

Counting - Chapter 6	3
Pigeonhole Principle	3
Permutations and combinations	3
Binomial Coefficients	4
Pascals Triangle	5
Vandermonde's identity	6
Generalized Permutation and Combinations	6
Generating Permutations and Combinations	7
Discrete Probability - Chapter 7	8
Finite Probability	8
Probabilities of Complements and Unions of Events	8
Probabilistic Reasoning - Monty Hall Puzzle	9
Probability Theory	10
Conditional Probability	11
Independence	11
Bernoulli Trials and Binomial Distribution	12
Random Variables	13
The Birthday Problem	14
Monte Carlo Algorithms	15
The Probabilistic Method	15
Bayes Theorem	16
Expected Value and Variance	19
Linearity of Expectations	20
Geometric Distribution	22
Variance	22
Markov's inequality	23
Chebyshev's inequality	24
Advanced Counting Techniques - Chapter 8	25
Linear Recurrence Relations - Modeling	25
Solving linear homogeneous recurrence relations	25

Linear non homogeneous recurrence relations	27
Inclusion-Exclusion	29
Randomized Algorithms - Kleinberg & Tardos	32
Contention Resolution	32
Global Minimum Cut	34
Coupon Collector	36
Randomized Approximation for MAX 3-SAT	36
Finding the Median	37
Quicksort	38
Chernoff Bounds	39
Randomized Algorithms - CLRS	40
The Hiring Problem	40
Randomized Quicksort	42
Universal Hashing	43
Network Flows	45
Ford-Fulkerson	46
Max flow min cut theorem:	47
Edmonds-Karp	48
Maximum bipartite matching	48
String Matching	49
Naive String Matching	49
Rubin-Karp Algorithm	50

Counting - Chapter 6

Pigeonhole Principle

THEOREM 1

THE PIGEONHOLE PRINCIPLE If k is a positive integer and $k + 1$ or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.

Hvis man har mere end k objekter og k bokse, vil mindst en boks have mere end en objekt

THEOREM 2

THE GENERALIZED PIGEONHOLE PRINCIPLE If N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/k \rceil$ objects.

N objekter placeres ind i k kasser, så er der mindst en box der har $\left\lceil \frac{N}{k} \right\rceil$ objekter

Proof by contraposition:

$$N > k \rightarrow \left\lceil \frac{N}{k} \right\rceil > 1$$

$$\psi \rightarrow \varphi$$

$$\left\lceil \frac{N}{k} \right\rceil \leq 1 \rightarrow N \leq k$$

$$\neg\varphi \rightarrow \neg\psi$$

$$N \leq k \leftrightarrow \frac{N}{k} \leq 1$$

assume $\neg\varphi$

$$\rightarrow N > k \rightarrow \left\lceil \frac{N}{k} \right\rceil > 1$$

prove $\neg\psi$

□

Permutations and combinations

Permutationer er en ordning af et sæt af elementer

Eksempel 1: På hvor mange måder kan vi vælge 3 studerende fra en gruppe af 5 til at stå på en linje?

Rækkefølgen betyder noget her. Fem muligheder for første elev, fire muligheder for anden elev og tre muligheder for tredje elev:

$$5 \cdot 4 \cdot 3 = 60$$

r -Permutation = $P(n, r)$, så n er alle elementer hvor r er hvor mange elementer vi har i ordningen.

$$P(n, r), 1 \leq r \leq n$$

Product rule:

$$P(n, r), 1 \leq r \leq n = n(n-1)(n-2) \dots (n-r+1) = \frac{n!}{(n-r)!}$$

Combinations:

THEOREM 2

The number of r -combinations of a set with n elements, where n is a nonnegative integer and r is an integer with $0 \leq r \leq n$, equals

$$C(n, r) = \frac{n!}{r!(n-r)!}.$$

Proof:

Hvis r-permutationen er et valgt subset og så de forskellige ordninger af det subset, kan r-permutation opsættes:

$$P(n, r) = C(n, r) \cdot P(r, r)$$

The implicerer at:

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{n!/(n-r)!}{r!/(r-r)!} = \frac{n!}{r!(n-r)!}$$

Eksempel 15:

Tre studerende fra matematik og fire studerende fra datalogi skal lave en committee. Der er 11 datalogi studerende i alt, og 9 matematik studerende i alt.

$$C(9,3) \cdot C(11,4) = \frac{9!}{3!6!} \cdot \frac{11!}{4!7!} = 84 \cdot 330 = 27720$$

Binomial Coefficients

Binomial theorem:

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

Proof by combinatorics:

$$T = \{x, y\} + \{x, y\} + \{x, y\} + \dots + \{x, y\}, |T| = n$$

Vi kigger på alle termer som mængder af x og y i hvert term, hvor antallet af term er n.

Så ud fra teoremet, vælger vi en kombination af x'er og y'er hvor hvis vælger ingen y'er, får man x n gange da man vælger x fra hvert term:

$$\binom{n}{0} x^n$$

Så med ét y:

$$\binom{n}{1} x^{n-1} y^1$$

Og da det ene term vælger en y har vi kun n-1 termer at vælge x fra.

Osv op til :

$$\binom{n}{n} x^0 y^n$$

THEOREM 2
PASCAL'S IDENTITY

Let n and k be positive integers with $n \geq k$. Then

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$

COROLLARY 4

If n is a nonnegative integer, then

$$\binom{2n}{n} = \sum_{k=0}^n \binom{n}{k}^2.$$

Proof: We use Vandermonde's identity with $m = r = n$ to obtain

$$\binom{2n}{n} = \sum_{k=0}^n \binom{n}{n-k} \binom{n}{k} = \sum_{k=0}^n \binom{n}{k}^2.$$

The last equality was obtained using the identity $\binom{n}{k} = \binom{n}{n-k}$.

Pascals Triangle

For at finde binomial coefficients fra en binomial opløftet i n. Vælger man det $n+1$ 'ende række fra trekanten.

Eksempel: $n = 6$

$$(x+y)^6 = 1x^6 + 6x^5y^1 + 15x^4y^2 + 20x^3y^3 + 15x^2y^4 + 6x^1y^5 + 1y^6, \text{ ved at kigge på}$$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad 1 \quad 1$$

$$\binom{2}{0} \binom{2}{1} \binom{2}{2} \quad \text{By Pascal's identity:} \quad 1 \quad 2 \quad 1$$

$$\begin{pmatrix} 3 \\ 0 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \end{pmatrix} \quad \quad \begin{pmatrix} 6 \\ 4 \end{pmatrix} + \begin{pmatrix} 6 \\ 5 \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \end{pmatrix} \quad \quad 1 \quad 3 \quad 3 \quad 1$$

$$\begin{pmatrix} 4 \\ 0 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix} \begin{pmatrix} 4 \\ 3 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \end{pmatrix} \quad 1 \quad 4 \quad 6 \quad 4 \quad 1$$

$$\binom{5}{0} \binom{5}{1} \binom{5}{2} \binom{5}{3} \binom{5}{4} \binom{5}{5} \quad 1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1$$

$$\binom{6}{0} \binom{6}{1} \binom{6}{2} \binom{6}{3} \binom{6}{4} \underbrace{\binom{6}{5}}_{\binom{6}{6}} \quad 1 \quad 6 \quad 15 \quad 20 \quad \cancel{15} \quad \cancel{6} \quad 1$$

$$\binom{7}{0} \binom{7}{1} \binom{7}{2} \binom{7}{3} \binom{7}{4} \binom{7}{5} \binom{7}{6} \binom{7}{7} \quad 1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1$$

$$\left(\begin{matrix} 8 \\ 0 \end{matrix}\right) \left(\begin{matrix} 8 \\ 1 \end{matrix}\right) \left(\begin{matrix} 8 \\ 2 \end{matrix}\right) \left(\begin{matrix} 8 \\ 3 \end{matrix}\right) \left(\begin{matrix} 8 \\ 4 \end{matrix}\right) \left(\begin{matrix} 8 \\ 5 \end{matrix}\right) \left(\begin{matrix} 8 \\ 6 \end{matrix}\right) \left(\begin{matrix} 8 \\ 7 \end{matrix}\right) \left(\begin{matrix} 8 \\ 8 \end{matrix}\right) \quad 1 \quad 8 \quad 28 \quad 56 \quad 70 \quad 56 \quad 28 \quad 8 \quad 1$$

•

(a)

(1-)

række 6+1 i trekanten.

