2, \ldots, A_n$ være endelig mængder. Så er

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{1 \leq j \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots A_n|$$

Bevis for princippet for inklusion-eksklusion

(Antag at et element a eksisterer i præcis r af mænderne A_i. Argumenter for at a tælles $\binom{r}{m}$ gange når der indgår m A_i'er i summeringen. Brug dette til at være at a tælles præcis én gang.)

Antag for et vilkårligt element a, at der $\exists! r \leq n : a \in A_i$ for r værdier af i. Da må det være tilfældet at

$$\sum_{1\leq i\leq n} |A_i| \Rightarrow a \text{ tælles } \binom{r}{1} = r \text{ gange}$$

fordi a indgår i præcis r af mængderne A_i. Desuden

$$\sum_{1\leq i<j\leq n} |A_i \cap A_j| \Rightarrow a \text{ tælles } \binom{r}{2} \text{ gange}$$

fordi da a indgår i r forskellige A_i indgår den i alle par af fællesmængder herimellem. Da vi har indeks $1 \leq i < j \leq n$ sikrer vi at alle fællesmængde-par $A_i \cap A_j$ indgår præcis én gang. Så det svarer til at vi vælger to mængder A_i og A_j for en mængde bestående af r mængder. Vi generaliserer til

$$\sum_{1\leq i<j<\ldots<m\leq n} |A_i \cap A_j \cap \cdots \cap A_m| \Rightarrow a \text{ tælles } \binom{r}{m} \text{ gange}$$

Dette kan vi bruge til at afgøre, hvor mange gange a tælles alt i alt. Ved at benytte højresiden af lighedstegnet i teoremet for inklusion-eksklusion får vi at a tælles følgende antal gange

$$\binom{r}{1} - \binom{r}{2} + \binom{r}{3} - \cdots + (-1)^{r+1}\binom{r}{r}$$

Ved at bruge korollar 6.4.2 der siger os at

$$\sum_{k=0}^{n}(-1)^k\binom{n}{k} = 0$$

som anvendt på vores udregning givet

$$0 = \binom{r}{0} - \binom{r}{1} + \binom{r}{2} - \cdots + (-1)^r\binom{r}{r} \quad \Leftrightarrow \tag{9}$$

$$1 = \binom{r}{0} = \binom{r}{1} - \binom{r}{2} + \binom{r}{3} - \cdots + (-1)^{r+1}\binom{r}{r} \tag{10}$$

Udtrykket i (9) får vi ved at skrive summen af binomialkoefficienter op som korollar 6.4.2 foreskriver. (10) får vi ved først at isolere $\binom{r}{0}$ ved at hive alle andre led over på den anden side af lighedstegnet og ændres deres fortegn. Dernæst bemærker vi at $\binom{r}{1} = 1$. Vi skifter også $(1)^r$ til $(-1)^{r+1}$ fordi vi skiftede fortegn på alle led, så det sidste led skal også skifte fortegn.

Da a var et vilkårligt element i r mængder, har vi vist at et element i foreningsmængden mellem et vilkårligt antal tælles præcis én gang.

□

Anvendelseseksempel: Antal surjektive funktioner

Vi ønsker at finde antallet af surjektive (onto) funktioner fra mængden A til mængden B, hvor $|A| = 7$ og $|B| = 3$. Vi kan definere en surjektiv funktion $f : A \to B$ som

$$\forall y \in B : \exists x \in A : (f(x) = y)$$

Lad $b_1, b_2, b_3 \in B$ være elementerne i B (sekundærmængden). Lad P_1, P_2, P_3 være egenskaberne at hhv. b_1, b_2, b_3 ikke mappes til at noget element i A (definitionsmængden). Vi bemærker at en funktion er surjektiv hvis og kun hvis den ikke har nogen af egenskaberne P_1, P_2, P_3, d.v.s. har alle egenskaberne P_1', P_2', P_3'.

Ved at bruge princippet for inklusion-eksklusion må antallet af surjektive funktioner $f : A \to B$ være

$$\begin{aligned} N(P_1', P_2', P_3') =& N - [N(P_1) + N(P_2) + N(P_3)] + [N(P_1P_2) + N(P_1P_3) + N(P_2P_3)] \\ & - N(P_1P_2P_3) \end{aligned}$$

hvor N er det samlede antal surjektive funktioner fra $A \to B$ og $N(P_i)$ er antallet af funktioner der ikke mapper til b_i.

- $N = 3^7$ fordi alle elementer i A kan vælge mellem alle tre elementer i B.
- $N(P_i) = 2^7$ fordi der er ét b_i der ikke mappes til så hvert element i A kan mappe til 2 værdier i B. Der er $\binom{3}{1} = 3$ af disse led.
- $N(P_iP_j) = 1^7$ fordi b_i og b_j ikke mappes til. Der er $\binom{3}{2} = 3$ af disse led.
- $N(P_1P_2P_3) = 0$ fordi det per definition ikke er en funktion, hvis elementerne i A ikke mappes til sekundærmængden. Der er $\binom{3}{3} = 1$

Så alt i alt:

$$N(P_1'P_2'P_3') = 3^7 - 2^7 \cdot \binom{3}{1} + 1^7\binom{3}{2} - 0\binom{3}{3} = 3^7 \cdot 3 - 2^7 \cdot 3 + 1 \cdot 3 = 1806$$

Det viser sig at vi kan generalisere dette løsningsmønster til hvis $|A| = m$ og $|B| = n$. Antallet af surjektive funktioner $f : A \to B$ er i så fald

$$n^m - (n-1)^m \cdot \binom{n}{1} + (n-2)^m \cdot \binom{n}{2} - \cdots + (-1)^{n-1}1^m \cdot \binom{n}{n-1}$$

fordi det sidste led, d.v.s. det med $\binom{n}{n}$ altid er 0 jf. definitionen af en funktion. (Ovenstående er Thm. 8.6.1. p. 544).

3 Emne 3: Recurrence relations

Intro og overblik over præsentation

- Jeg vil fokusere på lineære ikke-homogene rekursionsligninger med konstante koefficienter.
- Først definere denne type rekursionsligninger.
- Dernæst give løsningsmetode og bevise at denne er korrekt.
- Dernæst: Anvendelseseksempel.
- Til sidst: Demonstrere iterativ løsning af rekursionsligning for Tower of Hanoi

Definition

En lineær ikke-homogen rekursionsligning med konstante koefficienter har formen

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots c_k a_{n-k} + F(n)$$

hvor $c_1, c_2, \ldots, c_k \in \mathbb{R}$ og $\exists n : F(n) \neq 0$. Ikke-homogeniteten består således i at $F(n)$ ikke er en linearkombination af forrige led.

Løsningsmetode

For at løse en lineær, ikke-homogen rekursionsligning med konstante koefficienter a_n skal man løse dens forbundne lineære, homogene rekursionsligning (d.v.s. a_n bare uden $F(n)$) samt finde en partikulær løsning for hele den ikke-homogene rekursionsligning. Mere formelt:

Thm. Hvis $\{a_n^{(p)}\}$ er en partikulær løsning til den ikke-homogene rekursionsligning med konstante koefficienter

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n)$$

så har alle løsninger formen $\{a_n^{(p)} + a_n^{(h)}\}$, hvor $a_n^{(h)}$ er en løsning til den forbundne homogene rekursionsligning.

Bevis Da $a_n^{(p)}$ er en løsning til hele den ikke-homogene rekursionsligning ved vi at

$$a_n^{(p)} = c_1 a_{n-1}^{(p)} + c_2 a_{n-2}^{(p)} + \cdots + c_k a_{n-k}^{(p)} + F(n)$$

Antag nu at $\{b_n\}$ er endnu en løsing til den ikke-homogene rekursionsligning, d.v.s.

$$b_n = c_1 b_{n-1} + c_2 b_{n-2} + \cdots - c_k b_{n-k} + F(n)$$

Hvis vi beregner $b_n - a_n^{(p)}$ får vi

$$b_n - a_n^{(p)} = c_1(b_{n-1} - a_{n-1}^{(p)}) + c_2(b_{n-2} - a_{n-2}^{(p)}) + \cdots + c_k(b_{n-k} - a_{n-k}^{(p)})$$

Fordi de to $F(n)$'er går ud med hinanden. Det følger at $\{b_n - a_n^{(p)}\}$ er en løsning til den forbundne homogene rekursionsligning, d.v.s. $\{b_n - a_n^{(p)}\} = \{a_n^{(h)}\}$. Dette medfører at

$$a_n^{(h)} = b_n - a_n^{(p)} \;\Leftrightarrow\; b_n = a_n^{(h)} + a_n^{(p)}$$

Da vi definerede b_n til at være en løsning til hele den ikke-homogene rekursionsligning har vi vist at løsningen er givet ved $\{a_n^{(h)} + a_n^{(p)}\}$.