Eksempel: $n = 8$

$$(x + y)^8 = x^8 + 8x^7y + 28x^6y^2 + 56x^5y^3 + 70x^4y^4 + 56x^3y^5 + 28x^2y^6 + 8xy^7 + y^8$$

Vandermonde's identity

THEOREM 3

VANDERMONDE'S IDENTITY

Let m , n , and r be nonnegative integers with r not exceeding either m or n . Then

$$\binom{m+n}{r} = \sum_{k=0}^r \binom{m}{r-k} \binom{n}{k}.$$

Vælger man r elementer fra to sæt m og n , er det samme som at vælge k elementer fra det første sæt og så $r-k$ fra det andet sæt. Ved brug af produktreglen kan man gange de to udtræk fra begge sæt for at finde måder at vælge r elementer fra begge sæt.

$$\binom{10}{3} = \binom{10}{10-3} = \binom{10}{7}$$

Generalized Permutation and Combinations

Permutations with repetitions:

Antallet af r -permutationer i et sæt af n elementer er n^r fordi repetition er tilladt vil der være n muligheder for hvert element i sættet.

Combination with repetitions:

Kombinationer hvor rækkefølge ikke betyder noget.

$$\binom{n+r-1}{r}$$

Eksempel:

4 forskellige kopper vælges ud af 15 kopper. Hvor mange måder er der at vælge 4 kopper med repetition?

$$\binom{15+4-1}{4} = \binom{18}{4} = 3060$$

Stars and bars:

Bars divider de forskellige typer. Så hvis der er 7 forskellige typer af elementer vil der være 6 dividere. Stjerner repræsenterer hvad der bliver trukket (eksempelvis 5). Derfor skal vi udregne hvad mange måder der er at opstille 6 bars og 5, som er $C(11,5) = 462$

Permutations with indistinguishable objects:

THEOREM 3

The number of different permutations of n objects, where there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type 2, ..., and n_k indistinguishable objects of type k , is

$$\frac{n!}{n_1! n_2! \cdots n_k!}.$$

Generating Permutations and Combinations

Generating Permutations:

Eksempel:

$$N = \{1, 2, 3, 4, 5, 6\}$$

Permutationen: 362541 - find næste permutation lexografisk.

Find a_j and a_{j+1} , where $a_j < a_{j+1}$, closest to the right most digit.

$a_j = 2$, $a_{j+1} = 5$. Find det tal tættest til højre for a_j som er større end a_j .

$$4 > a_j$$

Sæt 4 ind på a_j 's plads: 364251

Alle tal efter 4 skal ordnes lexografisk: 251 → 125:

364125

Næste permutation ville være:

$$a_j = 2, a_{j+1} = 5$$

Indsæt 5 på a_j 's plads:

364152

Generating Combinations:

EXAMPLE 5 Find the next larger 4-combination of the set $\{1, 2, 3, 4, 5, 6\}$ after $\{1, 2, 5, 6\}$.

Solution: The last term among the terms a_i with $a_1 = 1$, $a_2 = 2$, $a_3 = 5$, and $a_4 = 6$ such that $a_i \neq 6 - 4 + i$ is $a_2 = 2$. To obtain the next larger 4-combination, increment a_2 by 1 to obtain $a_2 = 3$. Then set $a_3 = 3 + 1 = 4$ and $a_4 = 3 + 2 = 5$. Hence the next larger 4-combination is $\{1, 3, 4, 5\}$. 

Lav en n-bitstring. $n-r+i \neq a_1 \dots a_2 \dots$ increment a_2 med 1, resten af tallene = $a_i + j - i$, som er $3 + \text{deres plads} - a_i$'s plads, så plads $4 = 3 + 4 - 2 = 5$. $\Rightarrow \{1, 3, 4, 5\}$

Discrete Probability - Chapter 7

Finite Probability

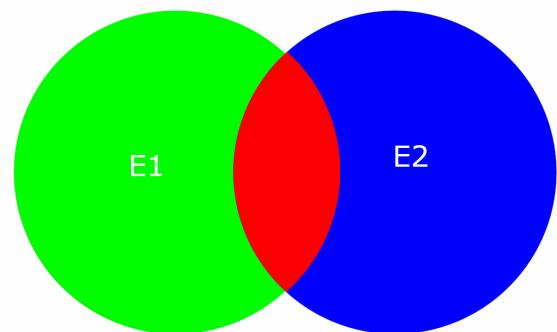
Definition 1 If S is a finite nonempty sample space of equally likely outcomes, and E is an event, that is, a subset of S , then the *probability* of E is $p(E) = \frac{|E|}{|S|}$.

Sample space, S , er mulige outcomes, event, E , er det event som møder vores constraints.

Eksempel: to terninger bliver slået, hvad er chancen for at summen er 7?

Sample space: $6^2 = 36$, fordi for hvert slag på den ene terning er der 6 muligheder på den anden terning.

Events: $\{(6,1), (5,2), (4,3), (3,4), (2,5), (1,6)\}$



$$P(E) = \frac{|E|}{|S|} = \frac{6}{36} = \frac{1}{6}$$

Probabilities of Complements and Unions of Events

THEOREM 1

Let E be an event in a sample space S . The probability of the event $\bar{E} = S - E$, the complementary event of E , is given by

$$p(\bar{E}) = 1 - p(E).$$

Proof: To find the probability of the event $\bar{E} = S - E$, note that $|\bar{E}| = |S| - |E|$. Hence,

$$p(\bar{E}) = \frac{|S| - |E|}{|S|} = 1 - \frac{|E|}{|S|} = 1 - p(E). \quad \triangleleft$$

Theorem 1 finder chancen for at et event ikke sker. Dette er lige med 1 minus chancen for at det sker.

Eksempel en random bit string med længden 8 bliver genereret. Hvad er chancen for at bit stringen har et 0 i sig? Her er det nemmere at kigge på hvad er chancen for at det ikke sker? Altså chancen for at alle 8 bit er 1'ere:

$$P(E) = 1 - P(\bar{E})$$

$$P(\bar{E}) = \frac{1}{2^8}$$

$$P(E) = 1 - \frac{1}{2^8} = \frac{255}{256}$$

THEOREM 2 Let E_1 and E_2 be events in the sample space S . Then

Let S be the sample space of an experiment with a finite or countable number of outcomes. We assign a probability $p(s)$ to each outcome s . We require that two conditions be met:

$$(i) \quad 0 \leq p(s) \leq 1 \text{ for each } s \in S$$

and

$$(ii) \quad \sum_{s \in S} p(s) = 1.$$

Hvad er chancen for at to events sker ud fra et samplespace. Chancen for det ene event plus chancen for det andet event minus chancen for at de begge sker. Dette gør man for at man undgår at tælle events to gange

$$|E_1 \cup E_2| = |E_1| + |E_2| - |E_1 \cap E_2|$$

$$\begin{aligned} p(E_1 \cup E_2) &= \frac{|E_1 \cup E_2|}{|S|} \\ &= \frac{|E_1| + |E_2| - |E_1 \cap E_2|}{|S|} \\ &= \frac{|E_1|}{|S|} + \frac{|E_2|}{|S|} - \frac{|E_1 \cap E_2|}{|S|} \\ &= p(E_1) + p(E_2) - p(E_1 \cap E_2) \end{aligned}$$

$$\sum_{s \in S} p(s) = 1 = p(E) + p(\bar{E}).$$

$$\text{Hence, } p(\bar{E}) = 1 - p(E).$$

Under Laplace's definition, by Theorem 2 in Section 7.1, we have

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

Probabilistic Reasoning - Monty Hall Puzzle

Altid byt når dør med zonk bliver åbnet. En tredjedels chance for at vælge rigtigt til at starte med og to tredjedels chance for at man vælger forkert.

Chancen for at man har valgt rigtigt til at starte med og dør nummer to bliver åbnet med zonk:

$$p(x \& y) = p(x|y) \cdot p(y)$$

$$= p(y \& x) = p(y|x) \cdot p(x)$$

$$\Rightarrow p(x|y) \cdot p(y) = p(y|x) \cdot p(x)$$

$$\Rightarrow p(x|y) = p(y|x) \cdot \frac{p(x)}{p(y)}$$

$p(y|x) = \frac{1}{2}$, der er 50% chance for at han åbner dør nummer to, når man har valgt dør 1

$$p(x) = \frac{1}{3}$$

$$p(y) = \frac{1}{2}$$

$$p(x|y) = \frac{1}{2} \cdot \frac{\frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}$$

Probability Theory

Probability Distribution er funktionen for alle outcomes i et sample space.

Definition 1

Suppose that S is a set with n elements. The *uniform distribution* assigns the probability $1/n$ to each element of S .

We now define the probability of an event as the sum of the probabilities of the outcomes in this event.

Definition 2

The *probability* of the event E is the sum of the probabilities of the outcomes in E . That is,

$$p(E) = \sum_{s \in E} p(s).$$

(Note that when E is an infinite set, $\sum_{s \in E} p(s)$ is a convergent infinite series.)

THEOREM 1

If E_1, E_2, \dots is a sequence of pairwise disjoint events in a sample space S , then

$$p\left(\bigcup_i E_i\right) = \sum_i p(E_i).$$

(Note that this theorem applies when the sequence E_1, E_2, \dots consists of a finite number or a countably infinite number of pairwise disjoint events.)

Unions:

Conditional Probability

Definition 3

Let E and F be events with $p(F) > 0$. The *conditional probability* of E given F , denoted by $p(E | F)$, is defined as

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

Chancen for at E sker hvis F sker. E givet F .

Example: Familie med to børn. Hvad er chancen for at de får to drenge hvis de allerede har en dreng?

F er eventet hvor familien får én dreng.

E er eventet hvor familien får to drenge.

$$E = \{BB\}$$

$$F = \{BB, GB, BG\}$$

Så i følge formlen:

$$p(E | F) = \frac{p(E \cap F)}{p(F)} = \frac{p(\{BB\})}{p(\{BB, GB, BG\})} = \frac{1/4}{3/4} = \frac{1}{3}, \text{ fordi der er } 4 \text{ muligheder i alt.}$$

Altså $S = \{BB, GG, BG, GB\}$

Independence



Suppose a coin is flipped three times, as described in the introduction to our discussion of conditional probability. Does knowing that the first flip comes up tails (event F) alter the probability that tails comes up an odd number of times (event E)? In other words, is it the case that $p(E | F) = p(E)$? This equality is valid for the events E and F , because $p(E | F) = 1/2$ and $p(E) = 1/2$. Because this equality holds, we say that E and F are **independent events**. When two events are independent, the occurrence of one of the events gives no information about the probability that the other event occurs.

Because $p(E | F) = p(E \cap F)/p(F)$, asking whether $p(E | F) = p(E)$ is the same as asking whether $p(E \cap F) = p(E)p(F)$. This leads to Definition 4.

Definition 4

The events E and F are *independent* if and only if $p(E \cap F) = p(E)p(F)$.

Example fra før: Familie får to børn, hvad er chancen for at de får to drenge hvis de ved at det ene barn er en dreng?

Fordi $p(E)$ er $1/4$, og $p(F) = 3/4$ så kan vi se at de IKKE er independent, fordi:

$$p(E \cap F) \neq p(E)p(F) \Rightarrow \frac{1}{4} \neq \frac{1}{4} \cdot \frac{3}{4}$$

Andet example: Tre børn i en familie: To events, E chancen for at få begge køn, F Chancen for at få højest 1 dreng:

$$S = \{BBB, BBG, BGB, GBB, GGB, GBG, BGG, GGG\}$$

$$E = \{BBG, BGB, GBB, GGB, GBG, BGG\}$$

$$F = \{GGB, GBG, BGG, GGG\}$$

$$E \cap F = \{GGB, GBG, BGG\}$$

$$p(E \cap F) = \frac{|E \cap F|}{|S|} = \frac{3}{8}$$

$$p(E) = \frac{6}{8} = \frac{3}{4}$$

$$p(F) = \frac{4}{8} = \frac{1}{2}$$

$$p(E)p(F) = \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8} = p(E \cap F)$$

Så F og E er independent

PAIRWISE AND MUTUAL INDEPENDENCE We can also define the independence of more than two events. However, there are two different types of independence, given in Definition 5.

Definition 5

The events E_1, E_2, \dots, E_n are *pairwise independent* if and only if $p(E_i \cap E_j) = p(E_i)p(E_j)$ for all pairs of integers i and j with $1 \leq i < j \leq n$. These events are *mutually independent* if $p(E_{i_1} \cap E_{i_2} \cap \dots \cap E_{i_m}) = p(E_{i_1})p(E_{i_2}) \dots p(E_{i_m})$ whenever $i_j, j = 1, 2, \dots, m$, are integers with $1 \leq i_1 < i_2 < \dots < i_m \leq n$ and $m \geq 2$.

Bernoulli Trials and Binomial Distribution

En Bernoulli trial er et eksperiment vi kan beskrive som har to outcomes, som vi beskriver med enten 1, for succes, eller 0 for fejl.

Hvis p er chancen for succes og q er chancen for fejl, så er $p + q = 1$

Hver trial er mutually independent, altså de afhænger ikke af hinanden.

Man kan beskrive sandsynligheden for n Bernoulli trials med en binomial distribution

THEOREM 2

The probability of exactly k successes in n independent Bernoulli trials, with probability of success p and probability of failure $q = 1 - p$, is

$$C(n, k)p^k q^{n-k}.$$

Outcome af en n Bernoulli trials, har vi n elementer, som enten er S, eller F. Der skulle være p^k S'er og resten skulle være F. Der er så $C(n, k)$ forskellige måder at opstille de S og F'er, som giver formlen:

$$C(n, k)p^k q^{n-k}$$

Eksempel:

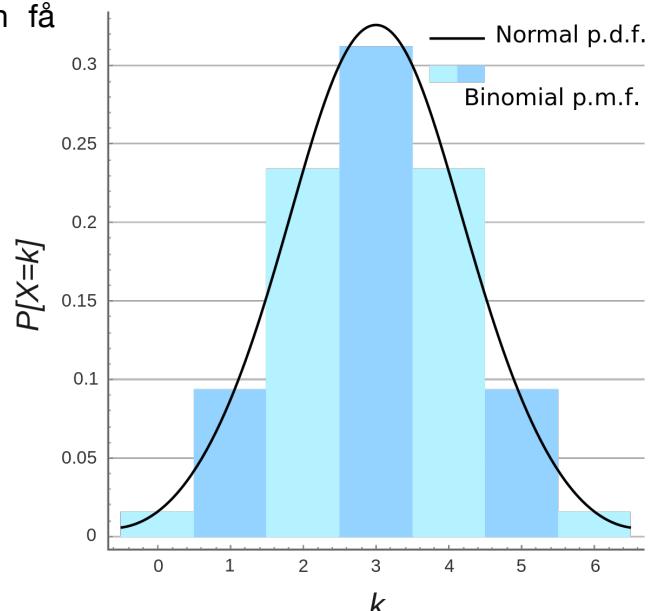
Vælger man en tilfældig dansk mand på alderen 80, er chancen for at han overlever endnu et år: 0,8

Ud af 20 mænd, hvad er så cachen for at X mænd overlever endnu et år?

Hvis man så vælger X til at være: 15, altså, man vælger 15 tilfældige mænd, så er chancen for at de alle overlever endnu et år:

$$p(X = 15) = \binom{20}{15} \cdot (0.80)^{15} \cdot (1 - 0.80)^{20-15} = \binom{20}{15} \cdot 0.0352 \cdot 0.0003$$

Hvis man gør det for alle X 'er, ville man få binomial distribution:



Random Variables

Definition 6

A *random variable* is a function from the sample space of an experiment to the set of real numbers. That is, a random variable assigns a real number to each possible outcome.

Definition 7

The *distribution* of a random variable X on a sample space S is the set of pairs $(r, p(X = r))$ for all $r \in X(S)$, where $p(X = r)$ is the probability that X takes the value r . (The set of pairs in this distribution is determined by the probabilities $p(X = r)$ for $r \in X(S)$.)

En random variable beskriver de outcomes der er ved at lave en tilfældig process, som f.eks. at flippe en mønt eller kaste en terning.

Outcomes for en mønt:

$$X = \begin{cases} 1 & \text{if heads} \\ 0 & \text{if tails} \end{cases}$$

Man kan videre udvide til at X er random variable af heads som kommer når man kaster en mønt 3 gange:

$p(X = 0) = 1/8$, som betyder der er 1/8 chance for at få ingen heads,
 $p(X = 2) = 3/8$, fordi der er tre ud af otte muligheder for at få to heads når man kaster tre terninger.