□

Anvendelseseksempel

Vi har en helt fast metode til at finde en partikulær løsning når $F(n)$ er produktet af et polynomium i n og en konstant i n'te. Så i stedet for at demonstrere det, vil jeg vise, hvordan man kan anvende denne type rekursionsligninger til at løse en datalogisk problemstilling. Problemet handlet om at tælle antal gyldige kodeord og eksemplet svarer til eksempel 8.1.4 i Rosen (s. 490).

Problem. Et computsystem betragter en streng af tal som et gyldigt kodeord hvis og kun hvis det indeholder et lige antal 0'er.

Løsning. Lad a_n være antallet af gyldige kodeord af længde n.

Vi kan nemt indse, at $a_1 = 9$, fordi alle tallene $i \in \{1, 2, 3, \ldots, 9\}$ udgør gyldige kodeord af længde 1. 0 er ikke et gyldigt kodeord, fordi det 1 – og dermed et ulige antal – 0'er i sig.

Vi bemærker at vi kan finde a_n ved at betragte a_{n-1}. For alle de gyldige kodeord af længde $n-1$ – d.v.s. a_{n-1} – kan vi lave et nyt gyldigt kodeord ved at konkatanere strengen af længde $n-1$ med et tal $i \neq 0$. Dette kan gøres på 9 måder. Så indtil videre kan vi sige at $a_n = 9a_{n-1}$.

Men vi kan også danne et gyldigt kodeord af længde n ud fra et ugyldigt kodeord af længde $n-1$ ved at konkatanere det ugyldige kodeord med et 0. Da går det fra et have et ulige til et lige antal 0'er. Der eksisterer præcis 10^{n-1} strenge af længde $n-1$ og a_{n-1} af disse er gyldige jf. ovenstående. Så der må være $10^{n-1} - a_{n-1}$ ugyldige der alle kan gøres gyldige med ovenstående metode.

Så alt i alt er antallet af gyldige kodeord af længde n givet ved

$$a_n = 9a_{n-1} + (10^{n-1} - a_{n-1}) = 8a_{n-1} + 10^{n-1}$$

hvilket er en lineær, ikke-homogen rekursionsligning med konstante koefficienter.

Iterativ løsning af Tower of Hanoi

Tower of Hanoi er et spil, hvor man skal flytte n skiver med unikke størrelser fra én pind til en anden, hvor hvert træk indebærer rykning af én skive og ingen skive må på noget tidspunkt være ovenpå en skive af mindre størrelse. Lad H_n være antallet træk det kræves for at løse Tower of Hanoi med n skiver. Det viser sig at $H_n = 2H_{n-1} + 1$, hvor $H_1 = 1$.

$H_n = 2H_{n-1} + 1$ er en lineær, ikke-homogen rekursionsligning med konstante koefficienter. Som et alternativ til at finde løsningen til den associerede rekursionsløsning samt en partikulær løsning, kan man løse ligningen iterativt. Dvs. ved gentagne gange at udtrykke H_n vha. forrige led for til sidst at nå startbetingelsen $H_1 = 1$. Observer følgende (linjer med * er der, hvor et led udtrykkes ved et forrige led, og de øvrige linjer er mellemregninger)[1]:

$$\begin{aligned}
H_n =& 2H_{n-1} + 1 \\
=& 2(2H_{n-2} + 1) + 1 && (*) \\
=& 2^2 H_{n-2} + 2 + 1 \\
=& 2^2(2H_{n-3} + 1) + 2 + 1 && (*) \\
=& 2^3 H_{n-3} + 2^2 + 2 + 1 \\
=& 2^3(2H_{n-4} + 1) + 2^2 + 2 + 1 && (*) \\
=& 2^4 H_{n-4} + 2^3 + 2^2 + 2 + 1 \\
& \vdots \\
=& 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\
=& 2^n - 1
\end{aligned}$$

[1] Induktionsbevis for at løsningen er korrekt. Base case: $H_1 = 1 = 2^1 - 1$. Induktionstrin: $H_{n+1} = 2H_n + 1 \overset{IH}{=} 2(2^n - 1) + 1 = 2^{n+1} - 1$. $\square$

4 Emne 4: Discrete probability, random variables, and bounds

Intro og overblik over præsentation

- Jeg vil fokusere på Bayes' Teorem.
- Først vil jeg præsentere teoremet, hvorefter jeg vil bevise det.
- Afslutningsvis vil jeg give et anvendelseseksempel, der omhandler spam-filtre.

Hvad er Bayes' Teorem?

Bayes' Teorem er en sætning vi kan bruge til at udregne en betinget sandsynlighed når vi har nogle informationer relateret til udfaldet. Hvis vi fx kender sandsynligheden for at et givet ord indgår i en spam-mail og sandsynligheden for at den indgår i en ikke-spam-mail samt sandsynligheden for at en indkommende mail er spam, så vi udregne sandsynligheden for at en indkommende mail er spam givet at den indeholde det pågældende ord.

Bayes Teorem. Lad E og F være udfald fra et udfaldsrum S, hvor $p(E) \neq 0$ og $p(F) \neq 0$. Da gælder

$$p(F|E) = \frac{p(E|F) \cdot p(F)}{p(E|F) \cdot p(F) + p(E|\overline{F}) \cdot p(\overline{F})}$$

Bevis

(Brug betinget sandsynlighed til at udtrykke $p(E|F)$ og $p(F|E)$.
Sæt udtryk lig hinanden for forkort med $p(E)$.
Vis at $p(E) = p(E|F) \cdot p(F) + p(E|\overline{F}) \cdot p(\overline{F})$.
Indsæt dette udtryk for $p(E)$ i $p(F|E) = \frac{p(E|F) \cdot p(F)}{p(E)}$ og derefter færdig.)

Vi ved fra definitionen af betinget sandsynlighed (def. 7.2.3) at

$$p(E|F) = \frac{p(E \cap F)}{p(F)} \Leftrightarrow p(E \cap F) = p(E|F) \cdot p(F)$$

og

$$p(F|E) = \frac{p(F \cap E)}{p(E)} \Leftrightarrow p(F \cap E) = p(F|E) \cdot p(E)$$

Når vi ssætter disse udtryk lig med hinanden og forkorter med $p(E)$ får vi

$$p(F|E) \cdot p(E) = p(E|F) \cdot p(F) \;\Leftrightarrow\; p(F|E) = \frac{p(E|F) \cdot p(F)}{p(E)} \tag{11}$$

Nu vil vi vise at $p(E) = p(E|F) \cdot p(F) + p(E|\overline{F}) \cdot p(\overline{F})$:

Vi observerer at

$$\begin{aligned} E =& E \cap S && \text{(Da } S \text{ er udfaldsrummet)} \\ =& E \cap \overbrace{(F \cup \overline{F})}^{=S} \\ =& (E \cap F) \cup (E \cap \overline{F}) && \text{(Jf. distributive lov)} \end{aligned}$$

Nu vil vi finde $p(E)$. Det gør vi ved at tage sandsynligheden på begge sider af lignedestegnet af ovenstående udtryk for E. Da får vi

$$\begin{aligned} p(E) =& p((E \cap F) \cup (E \cap \overline{F})) \\ =& p(E \cap F) + p(E \cap \overline{F}) - p((E \cap F) \cap (E \cap \overline{F})) && \text{(Iflg Thm. 7.1.2)} \\ =& p(E \cap F) + p(E \cap \overline{F}) && \text{(Se nedenfor)} \end{aligned}$$

Den sidste omskrivning er gyldig da

$$(E \cap F) \cap (E \cap \overline{F}) = \emptyset$$

fordi

$$(x \in (E \cap F) \wedge x \in (E \cap \overline{F})) \;\Rightarrow\; x \in (F \cap \overline{F}) = \emptyset \qquad \text{(Da fællesmængder)}$$

Da vi allerede har vist at $p(E \cap F) = p(E|F) \cdot p(F)$ og nemt ser at $p(E \cap \overline{F}) = p(E|\overline{F}) \cdot p(\overline{F})$ får vi at

$$\begin{aligned} p(E) =& p(E \cap F) + p(E \cap \overline{F}) && \text{(Fra forrige omskrivning)} \\ =& p(E|F) \cdot p(E) + p(E|\overline{F}) \cdot p(\overline{F}) \end{aligned}$$

Nu mangler vi blot at indsætte udtrykket for $p(E)$ i (11), hvilket giver os

$$\begin{aligned} p(F|E) =& \frac{p(E|F) \cdot p(F)}{p(E)} && \text{(Resultat fra eq. 11)} \\ =& \frac{p(E|F) \cdot p(F)}{p(E|F) \cdot p(E) + p(E|\overline{F}) \cdot p(\overline{F})} \end{aligned}$$

□

Anvendelseseksempel: Spam-filtre

Vi ønsker at udvikle et simpelt spam-filter, der benytter Bayes' Teorem til at komme med et kvalificeret gæt på om en e-mail er spam eller ikke spam givet, hvilke ord, der indgår i den.