The Birthday Problem

For at finde ud af hvad chancen for at nogen mennesker i en gruppe af n mennesker deler fødselsdag, skal vi kigge på hvad chancen er for at ingen af dem deler deres fødselsdag.

Så chancen for at nogen deler fødselsdag:

$$p(s) = 1 - p(d)$$

Så får at finde $p(s)$, shared birthday. Skal vi finde chancen for at de har distinct $p(d)$:

Så hvis man sætter n til at være 2:

Så har første person:

En chance på: $\frac{365}{365}$ for at vælge en dag hvor ingen andre har fødselsdag:

Anden person har derfor en chance på: $\frac{364}{365}$ for at vælge en dag hvor ingen andre har fødselsdag.

Dette kan vi skrive ud generelt som:

$$\frac{365 \cdot 364}{365^2}$$

Skulle man udvide til n mennesker:

$$\frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot \dots \cdot (365 - n)}{365^n}$$

Tælleren kan omskrives til:

$$\frac{365!}{(365 - n)!}$$

Så hele udtrykket bliver til:

$$\frac{365!}{(365 - n)!} = \frac{365!}{365^n \cdot (365 - n)! \cdot 365^n}$$

Med den formel kan man så finde chancen for at n mennesker har forskellige fødselsdage. Så for at finde chancen for at nogen deler fødselsdag:

$$p(n) = 1 - \frac{365!}{(365 - n)! \cdot 365^n}$$

Monte Carlo Algorithms

En algoritme hvor tilfældighed spiller en rolle. De giver en approximation over noget ved at køre algoritmen flere gange med random værdi(er). Dette kan blive brugt til error estimation:

Eksempel fra bogen: Hvis en (computer) chip maker, skal finde ud af om et batch er testet for fejl, ville den lave stikprøver k antal gange. Hvis algoritmen finder en chip med fejl, ville den returnere "true" og stoppe. Hvis algoritmen kører k gange og ikke finder fejl, ville den stoppe og returnerer false. Hvilket betyder at der ikke blev fundet fejl, og at alle chips er blevet testet i det batch.

Så hvis algoritmen skal lave fejl så skal den vurdere et utedstet batch testet. Lad os sige at chancen for at maskinen finder en god chip i et utedstet batch er 0.9. Så ville chancen for fejl være 0.9^k . Så jo højere k bliver jo mindre er chancen for fejl.

The Probabilistic Method

I et set S af elementer, kan vi give alle elementer nogle properties (egenskaber). Vi kan bestemme hvad chancen er for at et element med en specific egenskab eksisterer i S .

THEOREM 3

THE PROBABILISTIC METHOD If the probability that an element chosen at random from a S does not have a particular property is less than 1, there exists an element in S with this property.

Ramsey number: Det mindste antal af mennesker der skal til for at der hverken eksisterer k sæt venner eller k sæt fjender.

Hvis man har et en complete graph af n nodes, ville der være $C(n,2)$ edges. Kan man finde en coloring, hvor der hverken eksistere k mutual friends eller k mutual enemies.

For et bruge den probabilistic metode skal vi først opsætte en passende probability space, og så vil vi finde sandsynligheden for det event.

Probability space vil være at vi har en komplet graf hvor vi skal farve hver edge entet rød eller blå, alt efter om de er fjender eller venner.

Så hvis vi kigger på en sub-graph a en graph med n nodes, S . $|S| = k$.

Her kigger vi på det event at alle edges i S er monochromatic. $E_s \doteq S$ is monochromatic $E \doteq \exists S$, which is monochromatic

Hvis $p(E) < 1$, betyder det at n er et lower bound. Altså der eksistere en coloring uden en monochromatic clique.

Så får at finde $p(E)$ skal vi finde $p(E_s)$, som er sandsynligheden for at alle edges i et bestemt subset har samme farve...

Da vi ved at der er $C(k,2)$ edges i S kan vi regne sandsynligheden for monochromatism til at være:

$$p(E_s) = \frac{1}{2^{\binom{k}{2}}} \cdot 2$$

Her kan vi finde $p(E)$ ved at kigge på alle subset:

$$\begin{aligned} p(E) &= \bigcup_{s \in G_n} p(E_s), \text{ ved brug af Boole's inequality} \\ &\leq \sum_{s \in G_n} p(E_s) \\ &= \binom{n}{k} \frac{1}{2^{\binom{k}{2}}} \cdot 2 \\ &= \binom{n}{k} 2 \cdot \left(\frac{1}{2}\right)^{k(k-1)/2} \end{aligned}$$

Hvis det er mindre end 1, så er n lower bound for $R(k, k)$

Bayes Theorem

Eksempel: Hvad er chancen for at en rød bold bliver taget fra 1. kasse når vi ved at bolden er rød.. Kasse 1 har 7 røde bolde og 2 grønne, kasse 2 har 3 røde bolde og 4 grønne.

$F \doteq$ picked from box 1

$\bar{F} \doteq$ picked from box 2

$E \doteq$ Red ball was picked

$\bar{E} \doteq$ Green ball was picked

Så det vi leder efter:

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

Vi ved at der er 7 røde bolde ud af 9 i første box, så:

$$p(E | F) = \frac{7}{9}$$

Og vi ved at der er 3 røde bolde ud af 7 i anden box:

$$p(E | \bar{F}) = \frac{3}{7}$$

Vi går ud fra at chancen for at vælge fra hvilken som helst box er ens så:

$$p(F) = p(\bar{F}) = \frac{1}{2}$$

Fra conditional probability ved vi:

$$p(E|F) = \frac{p(E \cap F)}{p(F)} \Leftrightarrow p(E \cap F) = p(E|F)p(F) = \frac{7}{9} \cdot \frac{1}{2} = \frac{7}{18}$$

Her kan vi finde for den ande kasse:

$$p(E|\bar{F}) = \frac{p(E \cap \bar{F})}{p(\bar{F})} \Leftrightarrow p(E \cap \bar{F}) = p(E|\bar{F})p(\bar{F}) = \frac{3}{7} \cdot \frac{1}{2} = \frac{3}{14}$$

Så skal vi bare finde $p(E)$:

Vi ved at E er defineret som at en rød bold bliver valgt fra enten kasse 1 eller kasse 2:

$$\begin{aligned} E &= (E \cap F) \cup (E \cap \bar{F}) \Rightarrow p(E) = p(E \cap F) + p(E \cap \bar{F}) \\ &= \frac{7}{14} + \frac{3}{14} = \frac{38}{63} \end{aligned}$$

Tilbage til første formel:

$$p(F|E) = \frac{p(F \cap E)}{p(E)} = \frac{7/18}{38/63} = \frac{49}{76} \approx 0.645$$

THEOREM 1

BAYES' THEOREM Suppose that E and F are events from a sample space S such that $p(E) \neq 0$ and $p(F) \neq 0$. Then

$$p(F|E) = \frac{p(E|F)p(F)}{p(E|F)p(F) + p(E|\bar{F})p(\bar{F})}.$$

Proof:

Vi ved at:

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

Ud fra conditional combinatorics:

$$p(E \cap F) = p(F|E)p(E), \text{ og } p(E \cap F) = p(E|F)p(F)$$

Sætter ligningerne lig hinanden:

$$p(F|E)p(E) = p(E|F)p(F)$$

Deler med $p(E)$ på hver side:

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

Så skal vi bevise at $p(E) = p(E|F)p(F) + p(E|\bar{F})p(\bar{F})$:

Som beskrevet før:

$$E = (E \cap F) \cup (E \cap \bar{F}), \text{ fordi: } F \cap \bar{F} = \emptyset$$

Hvilket betyder:

$$p(E) = p(E \cap F) + p(E \cap \bar{F})$$

Det første led:

$$p(E \cap F) = p(E|F)p(F)$$

Andet led:

$$p(E \cap \bar{F}) = p(E|\bar{F})p(\bar{F})$$

$$\Rightarrow p(E) = p(E|F)p(F) + p(E|\bar{F})p(\bar{F})$$

Så indsætter vi det på $p(E)$'s plads:

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

□

Dette kan udvides til Generalized Bayes Theorem:

THEOREM 2

GENERALIZED BAYES' THEOREM Suppose that E is an event from a sample space S and that F_1, F_2, \dots, F_n are mutually exclusive events such that $\bigcup_{i=1}^n F_i = S$. Assume that $p(E) \neq 0$ and $p(F_i) \neq 0$ for $i = 1, 2, \dots, n$. Then

$$p(F_j | E) = \frac{p(E | F_j)p(F_j)}{\sum_{i=1}^n p(E | F_i)p(F_i)}.$$

I stedet for at vælge bolde fra to kasser, hvad med 50 kasser?

Hvad er chancen for at man har valgt en rød bold fra kasse 30?

Lille bevis til nævneren:

$p(E)$ skal udvides til at have alle muligheder fra alle kasser:

$$p(E) = \sum_{i=1}^n p(E \cap F_i)$$

Bruger samme teknik som før på alle led:

$$p(E \cap F_i) = p(E|F_i)p(F_i)$$

□

$$\Rightarrow p(E) = \sum_{i=1}^n p(E|F_i)p(F_i)$$

Expected Value and Variance

Expected Value beskrives som den et vægtet gennemsnit af alle værdier en random værdi kan tage. Det kan bruges til at finde svar på spørgsmål som, hvor mange heads er expected efter 100 kast? Og hvor mange comparisons skal vi forvente en linear search algoritme laver?

Definition 1

The *expected value*, also called the *expectation* or *mean*, of the random variable X on the sample space S is equal to

$$E(X) = \sum_{s \in S} p(s)X(s).$$

The *deviation* of X at $s \in S$ is $X(s) - E(X)$, the difference between the value of X and the mean of X .

$X(s)$ er de værdier som X kan tage ud fra sample space S . $p(s)$ er chancen for at vi X tager den værdi.

Eksempel 1: Hvad er expected value af et terningkast?

$$X(s) = \{1, 2, 3, 4, 5, 6\}$$

$$p(s) = \frac{1}{6}$$

$$\begin{aligned} E(X) &= \sum_{s \in S} p(s)X(s) = \sum_{s \in S} \frac{1}{6} \cdot s = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} \\ &= \frac{7}{2} \end{aligned}$$

Definition 1 kan skrives videre til:

THEOREM 1

If X is a random variable and $p(X = r)$ is the probability that $X = r$, so that $p(X = r) = \sum_{s \in S, X(s)=r} p(s)$, then

$$E(X) = \sum_{r \in X(S)} p(X = r)r.$$

Dette er en mere generel formel, Chancen for at X bliver r , ganger man med værdien selv. Så fra vores tidligere eksempel. Chancen for at terningen bliver 3, alts $p(X = 3)$ er stadig $1/6$, men vi ganger det med r , altså 3 for at få den mere generelle formel.

THEOREM 2

The expected number of successes when n mutually independent Bernoulli trials are performed, where p is the probability of success on each trial, is np .

Proof: ved hjælp af binomial theorem

$$\begin{aligned}
 E(X) &= \sum_{k=1}^n k \cdot p(X=k) \\
 &= \sum_{k=0}^n k \cdot c(n, k)p^k q^{n-k}, \text{ ved brug af theorem 2 for Bernoulli trials} \\
 &= \sum_{k=0}^n n \cdot c(n-1, k-1)p^k q^{n-k}, \text{ bevist i CAS} \\
 &= np \sum_{k=1}^n c(n-1, k-1)p^{k-1}q^{n-k}, \text{ udfaktoriseret np} \\
 &= np \sum_{j=0}^{n-1} c(n-1, j)p^j q^{n-(j+1)}, j = k - 1 \\
 &= np \cdot (p + q)^{n-1}, \text{ binomial theorem } \Rightarrow (x + y)^n = \sum_{k=0}^n \binom{n}{k} x^k y^{n-k} \\
 &= np, \text{ da } (p + q)^{n-1} = 1
 \end{aligned}$$

□

Linearity of Expectations

Expected values af Random variables er linear hvilket betyder at:

$$E(aR + bS) = aE(R) + bE(S), \text{ for random variables R,S og konstanter a,b}$$

Proof:

$$\begin{aligned}
 E[aA + bB] &= \sum_{x \in S} (aA(x) + bB(x)) \cdot p(x) \\
 &= a \left(\sum_{x \in S} A(x)p(x) \right) + b \left(\sum_{x \in S} B(x)p(x) \right) \\
 &= aE[A] + bE[B]
 \end{aligned}$$

□

Sum reglen for expectations holder både hvis værdierne er dependent eller independent

Eksempel: expected #heads after n flips:

$$X_i = \begin{cases} 1 & \text{if heads} \\ 0 & o.w \end{cases}$$

$$\#heads = \sum_{j=1}^n X_j \Rightarrow E[X_i] = \sum_{j=1}^n E[X_j] = n \cdot p(X_i) = n \cdot \frac{1}{2}$$

Eksempel: for dependent værdier. Hat check, med n mænd:

$$p(X) = \frac{1}{n}$$

$$X_i = \begin{cases} 1 & i'th man got hat back \\ 0 & o.w \end{cases}$$

$$E[X_i] = \sum_{j=1}^n E[X_j] = n \cdot \frac{1}{n} = 1$$

Produktreglen: værdierne skal være **independent!**

$$\begin{aligned} E[AB] &= \sum_{a,b \in S} ab \cdot p(A = a \wedge B = b) \\ &= \sum_{a,b \in S} ab \cdot p(A = a) \cdot p(B = b) \\ &= \sum_{a \in S} \sum_{b \in S} ab \cdot p(A = a) \cdot p(B = b) \\ &= \sum_{a \in S} \left(a \cdot p(A = a) \sum_{b \in S} b \cdot p(B = b) \right) \\ &= \sum_{a \in S} a \cdot p(A = a) \cdot \sum_{b \in S} b \cdot p(B = b) \\ &= E[A] \cdot E[B] \end{aligned}$$

□

Geometric Distribution

Geometric distribution ser vi når vi kigger på hvor mange gange man skal gøre noget, men en hvis sandsynlighed, før at der sker en ting. f.eks. hvor mange gange man skal lave et eksperiment før det lykkes, hvor mange gange man kan bruge et produkt før det går i stykker osv.

Kigger vi på expected antal gange vi skal kaste en mønt før vi får heads. Lad os sige at det tager n kast. Så ville der være $n-1$ kast som er tails hvor det sidste så er heads, det kan vi opstille som:

$$(1 - p)^{n-1} p$$

Da $1-p$ er chancen for tails.

Så for at finde expected value kan vi bruge formlen for geometric distribution:

$$E(X) = \sum_{j=1}^{\infty} j \cdot p(X=j) = \sum_{j=1}^{\infty} j(1-p)^{j-1} p = p \sum_{j=1}^{\infty} j(1-p)^{j-1} = p \cdot \frac{1}{p^2} = \frac{1}{p}$$

THEOREM 4

If the random variable X has the geometric distribution with parameter p , then $E(X) = 1/p$.

Variance

Variance bruges til at fortælle hvor udbredt vores værdier er. Expected value fortæller hvad mean (gennemsnit) er, men det siger intet om hvor stort et span, værdierne har.

Definition 4

Let X be a random variable on a sample space S . The *variance* of X , denoted by $V(X)$, is

$$V(X) = \sum_{s \in S} (X(s) - E(X))^2 p(s).$$

That is, $V(X)$ is the weighted average of the square of the deviation of X . The *standard deviation* of X , denoted $\sigma(X)$, is defined to be $\sqrt{V(X)}$.

For at finde variance finder man den gennemsnitlige afstand fra mean.

Standard deviation, som er denotert som σ , er kvadratroden af variancen. Den forklarer hvor langt væk fra værdierne er fra mean. Dette kan bruges til at kigge på variance af andre værdier, og så sammenligne dem. Hvis et sæt af værdier har en standard deviation på 20, og et andet sæt har en på 2, vil det sige, gennemsnitligt, er værdierne i første sæt 10 gange længere fra hinanden i forhold til det andet sæt.

THEOREM 6

If X is a random variable on a sample space S , then $V(X) = E(X^2) - E(X)^2$.