Antag at vi har to mængder af e-mails, B og G, hvor B er spam-mails og G er ikke-spam-mails. Lad W være mængden af ord der indgår i mindst én meddelelse i både B og G. Nu finder vi antal meddelelser i B og G der indholder $w \in W$ og får $n_B(w)$ og $n_G(w)$. Den empiriske sandsynlighed for at en spam-mail $b \in B$ indeholder $w \in W$ er således $p(w) = \frac{n_B(w)}{|B|}$. Og den empiriske sandsynlighed for at en ikke-spam-meddelelse indeholder w er $q(w) = \frac{n_G(w)}{|G|}$.

Antag at vi nu modtager en e-mail e der indeholder ordet w. Lad E være udfaldet af e indeholder w. Lad S være udfaldet at e er spam. Så $\overline{S}$ er udfaldet at e ikke er spam. Da $S \cup \overline{S}$ udgør hele udfaldtrummet kan vi bruge Bayes' sætning til at finde sandsynligheden for at e er en spam email (vi antager at $p(S) = p(\overline{S}) = 1/2$):

$$\begin{aligned} p(S|E) =& \frac{p(E|S) \cdot p(S)}{p(E|S) \cdot p(S) + p(E|\overline{S}) \cdot p(\overline{s})} \\ =& = \frac{1/2(p(E|S)}{1/2(p(E|S) + p(E|\overline{S}))} \qquad \text{(Hvis vi antager } p(S) = p(\overline{S}) = 1/2\text{)} \\ =& \frac{p(E|S)}{p(E|S) + p(E|\overline{S})} \end{aligned}$$

Vi ser at $p(E|S)$ er sandsynligheden for at e indeholder w givet at e er spam. Dette svarer til $p(w)$. Vi ser desuden at $p(E|\overline{S})$ er sandsynligheden for at e indeholder w givet at e ikke er spam. Dette svarer til $q(w)$. Så hvis vi lader $r(w)$ være sandsynligheden $p(S|E)$ så får vi

$$r(w) = \frac{p(w)}{p(w) + q(w)}$$

hvilket er et basalt spam filter. Vi kan nu vælge en grænseværdi for $r(w)$, der afgør, hvornår en indkommende meddelelse e ryger i spamfilteret og hvornår den ryger i indbakken. En sådan grænseværdi kunne fx være at e kommer i spamfilteret hvis $r(w) > 0,9$.

5 Emne 5: Randomized algorithms

Intro og overblik over præsentation

- Jeg vil fokusere på min-cut i grafer
- Først vil jeg præsentere algoritmen og bevise en nedre grænse på sandsynligheden for at algoritmen returnerer et globalt min-cut
- Til sidst vil jeg vise en øvre grænse for antallet af globale min-cuts i en uorienteret graf (hvis der er tid)

Global min cut i graf: Problemet

Givet en uorienteret graf $G = (V, E)$, så er et cut en partitionering af V til to ikke-tomme mængder A og B. For et cut (A, B), så er størrelsen af (A, B) lig med antallet af kanter med ét endepunkt i A og det andet i B. Et globalt min-cut er et cut (A, B) med mindst mulig størrelse.

Det vil sige at størrelsen på et globalt min-cut er det mindst antal kanter der, hvis de fjernes, resulterer i at G ikke længere er sammenhængende.

The Contraction Algorithm

Lad $G = (V, E)$ være en uorienteret multigraf. Lad $S(v) = \{v\}, \forall v \in V$. Vælg en $e \in_R E$. Kontraktér e og lad G' være resultatet, hvor Z_{uv} erstatter de kontrakterede u og v fra G. Så $Z_{uv} = S(u) \cup S(v)$. Fortsæt rekursivt indtil $V = \{A, B\}$, hvor cuttet $(S(A), S(B))$ returneres.

Thm. The Contraction Algorithm returnerer et globalt min-cut af G med sandsynlighed mindst $1/\binom{n}{2}$, hvor $n = |V|$.

Bevis. Lad k være størrelsen på et globalt min-cut (A, B) i G og lad F være et vilkårligt sådant globalt min-cut. D.v.s. F indeholder k kanter med ét endepunkt i A og et andet i B.

Vi ønsker at finde en nedre grænse for sandsynligheden for, at algoritmen returnerer F. Dette sker i det tilfælde, hvor ingen kanter $e \in F$ kontrakteres i løbet af de $n - 2$ rekursive kald. Fordi hvis en kant $\in F$ blev kontrakteret, ville de to knuder u og v den forbandt blive 'slået sammen' i en superknude Z_{uv}, som ville havne i enten A eller B og så ville (A, B) ikke længere være et cut.

Så første trin er at finde sandsynligheden $p(e \in F \text{ kontrakteres})$. Observér følgende modstrid vedrørende antal knuder i G:

$$\exists v \in V : \ \deg(v) < k \Rightarrow \text{størrelsen af cuttet } (\{v\}, V - \{v\}) < k$$

hvilken er en modstrid ift. antagelsen om at (A, B) af størrelse k er et globalt min cut. Så vi kan slutte at $\forall v \in V : \deg(v) \geq k$. Vi kan nu bruge Thm. 10.2.1 i Rosen[2] til at indse at

$$2|E| = \sum_{\forall v \in V} \deg(v) \geq k \cdot n \;\Leftrightarrow\; |E| \geq \frac{1}{2} \cdot k \cdot n$$

Så

$$p(e \in F \text{ kontrakteres}) = \frac{k}{|E|} \leq \frac{k}{\frac{1}{2}kn} = \frac{2k}{kn} = \frac{2}{n}$$

fordi kanten der kontrakteres vælges uniformt tilfældt blandt alle kanter.

Nu betragter vi situationen efter j rekursive kald, d.v.s. når der eksisterer $n-j$ superknuder i den nuværende graf G'. Antag at ingen kanter $e \in F$ er blevet kontrakteret i disse j rekursive kald. Eftersom at ethvert cut af G' er et cut af G (altså den oprindelige graf), så har alle knuder i G' grad mindst k (følger af modstrid helt analog til ovenstående). Så G' har mindst $\frac{1}{2}k(n-j)$ kanter. Det følger at sandsynligheden $p(e \in F$ kontrakteres i rekursive kald $j+1)$ *givet at ingen kanter $e \in F$ blev er blevet kontrakteret i de forrige j kald* er lig med

$$\frac{k}{|E'|} \leq \frac{k}{\frac{1}{2}k(n-j)} = \frac{2k}{k(n-j)} = \frac{2}{n-j}$$

Vi husker at F returneres i præcis det tilfælde, hvor ingen kanter $e \in F$ kontrakteres i de $n-2$ rekursive kald. Lad X_j være udfaldet at det i det $j$te kald *ikke* er en kant fra F der kontrakteres (altså komplementerne til de ovenstående sandsynligheder). Vi følger det at ovenstående resultater at $p(X_1) \geq 1 - \frac{2}{n}$ og at $p(X_{j+1}|X_1 \cap X_2 \cap \cdots \cap X_j) \geq 1 - \frac{2}{n-j}$. Vi ønsker nu at finde sandsynligheden $p(X_1 \cap X_2 \cap \cdots \cap X_{n-2})$, altså sandsynligheden for at ingen $e \in F$ returneres i de $n-2$ rekursive kald.

$$\begin{aligned}
p(X_1 \cap X_2 \cap \cdots \cap X_{n-2}) =& p(X_1) \cdot p(X_2|X_1) \cdot p(X_3|X_1 \cap X_2) \cdots p(X_{n-2}|x_1 \cap X_2 \cap \cdots \cap X_{n-3}) \\
\geq& \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-j}\right) \cdots \left(1 - \frac{2}{n-(n-3)}\right) \\
=& \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right) \cdot \left(\frac{n-4}{n-2}\right) \cdots \left(\frac{1}{3}\right) \\
=& \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}
\end{aligned}$$

Omskrivningen til sidste linje kommer af at alle andre led går ud med hinanden.

□

[2] 'The Handshake Theorem'

Dette er en sandsynlighed tæt på 0, altså er der en meget lav sandsynlighed for at algoritmen rent faktisk returnerer et globalt min-cut.