Proof:

$$\begin{aligned}
 V(X) &= \sum_{s \in S} (X(s) - E[X])^2 p(X) && \text{By definition} \\
 &= \sum_{s \in S} (X(s)^2 + E[X]^2 - 2E[X] \cdot X(s)) \cdot p(X) && \text{Kvadratsætning udvides} \\
 &= \sum_{s \in S} X(s)^2 p(X) - 2E[X] \sum_{s \in S} X(s)p(X) + E[X]^2 \sum_{s \in S} p(X), \text{ udfoldet og pænere opstillet} \\
 &= E[X^2] - 2E[X] \cdot E[X] + E[X]^2 \cdot 1 && \text{Definition på expected value} \\
 &= E[X^2] - E[X]^2
 \end{aligned}$$

□

THEOREM 7

BIENAYMÉ'S FORMULA If X and Y are two independent random variables on a sample space S , then $V(X+Y) = V(X) + V(Y)$. Furthermore, if $X_i, i = 1, 2, \dots, n$, with n a positive integer, are pairwise independent random variables on S , then $V(X_1 + X_2 + \dots + X_n) = V(X_1) + V(X_2) + \dots + V(X_n)$.

COROLLARY 1

If X is a random variable on a sample space S and $E(X) = \mu$, then $V(X) = E((X - \mu)^2)$.

Markov's inequality

Hvad er chancen for at en random variable er større end et eller anden værdi:

$$p(X \geq a) \leq \frac{E[X]}{a}$$

Hvis vi starter med at sætte X op som indicator function:

$$X_i = \begin{cases} 1 & X \geq a \\ 0 & o.w \end{cases}$$

Dette laver en step function som bliver i lige så snart vores værdi kommer over a .

Vi kan forlænge denne step funktion.

$$aX_i = \begin{cases} a & X \geq a \\ 0 & o.w \end{cases}$$

Dette fortæller os at hvis vores indicator function er 0, så må det holde at X er større eller lig med:

$$aX_i = 0 \Rightarrow 0 \leq X$$

Samtidig:

$$aX_i = a \Rightarrow a \leq X$$

Det må betyde at indicator function altid er mindre end, eller lig med, X:

$$aX_i \leq X$$

Her skriver vi vores indicator function ud som expected value:

$$aE[X_i] \leq E[X]$$

$$= a \cdot p(X_i) \leq E[X]$$

$$= a \cdot p(X \geq a) \leq E[X]$$

$$= p(X \geq a) \leq \frac{E[X]}{a}$$

Da det er en indicator variable går fra 0, 1 kan vi nemt finde sandsynligheden

Chebyshev's inequality

Her kigger vi på hvad chancen er for at en random variable i vores sample space har en afstand fra gennemsnittet der er større end en eller anden værdi.

THEOREM 8

CHEBYSHEV'S INEQUALITY Let X be a random variable on a sample space S with probability function p . If r is a positive real number, then

$$p(|X(s) - E(X)| \geq r) \leq V(X)/r^2.$$

Ud fra Markovs inequality, kan vi bare erstatte X med $|X-\mu|$, da de begge er Random variables:

$$p(|X - \mu| \geq a) \leq \frac{E[|X - \mu|]}{a}$$

Hvis vi kigger på venstre side af uligheden:

$$p(|X - \mu| \geq a)$$

Så er det trivielt at uanset hvilken værdi X har, så ville det holde at:

$$p(|X - \mu| \geq a) = p((X - \mu)^2 \geq a^2)$$

Dette betyder at vi kan sætte det ind på venstre side af ligningen:

$$p((X - \mu)^2 \geq a^2) \leq \frac{E[(X - \mu)^2]}{a^2}$$

$$\Rightarrow p(|X - \mu| \geq a) \leq \frac{E[(X - \mu)^2]}{a^2}$$

$$= p(|X - \mu| \geq a) \leq \frac{V(X)}{a^2}, \text{ ud fra collorary 1 s. 513}$$

□

Advanced Counting Techniques - Chapter 8

Linear Recurrence Relations - Modeling

Linear recurrence relations can be used to a lot of things, first of all: modelling.

So if we look at the towers of Hanoi. We can write a recurrence relation to describe how to solve the puzzle.

Der er n diske, og H_n er så mængden af træk der skal til for at løse den.

$n=1$		$T(1) = 1$
$n=2$		$T(2) = 3$
$n=3$		$T(3) = 2 \cdot T(2) + 1$

Så ud fra dette kan vi konkludere at:

$$T_n = 2T_{n-1} + 1$$

For at løse relationen, altså finde en måde at fjerne den rekursive del af relationen:

$$\begin{aligned}H_n &= 2H_{n-1} + 1 \\&= 2(2H_{n-2} + 1) + 1 = 2^2H_{n-2} + 2 + 1 \\&= 2^2(2H_{n-3} + 1) + 2 + 1 = 2^3H_{n-3} + 2^2 + 2 + 1 \\\vdots \\&= 2^{n-1}H_1 + 2^{n-2} + 2^{n-3} + \dots + 2 + 1 \\&= 2^{n-1} + 2^{n-2} + \dots + 2 + 1 \\&= 2^n - 1.\end{aligned}$$

Algorithms and recurrence:

Dynamisk programmering er brugbart til at udvikle algoritmer. Algoritmer der deler et problem op i flere sub-problemer. Merge sort er et eksempel på en algoritme. Recurrence relations kan bruges til at finde den overordnet løsning til en masse sub-problemer.

Solving linear homogeneous recurrence relations

Definition 1

A *linear homogeneous recurrence relation of degree k with constant coefficients* is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k},$$

where c_1, c_2, \dots, c_k are real numbers, and $c_k \neq 0$.

For at en recurrence relation skal være linear betyder det at der ikke må forkomme opløftet udtryk, og for at en relation skal være homogeneous skal definition 1 gælde. Det vil sige at $H_n = 2H_{n-1} - 1$ ikke er homogeneous pga. "-1".

THEOREM 1

Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1r - c_2 = 0$ has two distinct roots r_1 and r_2 . Then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2}$ if and only if $a_n = \alpha_1r_1^n + \alpha_2r_2^n$ for $n = 0, 1, 2, \dots$, where α_1 and α_2 are constants.

Proof: $a_n = c_1a_{n-1} + c_2a_{n-2} \Leftrightarrow a_n = \alpha_1r_1^n + \alpha_2r_2^n$

Bevis at det virker i begge retninger.

Hvis vi går ud fra at: $a_n = \alpha_1r_1^n + \alpha_2r_2^n$

r_1 og r_2 er roots af $r^2 - c_1r - c_2 = 0$

$$\Rightarrow r_1^2 - c_1r_1 - c_2 = 0 \Rightarrow r_1^2 = c_1r_1 + c_2 \quad \& \quad r_2^2 - c_1r_2 - c_2 = 0 \Rightarrow r_2^2 = c_1r_2 + c_2$$

Herfra:

$$\begin{aligned} c_1a_{n-1} + c_2a_{n-2} &= c_1(\alpha_1r_1^{n-1} + \alpha_2r_2^{n-1}) + c_2(\alpha_1r_1^{n-2} + \alpha_2r_2^{n-2}) \\ &= c_1\alpha_1r_1^{n-1} + c_1\alpha_2r_2^{n-1} + c_2\alpha_1r_1^{n-2} + c_2\alpha_2r_2^{n-2} \\ &= \alpha_1r_1^{n-2}(c_1r_1 + c_2) + \alpha_2r_2^{n-2}(c_1r_2 + c_2) \\ &= \alpha_1r_1^{n-2}(r_1^2) + \alpha_2r_2^{n-2}(r_2^2) \\ &= \alpha_1r_1^2 + \alpha_2r_2^2 = a_n \end{aligned}$$

Så for at vise at det virker den anden vej:

$$a_0 = C_0 = \alpha_1 + \alpha_2$$

$$a_1 = C_1 = \alpha_1r_1 + \alpha_2r_2$$

Hvis det holder så ved vi at det er en løsning for a_0 og a_1 . Da der kun eksisterer en løsning til en recurrence relation af graden to, med to initial conditions.