Vi kan forbedre sandsynligheden ved at lade algoritmen køres $\binom{n}{2}$ gange. Da vil sandsynligheden for at vi *ikke* finder et globalt min-cut i én af kørslerne være

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leq \frac{1}{e}$$

jf. fact (13.1) i Kleinberg & Tardos s. 711. Dette er altså en markant bedre sandsynlighed.

Antal globale min-cuts i en graf

Jeg vil nu præsentere og vise et resultat der ligger i umiddelbar forlængelse af ovenstående.

Thm. En uorienteret graf $G = (V, E)$, hvor $|V| = n$ har højest $\binom{n}{2}$ globale min cuts.

Bevis. Lad $G = (V, E)$ være en uorienteret graf og lad $C_1, C_2 \ldots C_r$ være alle globale min-cuts af G. Så r er altså antallet af globale min-cuts i G.

Lad T_i være udfaldet at algoritmen returnerer C_i og lad $T = \bigcup_{i=1}^{r} T_i$ være udfaldet at et hvilket som helst min cut returneres af algoritmen. I forrige bevis viste vi at

$$p(T_i) \geq \frac{1}{\binom{n}{2}}$$

da F var et vilkårligt min-cut, d.v.s. $\exists i : F = C_i$. Vi bemærker at T_i og T_j er disjunkte udfald fordi der kun returneres højest ét globalt min-cut i en kørsel af algoritmen. Så v.h.a. Union Bound for parvist disjunkte udfald (s. 772) får vi

$$p(T) = \left(\bigcup_{i=1}^{r} T_j\right) = \sum_{i=1}^{r} p(T_i) \geq r \cdot \frac{1}{\binom{n}{2}} = \frac{r}{\binom{n}{2}}$$

Og da det selvfølgelig gælder at $p(T) \leq 1$ har vi at $r \leq \binom{n}{2}$. Altså eksisterer der højest $\binom{n}{2}$ globale min-cuts i G.

□

6 Emne 6: The probabilistic method, Monte Carlo algorithms, the Birthday Problem

Intro og overblik over præsentation

- Jeg vil fokusere på den probabilistiske metode og The Birthday Problem
- Først vil jeg bruge den probabilistiske metode til at bevise et Teorem omkring Ramsey-tal.
- Til sidst vil jeg præsentere The Birthday Problem, hvis der er tid.

Den probabilistiske metode

Den probabilistiske metode er en metode til at lave ikke-konstruktive eksistensbeviser. Konkret siger den, at hvis sandsynligheden for at et element, der er tilfældigt udvalgt fra en mængde S ikke har en bestemt egenskab er < 1, så eksisterer der et element i S med den pågældende egenskab.

Anvendelse af den probabilistiske metode

Jeg vil benytte den probabilistiske metode til at bevise et teorem vedr. Ramsey-tal. Ramsey-tallet $R(n, m)$, hvor $n, m \in \mathbb{Z}^+, n, m \geq 2$ angiver det mindst antal mennesker til en fest, således at der er n indbyrdes venner eller m indbyrdes fjender og hvor hvert par af personer er enten venner eller fjender.

Thm. Hvis k er et heltal med $k \geq 2$ så er $R(k, k) \geq 2^{k/2}$.

Bevis Vi tager $R(2, 2) = 2$ og $R(3, 3) = 6$ for givet[3] og bemærker at $2 \geq 2^{2/2} = 2$ og $6 \geq 2^{3/2} \approx 2,88$. Antag desuden for alle par af mennesker r, s at $p(r \text{ og } s \text{ venner}) = p(r \text{ og } s \text{ fjender}) = 1/2$.

Antag at der er n personer til en fest. Så er der $\binom{n}{k}$ forskellige mængder bestående af k personer ud af de n personer. Disse mængder kalder vi S_i for $i = 1, 2, \ldots, \binom{n}{k}$.

Lad E_i være udfaldet at alle personerne $k \in S_i$ enten alle er indbyrdes venner eller alle indbyrdes fjender. Så har vi at sandsynligheden

$$p(\exists k \text{ indbyrdes venner/fjender}) = p\left(\bigcup_{i=1}^{\binom{n}{k}} E_i\right)$$

[3] $R(2, 2) = 2$ er trivielt sandt eftersom at, hvis vi betrager 2 mennesker vil disse enten være venner eller fjender; altså 2 inbyrdes venner eller 2 indbyrdes fjender. Udledningen af at $R(3, 3) = 6$ kan findes i eksempel 6.2.13, p. 393.

Vi bemærker at der er

$$\binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2}$$

par af personer i alle S_i, fordi $|S_i| = k$, $\forall i$. Og eftersom at vi antog at det er lige sandsynligt at ethvert par af personer er venner eller fjender samt at alle par er enten venner eller fjender, så har vi følgende sandsynlighed

$$\begin{aligned} p(k \text{ personer i } S_i \text{ er venner}) =& p(k \text{ personer i } S_i \text{ er fjender}) \\ =& \overbrace{\left(\frac{1}{2}\right) \cdot \left(\frac{1}{2}\right) \cdots \left(\frac{1}{2}\right)}^{\binom{k}{2} \text{ times}} \\ =& \left(\frac{1}{2}\right)^{\binom{k}{2}} \\ =& \left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}} \end{aligned}$$

Så eftersom at E_i er udfaldet at alle k personer i S_i er indbyrdes venner *eller* indbyrdes fjender har vi iflg. sum-reglen at

$$p(E_i) = p(k \text{ personer i } S_i \text{ er venner}) + p(k \text{ personer i } S_i \text{ er fjender}) = 2\left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}}$$

Nu har vi hvad vi skal bruge til at afgrænse sandsynligheden $p(\bigcup_{i=1}^{\binom{n}{k}} E_i)$ som er det, vi er interesserede i

$$\begin{aligned} p(\exists k \text{ indbyrdes venner/fjender}) =& p\left(\bigcup_{i=1}^{\binom{n}{k}} E_i\right) \\ \leq& \sum_{i=1}^{\binom{n}{k}} p(E_i) && \text{(Jf. Union Bound)} \\ =& \binom{n}{k} \cdot 2 \cdot \left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}} \\ \leq& \frac{n^k}{2^{k-1}} \cdot 2 \cdot \left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}} && \left(\text{Da } \binom{n}{k} \leq \frac{n^k}{2^{k-1}} \text{ jf. exercise 6.4.9 [*]}\right) \end{aligned}$$

[*] $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 2} \leq \frac{\overbrace{n \cdot n \cdot n \cdots n}^{k \text{ gange}}}{\underbrace{2 \cdot 2 \cdot 2 \cdots 2}_{k-1 \text{ gange}}} = \frac{n^k}{2^{k-1}}$

Nu antager vi i det følgende at $n < 2^{\frac{k}{2}}$ og skriver videre på det ovenstående:

$$\begin{aligned}
\frac{n^k}{2^{k-1}} \cdot 2 \cdot \left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}} &< \frac{(2^{\frac{k}{2}})^k}{2^{k-1}} \cdot 2 \cdot \left(\frac{1}{2}\right)^{\frac{k(k-1)}{2}} && \text{(Iflg. antagelse ovenfor)} \\
&= \frac{2^{k(\frac{k}{2})} \cdot 2}{2^{k-1} \cdot 2^{\frac{k(k-1)}{2}}} && \text{(Samler led)} \\
&= \frac{2^{\frac{k^2+2}{2}}}{2^{\frac{2(k-1)}{2} + \frac{k(k-1)}{2}}} && \text{(Samler eksponenter)} \\
&= \frac{2^{\frac{k^2+2}{2}}}{2^{\frac{k^2+k-2}{2}}} && \text{(Samler eksponenter)} \\
&= 2^{\frac{k^2+2}{2} - \frac{k^2+k-2}{2}} && \text{(Samler eksponenter)} \\
&= 2^{\frac{4}{2} - \frac{k}{2}} && \text{(Samler eksponenter)} \\
&= 2^{2 - \frac{k}{2}} && \text{(Samler eksponenter)}
\end{aligned}$$

Vi bemærker at $2^{2-(k/2)} \leq 1$ da $k \geq 4$. Så vi kan konkludere at $p(\bigcup_{i=1}^{\binom{n}{k}} E_i) < 1$ når $k \geq 4$. Så nu kan vi bruge *den probabilistiske metode* til at sige, at hvis $n < 2^{k/2}$ så eksisterer der med sikkerhed mindst én mængde af n personer, således at der ikke er nogen delmængder bestående af k personer, hvor alle k er indbyrdes venner eller indbyrdes fjender. Det følger at $R(k,k) \geq 2^{k/2}$.

□

The Birthday Problem

The birthday problem går ud på at afgøre det mindst antal personer, der skal være i samme rum sådan at sandsynligheden for at mindst 2 af dem har fødselsdag samme dag er $> 1/2$.

Antagelser: (1) Fødselsdage er uafhængige af hinanden og (2) der er 366 dage på et år (skudår).

Vi betragter personerne i rummet sekventielt i den rækkefølge de træder ind i rummet. Vi beregner sandsynligheden p_n for at de alle har fødselsdage forskellige dage og så finder vi $1 - p_n$ som er sandsynligheden for at mindst to af personerne har fødselsdag samme dag.

Vi observerer at $p_1 = 1$, fordi den første person der træder ind i rummet åbenlyst har en unik fødselsdato da rummet er tomt. For den anden person der træder ind i rummet er sandsynligheden $p_2 = 365/366$. For den tredje person er den $p_3 = 364/366$ osv. Mere generelt er sandsynligheden for at den j'te person med $2 \leq j \leq 366$ har fødselsdag på en anden dag end alle de andre $j - 1$ personer i rummet lig med

$$p_j = \frac{366 - (j-1)}{366} = \frac{367 - j}{366}$$

Da vi antog af fødselsdagene var uafhængige, så er sandsynligheden for at n personer i rummet har forskellige fødselsdage

$$p_n = p_1 \cdot p_2 \cdots p_n = \frac{366}{366} \cdot \frac{365}{366} \cdot \frac{364}{366} \cdots \frac{367 - n}{366}$$

Så

$$1 - p_n = 1 - \frac{366}{366} \cdot \frac{365}{366} \cdot \frac{364}{366} \cdots \frac{367 - n}{366}$$

Det viser sig at når $n = 23$ er $1 - p_n \approx 0.506 > 1/2$. Så når der er 23 personer i et rum er der en sandsynlighed $> 1/2$ for at to personer har samme fødselsdag.

Dette problem har anvendelse ift. at udregne sandsynligheden for kollision i en hash-tabel, da 'forskellige fødselsdage' kan erstattes med 'hasher til forskellige entries i hash-tabellen'.

7 Emne 7: Probabilistic analysis

Intro og overblik over præsentation

- Jeg vil fokusere på `Randomized-Quicksort` fra CLRS kap. 7.
- Først vil jeg introducere begreberne random variable og indicator random variable.
- Dernæst vil jeg benytte disse til at analysere den forventede køretid af Randomized-Quicksort.

(Indicator) random variables

Definition. En random variable er en funktion $f : S \to \mathbb{R}$, hvor S er udfaldsrummet.

Definition. Fordelingen af en random variable X på et udfaldsrum S består af par $(r, p(X = r))$, $\forall r \in X(S)$, hvor $p(X = S)$ er sandsynligheden for at X har værdien r.

Hvis vi fx lader X være en random variable der betegner udfaldet af et kast med en terning. Så har vi at $p(X = 6) = 1/6$, fordi sandsynligheden for at slå en 6'er er $1/6$.

Indicator random variables. Dette er en særlig type af random variables, der givet et udfaldsrum S og et udfald A har værdien 1 hvis A forekommer og 0 ellers. Det smarte ved indicator random variables er, at man nemt kan konvertere mellem den forventede værdi af en en sådan og sandsynligheden for at et udfald forekommer, som det følgende lemma siger:

Lemma Givet er udfaldsrum S og et udfald $A \in S$, lad da random variable $X_A = Y$, hvor $Y = 1$ hvis A forekommer og 0 ellers. Så gælder det at $E(X_A) = p(Y = 1)$.

Bevis.

$$\begin{aligned} E(X_A) &= E(Y) && \text{(Forventning på begge sider)} \\ &= 1 \cdot p(Y = 1) + 0 \cdot p(Y = 0) && \text{(Def. af exp. val.)} \\ &= p(Y = 1) \end{aligned}$$

□

Randomized Quicksort: Analyse af forventet køretid

Randomized-Quicksort adskiller sig fra Quicksort ved at benytte Randomzed-Partition fremfor Partition. Partition(A,p,r) bruger $A[r]$ (dvs. det sidste element i A) som pivot. Randomized-Partition vælger et heltal $i \in [p; r]$, swapper dette med $A[r]$ og kalder så Partition(A,p,r). Dvs. Randomized-Partition vælger et pivot uniformt tilfældigt i det diskrete interval $[p; r]$. Dette er den eneste forskel mellem Randomized-Quicksort og Quicksort.

Forventet køretid af Randomized-Quicksort. Vi ved fra analysen af Quicksort at det er i Partition (ikke Randomized-Partition) at det meste arbejde foregår. I `for`-loopet (linje 3-6) foretages partitioneringen af A med x son pivot. Resten af Randomized-Quicksort, herunder Randomized-Partition, tager $O(1)$ tid fordi det hovedsageligt består af funktionskald. Selve arbejdet i Partition består af at sammenligne elementer med x og dernæst placere det i én af de to subarrays.

Lad X være en random variable der angiver antallet af sammenligninger der foretages i `for`-loopet i Partition *over en hel kørsel af algoritmen* på et input array A hvor A.length $= n$. Så er køretiden af Randomized-Quicksort $O(n + X)$. Vi vil derfor nu finde den forventede værdi af X for derigennem at få en forventet køretid af Randomized Quicksort.

Antag at alle elementer i A er unikke. Lad os betegne elementerne i A som $z_1, z_2, \ldots, z_n$, hvor z_i er det i'te mindste element i A (dvs. vi analyserer på en sorteret version af A). Lad $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$.

Observér at to vilkårlige elementer z_i og z_j sammenlignes højest én gang. Dette skyldes at elementerne kun sammenlignet med pivot-elementet og når ét element har været pivot indgår det ikke i nogle af de subarrays som fungerer som inputs til de efterfølgende rekursive kald. Vi definerer en indicator random variable X_{ij}, hvor

$$X_{ij} = \begin{cases} 1 & \text{hvis } z_i \text{ og } z_j \text{ sammenlignes på et tidspunkt} \\ 0 & \text{ellers} \end{cases}$$

Da ethvert par af elementer sammenlignes højest én gang[4] har vi at det samlede antal gange algoritmen foretager en sammenligning mellem to elementer

$$\begin{aligned} X &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \quad \Rightarrow \\ E(X) &= E\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{ij}) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \cdot 1 + p(z_i \text{ og } z_j \text{ sammenlignes ikke}) \cdot 0 \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \end{aligned}$$

[4]Dette er årsagen til at den inderste sum nedenfor indekseres med $j = i + 1$; hvis det havde været $j = 1$ ville alle elementer sammenlignes to gange og med sig selv.

(I anden linje tages expectation på begge side; i tredje linje bruges linearity of expectation; i fjerde linje bruges formel for expected value (se Thm. 7.4.1 i Rosen); i sidste linje simplificeres udtrykket.)

Så vi ønsker at finde $p(z_i$ og z_j sammenlignes). Bemærk at hvis der vælges et x, hvor $z_i < x < z_j$, så bliver z_i og z_j aldrig sammenlignet, fordi de efterfølgende vil være i forskellige subarrays. Hvis det derimod er tilfældet for det første pivot $x \in Z_{ij}$ at $x = z_i$ eller $x = z_j$, så sammenlignes z_i og z_j. Så z_i og z_j sammenlignes *hvis og kun hvis* det første pivot element $x \in Z_{ij}$ er enten z_i eller z_j.

Da Z_{ij} indeholder $j - i + 1$ elementer er sandsynligheden for dette udfald

$$\begin{aligned} p(z_i \text{ og } z_j \text{ sammenlignes}) =& p(z_i \text{ eller } z_j \text{ er første pivot fra } Z_{ij}) \\ =& p(z_i \text{ er første pivot fra } Z_{ij}) + p(z_j \text{ er første pivot fra } Z_{ij}) \\ =& \frac{1}{j-i+1} + \frac{1}{j-i+1} \\ =& \frac{2}{j-i+1} \end{aligned}$$

(Anden linje følger af sum-reglen; det er et 'enten-eller' udfald, hvor udfaldene er gensidigt udelukkende).

Hvilket giver os

$$
\begin{aligned}
E(X) =& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \\
=& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
=& \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} && \text{(Ændrer indeks så } k = j - i\text{)} \\
<& \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\
=& \sum_{i=1}^{n-1} \left(2 \sum_{k=1}^{n} \frac{1}{k} \right) && \text{(Hiver konstant ud og inkluderer flere led)} \\
=& \sum_{i=1}^{n-1} 2H_n \\
=& \sum_{i=1}^{n-1} O(\log_2 n) && \text{(Da } H_n = O(\log_2 n)\text{)} \\
=& O(n \cdot \log_2 n)
\end{aligned}
$$

Det vil sige at `for`-loopet i Partition forventes at foretage $O(n \log n)$ sammenligninger alt i alt, og da best-case for Quicksort er $\Theta(n \log n)$ så er den forventede køretid af Randmmized-Quicksort altså $\Theta(n + n \log n) = \Theta(n \log n)$.