Linear homogeneous recurrence relations kan løses systematisk:

Først opsætter vi characteristic equation

fra eksemplet $a_n = a_{n-1} + 2a_{n-2}$, $a_1 = 7$ & $a_0 = 2$:

$$a_n = a_{n-1} + 2a_{n-2} = a_n - a_{n-1} - 2a_{n-2}$$

$$\Rightarrow x^2 - x - 2 = 0$$

Her løser vi rødderne for den karakteristiske ligning:

$$x^2 - x - 2 = 0 \Rightarrow x = -1 \quad \& \quad x = 2$$

Her putter vi de to værdier ind i formlen for de to kendte værdier: Theorem 1.

$$a_0 = 2 = \alpha_1 \cdot (-1^0) + \alpha_2 \cdot (2^0)$$

$$a_1 = 7 = \alpha_1 \cdot (-1^1) + \alpha_2 \cdot (2^1)$$

Løs ligningssystem for at finde de to ubekendte:

$$\alpha_1 = -1 \text{ and } \alpha_2 = 3$$

Indsæt på de respektive pladser:

$$a_1 = (-1) \cdot (-1^1) + 3 \cdot (2^1) = 7$$

Her kan vi komme med den generelle løsning:

$$a_n = (-1) \cdot (-1^n) + 3 \cdot 2^n$$

Linear non homogeneous recurrence relations

Non homogeneous recurrence relations har formen:

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

Den første del af relationen er Associated del, og den er homogeneous

THEOREM 5

If $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous linear recurrence relation with constant coefficients

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

then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}.$$

Proof:

$$\text{Den particular solution: } a_n^{(p)} = c_1 a_{n-1}^{(p)} + c_2 a_{n-2}^{(p)} + \dots + c_k a_{n-k}^{(p)} + F(n)$$

Antag at b_n er en anden løsning til vores non-homogeneous recurrence relation:

$$b_n = c_1 b_{n-1} + c_2 b_{n-2} + \dots + c_k b_{n-k} + F(n). \text{ Fratræk den første ligning fra den anden ligning og vi får:}$$

$$b_n - a_n^{(p)} = c_1(b_{n-1} - a_{n-1}^{(p)}) + c_2(b_{n-2} - a_{n-2}^{(p)}) + c_3(b_{n-3} - a_{n-3}^{(p)}) + \dots + c_k(b_{n-k} - a_{n-k}^{(p)})$$

Hvilket betyder at $b_n - a_n^{(p)}$ er en løsning for den associated del af recurrence relationen, som er homogeneous. Så den kalder vi $a_n^{(h)}$

$$\text{Så: } a_n = a_n^{(h)} + a_n^{(p)}$$

Løsning af non-homogeneous recurrence relations:

$$a_n = 5a_{n-1} - 6a_{n-2} + 7^n$$

Opdeling:

$$a_n = a_n^{(h)} + a_n^{(p)}$$

Den homogeneous del:

$$a_n^{(h)} = 5a_{n-1} - 6a_{n-2}$$

Dette udregnes som før:

$$a_n^{(h)} - 5a_{n-1} + 6a_{n-2} = 0$$

$$\Rightarrow x^2 - 5x + 6 = 0 \Rightarrow x = 2 \text{ og } x = 3$$

$$\Rightarrow a_n^{(h)} = \alpha_1 2^n + \alpha_2 3^n$$

Nu skal løsningen for den particular del findes:

$$a_n^{(p)} = 5a_{n-1} - 6a_{n-2} + 7^n$$

Her ser vi ud fra denne tabel hvad $F(n)$, 7^n , skal substitueres med:

Gætter på: $A7^n$

Substituering af alle an:

$$A7^n = 5A7^{n-1} - 6A7^{n-2} + 7^n$$

Ud fakturering af det mindste led, 7^{n-2} :

$$A7^2 = 5A7^1 - 6A + 7^2$$

$$\Rightarrow 49A = 35A - 6A + 49A$$

$$\Rightarrow A = \frac{49}{20}$$

Så har vi svaret på vores particular del:

$$a_n^{(p)} = \frac{49}{20} 7^n$$

Så har vi et udtryk for alle løsninger:

$$a_n = \alpha_1 2^n + \alpha_2 3^n + \frac{49}{20} 7^n$$

Og hvis vi havde nogle initial conditions, kunne vi opstille et lignings system og finde α_1 og α_2 .

$f(n)$	$a_n^{(p)}$
c	A
n	$A_1 n + A_0$
n^2	$A_2 n^2 + A_1 n + A_0$
r^n	$A r^n$

Inclusion-Exclusion

Hvis vi gerne vil tælle mængden af kvinder og førsteårsstuderende i et rum, kan vi risikere at tælle den samme flere gange, hvis vi først tæller den ene grupper og så den anden. Vi skal vide hvor mange kvinderne er førsteårsstuderende.

For at sørge for ikke at tælle dobbelt følger vi denne formel, for to set:

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

Eksempel:

Hvor mange elever får en bachelor i enten datalogi eller matematik, 25 datalogi, 13 matematik og 8 tager to-faglig, så:

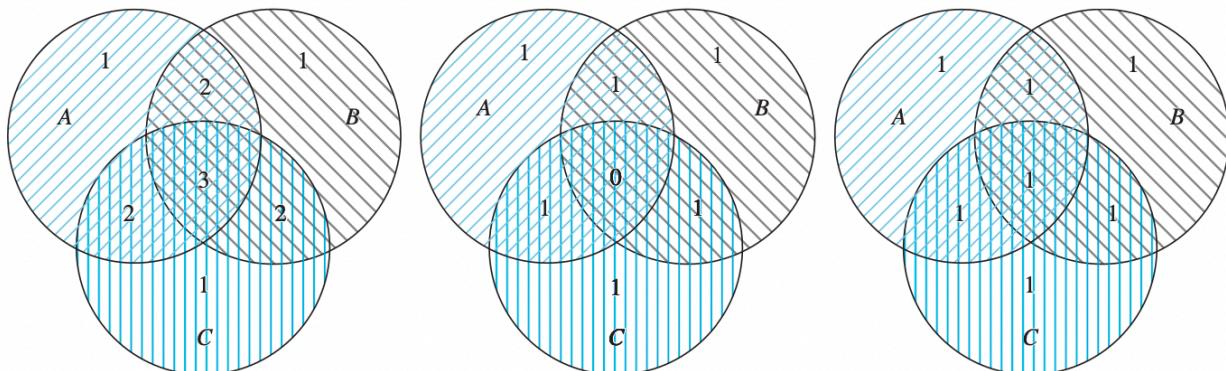
$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$\Rightarrow 25 + 13 - 8 = 30$$

Hvis der eksisterer tre set som skal tælles, så har vi:

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

Da vi skal sørge for ikke at trække den samme fra to gange når vi trækker fra.



Udfra dette kan vi opstille en generel formel for alle antal sets:

THEOREM 1

THE PRINCIPLE OF INCLUSION-EXCLUSION Let A_1, A_2, \dots, A_n be finite sets. Then

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_n| &= \sum_{1 \leq i \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| \\ &\quad + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n+1} |A_1 \cap A_2 \cap \dots \cap A_n|. \end{aligned}$$

Hvis vi har elementet a i r forskellige set. I det første udtryk på højresiden bliver a talt $C(r,1)$ gang, og i det andet udtryk bliver det talt $C(r,2)$ gang, og summationen med m elementer bliver det talt $C(r,m)$ gange. Og derfor ved vi at elementet bliver talt præcist:

$$C(r,1) - C(r,2) + C(r,3) - C(r,4) + \dots + (-1)^{r+1} C(r,r), \text{ gange}$$

Hvis vi gerne vil udregne den mængde, har vi ud fra Collorary 2 i 6.4:

$$C(r,0) - C(r,1) + C(r,2) - C(r,3) + \dots + (-1)^r C(r,r) = 0$$

So isoler $C(r,0)$:

$$0 = C(r,0) = C(r,1) - C(r,2) + C(r,3) - C(r,4) + \dots + (-1)^{r+1} C(r,r)$$

Alternate Form:

$$N(c_1) = \# \text{ af elementer som tilfredsstiller } c_1$$

$$N(\bar{c}_1 \bar{c}_2) = \# \text{ af elementer som IKKE tilfredsstiller } c_1 \text{ og } c_2$$

$$N(\overline{c_1 c_2}) = \# \text{ af elementer som IKKE tilfredsstiller } c_1 \text{ eller } c_2$$

$$N(\bar{c}_1) = N - N(c_1), \text{ hvor } N \text{ er universet, som kendt fra set-teori}$$

Eksempel:

Antallet af positive heltal $n \leq 100$ som IKKE går op i 2,3 eller 5

$$c_1 : n \text{ går op i } 2$$

$$c_2 : n \text{ går op i } 3$$

$$c_3 : n \text{ går op i } 5$$

$$N(\bar{c}_1 \bar{c}_2 \bar{c}_3) = N - N(c_1) - N(c_2) - N(c_3) + N(c_1 c_2) + N(c_2 c_3) + N(c_1 c_3) - N(c_1 c_2 c_3)$$

$$= 100 - \left\lfloor \frac{100}{2} \right\rfloor - \left\lfloor \frac{100}{3} \right\rfloor - \left\lfloor \frac{100}{5} \right\rfloor + \left\lfloor \frac{100}{2 \cdot 3} \right\rfloor + \left\lfloor \frac{100}{2 \cdot 5} \right\rfloor + \left\lfloor \frac{100}{3 \cdot 5} \right\rfloor - \left\lfloor \frac{100}{2 \cdot 3 \cdot 5} \right\rfloor$$

Sieve of Eratosthenes

Mængden af primal kan findes lidt på samme måde som eksemplet ovenover.