8 Emne 8: Examples of applications of indicator random variables

Intro og overblik over præsentation

- Jeg vil først definere indicator random variables og redegøre for, hvorfor de er nyttige via et lille bevis.
- Dernæst vil jeg bruge indicator random variables til at analysere den forventede køretid af Randomized-Quicksort.
- Hvis der er mere tid, vil vil jeg vise hvordan man kan bruge indicator random variables til at udregne det forventede antal inversioner i et array.

Definition: Indicator random variables

Indicator random variables. Dette er en særlig type af random variables, der givet et udfaldsrum S og et udfald A har værdien 1 hvis A forekommer og 0 ellers. Det smarte ved indicator random variables er, at man nemt kan konvertere mellem den forventede værdi af en en sådan og sandsynligheden for at et udfald forekommer, som det følgende lemma siger:

Lemma Givet er udfaldsrum S og et udfald $A \in S$, lad da random variable $X_A = Y$, hvor $Y = 1$ hvis A forekommer og 0 ellers. Så gælder det at $E(X_A) = p(Y = 1)$.

Bevis.

$$\begin{aligned} E(X_A) =& E(Y) && \text{(Forventning på begge sider)} \\ =& 1 \cdot p(Y = 1) + 0 \cdot p(Y = 0) && \text{(Def. af exp. val.)} \\ =& p(Y = 1) \end{aligned}$$

Randomized Quicksort: Analyse af forventet køretid

Randomized-Quicksort adskiller sig fra Quicksort ved at benytte Randomzed-Partition fremfor Partition. Partition(A,p,r) bruger $A[r]$ (dvs. det sidste element i A) som pivot. Randomized-Partition vælger et heltal $i \in [p; r]$, swapper dette med $A[r]$ og kalder så Partition(A,p,r). Dvs. Randomized-Partition vælger et pivot uniformt tilfældigt i det diskrete interval $[p; r]$. Dette er den eneste forskel mellem Randomized-Quicksort og Quicksort.

Forventet køretid af Randomized-Quicksort. Vi ved fra analysen af Quicksort at det er i Partition (ikke Randomized-Partition) at det meste arbejde foregår. I `for`-loopet (linje 3-6) foretages partitioneringen af A med x son pivot. Resten af Randomized-Quicksort, herunder Randomized-Partition, tager $O(1)$ tid fordi det hovedsageligt består af funktionskald. Selve arbejdet i Partition består af at sammenligne elementer med x og dernæst placere det i én

af de to subarrays.

Lad X være en random variable der angiver antallet af sammenligninger der foretages i `for`-loopet i Partition *over en hel kørsel af algoritmen* på et input array A hvor A.length $= n$. Så er køretiden af Randomized-Quicksort $O(n + X)$. Vi vil derfor nu finde den forventede værdi af X for derigennem at få en forventet køretid af Randomized Quicksort.

Antag at alle elementer i A er unikke. Lad os betegne elementerne i A som $z_1, z_2, \ldots, z_n$, hvor z_i er det i'te mindste element i A (dvs. vi analyserer på en sorteret version af A). Lad $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$.

Observér at to vilkårlige elementer z_i og z_j sammenlignes højest én gang. Dette skyldes at elementerne kun sammenlignet med pivot-elementet og når ét element har været pivot indgår det ikke i nogle af de subarrays som fungerer som inputs til de efterfølgende rekursive kald. Vi definerer en indicator random variable X_{ij}, hvor

$$X_{ij} = \begin{cases} 1 & \text{hvis } z_i \text{ og } z_j \text{ sammenlignes på et tidspunkt} \\ 0 & \text{ellers} \end{cases}$$

Da ethvert par af elementer sammenlignes højest én gang[5] har vi at det samlede antal gange algoritmen foretager en sammenligning mellem to elementer

$$\begin{aligned} X &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \quad \Rightarrow \\ E(X) &= E\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{ij}) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \cdot 1 + p(z_i \text{ og } z_j \text{ sammenlignes ikke}) \cdot 0 \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \end{aligned}$$

(I anden linje tages expectation på begge side; i tredje linje bruges linearity of expectation; i fjerde linje bruges formel for expected value (se Thm. 7.4.1 i Rosen); i sidste linje simplificeres udtrykket.)

Så vi ønsker at finde $p(z_i$ og z_j sammenlignes). Bemærk at hvis der vælges et x, hvor $z_i < x < z_j$, så bliver z_i og z_j aldrig sammenlignet, fordi de efterfølgende vil være i forskellige

[5] Dette er årsagen til at den inderste sum nedenfor indekseres med $j = i + 1$; hvis det havde været $j = 1$ ville alle elementer sammenlignes to gange og med sig selv.

subarrays. Hvis det derimod er tilfældet for det første pivot $x \in Z_{ij}$ at $x = z_i$ eller $x = z_j$, så sammenlignes z_i og z_j. Så z_i og z_j sammenlignes *hvis og kun hvis* det første pivot element $x \in Z_{ij}$ er enten z_i eller z_j.

Da Z_{ij} indeholder $j - i + 1$ elementer er sandsynligheden for dette udfald

$$\begin{aligned}
p(z_i \text{ og } z_j \text{ sammenlignes}) =& p(z_i \text{ eller } z_j \text{ er første pivot fra } Z_{ij}) \\
=& p(z_i \text{ er første pivot fra } Z_{ij}) + p(z_j \text{ er første pivot fra } Z_{ij}) \\
=& \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
=& \frac{2}{j-i+1}
\end{aligned}$$

(Anden linje følger af sum-reglen; det er et 'enten-eller' udfald, hvor udfaldene er gensidigt udelukkende).

Hvilket giver os

$$\begin{aligned}
E(X) =& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(z_i \text{ og } z_j \text{ sammenlignes}) \\
=& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
=& \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} && \text{(Ændrer indeks så } k = j - i\text{)} \\
<& \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\
=& \sum_{i=1}^{n-1} \left(2 \sum_{k=1}^{n} \frac{1}{k} \right) && \text{(Hiver konstant ud og inkluderer flere led)} \\
=& \sum_{i=1}^{n-1} 2H_n \\
=& \sum_{i=1}^{n-1} O(\log_2 n) && \text{(Da } H_n = O(\log_2 n)\text{)} \\
=& O(n \cdot \log_2 n)
\end{aligned}$$

Det vil sige at `for`-loopet i Partition forventes at foretage $O(n \log n)$ sammenligninger alt i alt, og da best-case for Quicksort er $\Theta(n \log n)$ så er den forventede køretid af Randmmized-Quicksort altså $\Theta(n + n \log n) = \Theta(n \log n)$.

Antal inversioner i et array (hvis der er tid)

Lad $A[1 \ldots n]$ være et array bestående af n unikke tal. Hvis $i < j$ og $A[i] > A[j]$, så udgør parret (i, j) en *inversion* i A. Antag at A er en tilfældig permutation af $\langle 1, 2, \ldots, n \rangle$. Så kan vi bruge indicator random variables til at finde det forventede antal inversioner i A.

Lad X_{ij} være en indicator random variable med

$$X_{ij} = \begin{cases} 1 & \text{hvis } i < j \text{ og } A[i] > A[j] \\ 0 & \text{ellers} \end{cases}$$

Lad X være det totale antal inversioner i A. Det er klart at

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

fordi dette er summen over alle mulige par i, j præcis én gang (se fornote 3). Da A er en tilfældig permutation må det være tilfældet at

$$E(X_{ij}) = 1 \cdot p(X_{ij} = 1) + 0 \cdot p(X_{ij} = 0) = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}$$

fordi for vilkårlige par i, j er $p(A[i] > A[j]) = p(A[i] < A[j]) = 1/2$.

Ved at tage expected value på begge sider får vi

$$\begin{aligned}
E(X) =& E\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right) \\
=& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{ij}) && \text{(Linearity of expectation)} \\
=& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} \\
=& \frac{1}{2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 && \text{(Hiver konstant ud)} \\
=& \frac{1}{2} \sum_{i=1}^{n-1} (n - i) \\
=& \frac{1}{2} \sum_{i=1}^{n-1} i && \text{(Dette vender blot summeringen om)} \\
=& \frac{1}{2} \cdot \frac{n(n-1)}{2} && \text{(Lukket form)} \\
=& \frac{n(n-1)}{4}
\end{aligned}$$

Så under fornævnte antagelser ville vi forvente $\frac{n(n-1)}{4}$ inversioner A.

9 Emne 9: Universal hashing

Intro og overblik over præsentation

- Jeg vil først definere universal hashing og fortælle, hvad fordelen ved denne type hashing er frem for normal hashing.
- Dernæst vil jeg designe en klasse af hashfunktioner og efterfølgende bevise at den er universel.

Univsersel hashing: Definitioner og egenskaber

Universel hashing betegner en approcah til at vælge en hashfunktion således at den har god performance gennemsnitligt set.

Lad $\mathcal{H}$ være en endelig mængde af hashfunktioner der mapper keys fra et univers U til et heltal i intervallet $\{0, 1, 2, \ldots, m-1\}$. $\mathcal{H}$ er *universel* hvis for et vilkårligt par af keys $k, l \in U$, hvor $k \neq l$, gælder, at hvis vi vælger $h \in_R \mathcal{H}$, så er $\Pr(h(k) = h(l)) \leq 1/m$.

Den gode gennemsnitlige performance består i at hvis vi vælger et $h \in_R \mathcal{H}$ og mapper n keys ind i en hashtabel T med m slots, og hvor kollisioner løses med chaining, så er $E(n_{h(k)})$ (dvs. den forventede længde at chain i det slot key k hasher til i T) højest n/m, hvis key k er i tabellen og $1 + n/m$ hvis k ikke er i T.

Design af en universel klasse af hashfunktioner

Vælg et primtal p sådan at det for alle $k \in U$ gælder at $0 \leq k \leq p - 1$. Lad $\mathbb{Z}_p = \{0, 1, 2, \ldots, p - 1\}$ og $\mathbb{Z}_p^* = \{1, 2, \ldots, p - 1\}$. Lad m være antallet af slots i hashtabellen. Det er klart at $m < |U|$, så $m < p$.

Nu definerer vi hashfunktionen $h_{ab}(k) = ((ak + b) \bmod p) \bmod m$, hvor $a \in \mathbb{Z}_p^*$ og $b \in \mathbb{Z}_p$. Så er klassen af hashfunktioner givet ved $\mathcal{H}_{pm} = \{h_{ab} | a \in \mathbb{Z}_p^* \wedge b \in \mathbb{Z}_p\}$.

Thm. $\mathcal{H}_{pm}$ er universel.

Bevis. Når vi skal vise at $\mathcal{H}_{ab}$ er universel, skal vi vise at for en vilkårlig hashfunktion $h \in \mathcal{H}$ så gælder det at for et vilkårligt par af keys $k, l \in U$, hvor $k \neq l$ så er sandsynligheden $\Pr(h(k) = h(l)) \leq 1/m$.

Vi starter med at vise at der ikke er nogen kollisioner på "mod p niveauet.":

Betragt to keys $k, l \in U$, hvor $k \neq l$. For en given hashfunktion h_{ab}, lader vi

$$\begin{aligned} r &= (ak + b) \bmod p \\ s &= (al + b) \bmod p \end{aligned}$$

Det må være tilfældet at $r \neq s$ fordi

$$\begin{aligned}(r-s) \bmod p =& (((ak+b) \bmod p) - ((al+b) \bmod p)) \bmod p \\ =& ((ak+b) - (al+b)) \bmod p && \text{(Corr. 4.1.2 i Rosen)} \\ =& a(k-l) \bmod p \\ \Rightarrow& (r-s) \equiv a(k-l) \pmod p\end{aligned}$$

Og eftersom at $a \bmod p \neq 0$ da $a < p$ og $(k-l) \bmod p \neq 0$ da $k \neq l$ og $k - l < p$ så er $a(k-l) \bmod p \neq 0$ da p er et primtal.[6] Derfor må det gælde at $r - s \neq 0$. Dette skyldes at hvis $r - s = 0$, så ville $r - s \bmod p = 0$ men $a(k-l) \bmod p \neq 0$ og de to ting er kongruente med hinanden, så det ville være en modstrid.

Så for en vilkårlig $h_{ab} \in \mathcal{H}_{pm}$ så mapper k og l til distinkte værdier r og s. Dette viser at der ikke er nogen kollisioner på modulo p niveauet.

Nu vil vi gerne vise at der er en bijektion mellem par (a, b) og de resulterende par (r, s), med andre ord at hvert par (a, b) med $a \neq 0$ har et forskellige par (r, s) med $r \neq s$. Hvis vi betragter en mapping fra (a, b) til (r, s), så viser vi først at denne er injektiv. Dette kan vi gøre ved at isolere a og b givet r og s. Hvis vi lader $(k-1)^{-1} \bmod p$ være den multiplikative invers til $(k-l)$ og isolerer a, får vi:

$$\begin{aligned}(r-s) \bmod p =& (((ak+b) \bmod p) - ((al+b) \bmod p)) \bmod p \\ =& a(k-l) \bmod p && \Leftrightarrow \\ (a(k-l)(k-l)^{-1} \bmod p)) \bmod p =& ((r-s)((k-l)^{-1} \bmod p)) \bmod p && \Leftrightarrow \\ \overbrace{a \bmod p}^{=a \text{ da } a<p} =& ((r-s)((k-l)^{-1} \bmod p)) \bmod p && \Leftrightarrow \\ a =& ((r-s)((k-l)^{-1} \bmod p)) \bmod p\end{aligned}$$

og isolerer b

[6] Hvis $a(k-l) \bmod p = 0$ når $a(k-l) \neq p$ (husk at $a(k-l) \neq p$ da dette ville være en modstrid til at p er et primtal), så skulle $a(k-l) = p \cdot c$ for et $c \in \mathbb{Z}$ jf. definition af modulo og division. Altså ville p være en primtalsfaktor i $p \cdot c$. Men p er ikke en pimtalsfaktor i $a(k-l)$ da a og $k-l$ har deres egen primtalsfaktorisering der ikke involverer p da $a < p$ og $k - l < p$. Da da en primtalsfaktorisering er unik kan de to sider af lighedstegnet ikke være lig med hinanden, ergo $a(k-l) \text{mod } p \neq 0$.

$$
\begin{aligned}
r &= (ak+b) \bmod p && \Rightarrow \\
\overbrace{r \bmod p}^{=r \text{ da } r<p} &= (ak+b) \bmod p && \Leftrightarrow \\
r &\equiv (ak+b) \pmod p && \Leftrightarrow \\
(r-ak) &\equiv b \pmod p && \Leftrightarrow \text{ (Trækker } ak \text{ fra på begge sider)} \\
(r-ak) \bmod p &= b \bmod p && \Rightarrow \text{ (Jf. def. af kongruens)} \\
(r-ak) \bmod p &= b && \text{(Fordi } b \bmod p = b \text{ da } b < p\text{)}
\end{aligned}
$$

Så alle par (a, b) har et unikt par (r, s). Vi bemærker desuden at der er $p(p-1)$ mulige par (r, s), hvor $r \neq s$ samt $p(p-1)$ mulige par (a, b), hvor $a \neq 0$. Så den førnævnte mapping er surjektiv. Dette medfører at der er en bijektion mellem mængden af par (a, b), hvor $a \neq 0$ og par (r, s), hvor $r \neq s$. Det følger, at hvis vi vælger $a, b \in_R \mathbb{Z}_p \times \mathbb{Z}_p^*$ så er der en lige stor sandsynlighed for at hvilket som helst talpar (r, s) modulo p bliver resultatet heraf.

Nu bevægervi os fra "modulo p niveauet" ned til "modulo m niveauet."

Det følger af det ovenstående, at $\Pr(h(l) = h(k)) = \Pr(r \equiv s \pmod m)$ da vi ved at $r \neq s$ men det er muligt at de er kongruente modulo m, i hvilket fald der vil være en kollision. Hvis vi betragter en vilkårligt værdi r, så kan vi udregne, hvor mange af de tilbageværrende $p-1$ værdier for s har $r \equiv s \pmod m$. Vi har $p-1$ valgmuligheder, så antallet af værdier hvor r og s, $r \neq s$, kolliderer modulo m er højest

$$
\begin{aligned}
\left\lceil \frac{p}{m} \right\rceil - 1 &\leq \frac{p+m-1}{m} - 1 && \text{(Jf. ulighed 3.6, p. 54 i CLRS)} \\
&= \frac{p}{m} + \frac{m}{m} - \frac{1}{m} - 1 \\
&= \frac{p-1}{m}
\end{aligned}
$$

Så sandsynligheden for at s kolliderer med r når de reduceres modulo m er højest

$$\Pr(r \equiv s \pmod m) \leq \frac{\frac{p-1}{m}}{p-1} = \frac{p-1}{m(p-1)} = \frac{1}{m}$$

fordi $(p-1)/m$ er den største mulige størrelse af udfaldet jf. ovenstående udregning, vi er interesserede i at finde sandsynligheden for, og $p-1$ er størrelsen af udfaldsrummet.

Så for et et hvilket som helst par $k, l \in \mathbb{Z}_p$ (og dermed et hvilket som helst par $k, l \in U$, da $U \subset \mathbb{Z}_p$), så er

$$\Pr(h_{ab}(k) = h_{ab}(l)) \leq \frac{1}{m}$$

Det følger at $\mathcal{H}$ er universel.

□

10 Emne 10: String matching

Intro og overblik over præsentation

- Jeg vil fokusere på endelige automater til string matching.
- Først vil jeg definere og præsentere terminologi.
- Dernæst vil jeg bevise automatens korrekthed.

Definitioner og terminologi

String matching med en endelig automat er en effektiv metode, fordi alle symbolerne i teksten behandles præcis én gang og tager $O(1)$ tid per symbol. Flere string matching algoritmer – fx KMP-algoritmen – benytter sig af logikken bag en endelig string matching automat.

En endelig automat (generelt) M er en 5-tuple $M = (Q, q_0, A, \Sigma, \delta)$, hvor

- Q er endelig mængde af states
- $q_0 \in Q$ er start state
- $A \subseteq Q$ er mængden af accept states
- Σ er input-alfabetet
- δ er en transition funktion der mapper $Q \times Z \to Q$. Det er beregningen af δ, der dominerer køretiden fordi denne er $O(m|\Sigma])$.

Automaten fungerer således at den begynder i state q_0, hvorefter den læser inputtet ét symbol ad gangen. Hvis automaten er i state q og læser symbolet a så laver den en state transition til $\delta(q, a)$. Når et state $q \in A$ nås, så har vi et match med vores pattern og hvis vi ikke når et $q \in A$ så har vi ikke et match.

En endelig automat til string matching M påfører inputtet en funktion $\phi(w)$ (final state function) som er det state M ender i efter strengen w er blevet læst. Den defineres rekursivt som

$$\begin{aligned} \phi(\varepsilon) &= q_0 && \text{hvor } \varepsilon \text{ betegner den tomme streng} \\ \phi(wa) &= \delta(\phi(w), a), && \text{hvor } w \in \Sigma^* \text{ og } a \in \Sigma \end{aligned}$$

Så M accepterer en streng w hvis og kun hvis $\phi(w) \in A$.

Når vi har med string matching at gøre har vi en tekst $T[1 \ldots n]$ og et mønster $P[1 \ldots m]$. Vi skal bruge en hjælpefunktion $\sigma : \Sigma^* \to \{0, 1, 2, \ldots, m\}$, hvor $\sigma(x)$ er længden af det det længste præfix i P der også er et suffix i x, d.v.s.

$$\sigma(x) = \max\{k |\, P_k \sqsupset x\}$$

Nu har vi hvad vi skal bruge til at definere en endelig string matching automat M givet en mønster $P[1 \ldots m]$ og en tekst $T[1 \ldots n]$:

- $Q = \{0, 1, 2, 3, \ldots, m\}$, $q_0 = 0$ og $A = \{m\}$
- Transitionsfunktionen δ er defineret som $\delta(q, a) = \sigma(P_q a)$ for et vilkårligt state q og et vilkårligt symbol a. Dette skyldes at vi ønsker at holde styr på, hvor langt et præfix af P vi har matchet i T indtil videre.

Vi vil nu bevise at M udfører korrekt string matching.

Bevis for korrekthed

Lemma 1. For en vilkårlig streng x og et vilkårligt symbol a gælder det at $\sigma(xa) \leq \sigma(x)+1$.

Bevis. Lad $r = \sigma(xa)$. To cases:

Case 1: $r = 0$. Trivielt sandt at $\sigma(xa) = 0 \leq \sigma(x) + 1$, fordi σ ikke kan være negativ.

Case 2: $r > 0$. I denne case har vi

$$\begin{aligned}
& P_r \sqsupset xa && \text{(Def. af } \sigma\text{)} \\
\Rightarrow\ & P_{r-1} \sqsupset x && \text{(Fjerner } a \text{ fra } P_r \text{ og } xa\text{)} \\
\Rightarrow\ & r - 1 \leq \sigma(x) && \text{(Kunne reelt også være } =\text{)} \\
\Rightarrow\ & \sigma(xa) = r \leq \sigma(x) + 1
\end{aligned}$$

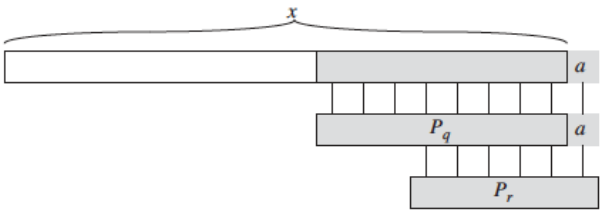
□

Lemma 2. For en vilkårlig streng x og et vilkårligt symbol a gælder at $q = \sigma(x) \Rightarrow \sigma(xa) = \sigma(P_q a)$.

Bevis.
Antag at hypotesen er sand. Så ved vi at

$$\begin{aligned}
& P_q \sqsupset x && \text{(Def. af } \sigma\text{)} \\
\Rightarrow\ & P_q a \sqsupset xa && \text{(Tilfjøer et } a \text{ til præfix og suffix)}
\end{aligned}$$

Figuren nedenfor illustrerer ovenstående.



Lad $r = \sigma(xa)$ får vi

$$
\begin{aligned}
& r = \sigma(xa) && \\
\Leftrightarrow\ & P_r \sqsupset xa && \text{(Def. af } \sigma\text{)} \\
\Rightarrow\ & r \leq \sigma(x) + 1 = q + 1 && \text{(Jf. lemma 1 og } q = \sigma(x)\text{)} \\
\Leftrightarrow\ & |P_r| = r \leq q + 1 = |P_q a| && \text{(Siger samme som forrige linje)}
\end{aligned}
$$

Så vi har at $P_q a \sqsupset xa$, $P_r \sqsupset xa$ og $|P_r| \leq |P_q a|$. Det følger at $P_r \sqsupset P_q a$. [7]

Nu kan vi slutte at $\sigma(xa) \leq \sigma(P_q a)$ fordi $P_r \sqsupset P_q a$. Men vi har også at $\sigma(p_q a) \leq \sigma(xa)$, da $P_q a \sqsupset xa$, dvs. $P_q a \sqsupset P_r$. Det følger at $\sigma(xa) = \sigma(P_q a)$.

□

Teorem. Hvis ϕ er final-state funktionen af en endelig streng matching automat M for et mønster P og $T[1 \ldots n]$ er input teksten til M så er $\phi(T_i) = \sigma(T_i)$ for $i = 0, 1, 2, \ldots, n$.

Bevis. Dette viser vi ved induktion over i.

Base case. $i = 0$. Her er sætningen trivielt sandt da $T_0 = \varepsilon$, så $\phi(T_0) = 0 = \sigma(T_0)$.

Induktionstrin. Antag at $q = \phi(T_i) = \sigma(T_i)$. Lad $a = T[i + 1]$. Så gælder det at

$$
\begin{aligned}
\phi(T_{i+1}) =& \phi(T_i a) && \text{(Def. af } T_{i+1} \text{ og } a\text{)} \\
=& \delta(\phi(T_i), a) && \text{(Def. af } \phi\text{)} \\
=& \delta(q, a) && \text{(Def af } q\text{)} \\
=& \sigma(P_q a) && \text{(Def. af } \delta\text{)} \\
=& \sigma(T_i a) && \text{(Jf. induktionsantagelse } [q = \phi(T_i) = \sigma(T_i)] \text{ og lemma 2)} \\
=& \sigma(T_{i+1}) && \text{(Def. af } T_{i+1}\text{)}
\end{aligned}
$$

(Den næstsidste linje får vi ved $q = \sigma(T_1) \Rightarrow \sigma(P_q a) = \sigma(T_i a)$, så vha. induktionsantagelse og lemma 2).

□

Det vi nu har vist er, at hvis M kommer i state q, så er det fordi q er den største værdi sådan at $P_q \sqsupset T_i$. Så vi har $q = m$ hvis og kun hvis automaten har lokaliseret en forekomst af P i T. Vi kan således konkludere M fungerer korrekt.

[7] Jf. lemma 32.1 p. 987 der siger at for strenge x, y, z, hvis $x \sqsupset z$ og $y \sqsupset y$ of $|y| \leq |x|$, så er $x \sqsupset y$.