Vi bruger primtal under 10, da de andre primtal kan findes ud fra dem som ikke går op i de fire:

$$c_1 : n \text{ går op i } 2$$

$$c_2 : n \text{ går op i } 3$$

$$c_3 : n \text{ går op i } 5$$

$$c_4 : n \text{ går op i } 7$$

Antal af primes over 1 og under 100 er:

$4 + N(\bar{c}_1 \bar{c}_2 \bar{c}_3 \bar{c}_4)$, da de fire første primtal også skal tælles med.

Så for at finde resten:

$$\begin{aligned}N(\bar{c}_1 \bar{c}_2 \bar{c}_3 \bar{c}_4) &= N - N(c_1) - N(c_2) - N(c_3) - N(c_4) \\&+ N(c_1 c_2) + N(c_1 c_3) + N(c_1 c_4) + N(c_2 c_3) + N(c_2 c_4) + N(c_3 c_4) \\&- N(c_1 c_2 c_3) - N(c_2 c_3 c_4) - N(c_1 c_2 c_4) - N(c_1 c_3 c_4) \\&+ N(c_1 c_2 c_3 c_4)\end{aligned}$$

Derangements - Hatcheck



A **derangement** is a permutation of objects that leaves no object in its original position. To solve the problem posed in Example 4 we will need to determine the number of derangements of a set of n objects.

Antal af permutationer af hattene er $n!$.

Vi skal finde antallet af de permutationer hvor ingen af personer får deres hat tilbage. Dette kaldet derangement

For derangements af n objekter:

$$\begin{aligned}D_n &= S_0 - S_1 + S_2 - \dots + (-1)^n S_n, \text{ som kendt fra almindelig inclusion-exclusion} \\&= n! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! - \dots + \binom{n}{n} 0!, \text{ bevis fra theorem 1.8.5} \\&= n! - \frac{n!(n-1)!}{1!(n-1)!} + \frac{n!(n-2)!}{2!(n-2)!} - \dots + 1 \\&= n! - \frac{n!}{1!} + \frac{n!}{2!} - \dots + \frac{n!}{n!} \\&= n! \left[1 - \frac{1}{1!} + \frac{1}{2!} - \dots + (-1)^n \frac{1}{n!} \right]\end{aligned}$$

Så for at finde chancen for at vi rammer en derangement ud fra alle mulige permutationer:

$$\frac{D_n}{n!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \dots + (-1)^n \frac{1}{n!}$$

Randomized Algorithms - Kleinberg & Tardos

Contention Resolution

Hvis vi har en række processor som vil access en database, men kun én kan ad gangen så opdeler vi accessing i runder og får alle processor til at access randomt.

$A[i, t]$ betyder access process i ved runde t . Chancen for at en process med success accesser databasen, altså uden nogen andre gør:

$$S[i, t] = A[i, t] \cap \left(\bigcap_{j \neq i} \overline{A[j, t]} \right)$$

Fordi alle events er uafhængige af hinanden kan vi gange alle events sammen:

$$\Pr[S[i, t]] = \Pr[A[i, t]] \cdot \prod_{j \neq i} \Pr[\overline{A[j, t]}] = p(1-p)^{n-1}$$

Her gælder det om at finde den bedst mulige p , her kan vi ikke vælge 1 eller 0, for ellers ville ingen komme igennem. Hvis vi tager den afledte funktion af $p(1-p)^{n-1}$ ser vi at den har et maximum i $1/n$, som er den bedste sandsynlighed.

$$\Pr[S[i, t]] = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1}$$

Dette er asymptotisk lige med $\Theta(1/n)$

Chancen for at en process aldrig kommer til databasen:

$$\Pr[\mathcal{F}[i, t]] = \Pr\left[\bigcap_{r=1}^t \overline{S[i, r]}\right] = \prod_{r=1}^t \Pr[\overline{S[i, r]}] = \left[1 - \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1}\right]^t$$

Asymptotisk er den bounded mellem $1/e$ (upper) og $1/2e$ (lower), så:

$$\Pr[\mathcal{F}[i, t]] = \prod_{r=1}^t \Pr[\overline{S[i, r]}] \leq \left(1 - \frac{1}{en}\right)^t$$

Chancen for at en process ikke når igennem er upper bounded ved e^{-1} , så hvis vi lægger en factor til t , så vil chancen for at P aldrig når igennem pludselig meget lavere.

$$\Pr[\mathcal{F}[i, t]] \leq \left(1 - \frac{1}{en}\right)^t = \left(\left(1 - \frac{1}{en}\right)^{\lceil en \rceil}\right)^{c \ln n} \leq e^{-c \ln n} = n^{-c}.$$

Her kan vi se at chancen bliver bounded med n^{-c} .

(13.1)

- (a) The function $\left(1 - \frac{1}{n}\right)^n$ converges monotonically from $\frac{1}{4}$ up to $\frac{1}{e}$ as n increases from 2.
- (b) The function $\left(1 - \frac{1}{n}\right)^{n-1}$ converges monotonically from $\frac{1}{2}$ down to $\frac{1}{e}$ as n increases from 2.

Using (13.1), we see that $1/(en) \leq \Pr[\mathcal{S}[i, t]] \leq 1/(2n)$, and hence $\Pr[\mathcal{S}[i, t]]$ is asymptotically equal to $\Theta(1/n)$.

Hvor mange runder skal vi så have før chancen for at alle processor har været igennem mindst en gang?

Her bruger vi union bound da processors ikke længere er independent.

(13.2) (The Union Bound) Given events $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n$, we have

$$\Pr\left[\bigcup_{i=1}^n \mathcal{E}_i\right] \leq \sum_{i=1}^n \Pr[\mathcal{E}_i].$$

E er en dårlig event i dette tilfælde. Vi vil gerne have upper bound på de dårlige events, så hvis vi har:

$F_t = \bigcup_{i=0}^n F[i, t]$ Ft er det event som sker hvis en af de andre events sker, altså det event at ikke alle er kommet igennem.

Så kan vi upper bound det med union bound:

$$\Pr[F_t] \leq \sum_{i=1}^n \Pr[F[i, t]]$$

Hvis vi sætter t til at være $t = \text{root}(en) \cdot (c \ln n)$, så kan vi upper bound sandsynligheden for Ft med n^{-c} .

Så hvis: $t = 2\lceil en \rceil \ln n$

$$\Pr[\mathcal{F}_t] \leq \sum_{i=1}^n \Pr[\mathcal{F}[i, t]] \leq n \cdot n^{-2} = n^{-1},$$

(13.3) With probability at least $1 - n^{-1}$, all processes succeed in accessing the database at least once within $t = 2\lceil en \rceil \ln n$ rounds.

Så hvis sandsynligheden for access er n^{-1} , så kommer alle processer igennem

Global Minimum Cut

Finde global min-cut i en UNDIRECTED graph som ikke har sink eller source nodes. Her bruger vi flow network min-cut metoden til at finde min-cut, som er den mindste antal edges vi kan fjerne for at opdele grafen G i to.

Metoden:

Vi udskifter først alle edges med to parallel edges og sætter deres capacity til 1

Herfra vælger vi to arbitrære nodes som s og t, og udfører min-cut på G' .

For at finde det globale min-cut, sætter vi s som en konstant node, og sætter, på skift, t til at være alle andre nodes, så $n-1$ computations for at finde global min-cut

Contraction Algorithm

Hurtigere algoritm til at finde global min-cut. Starter med at vælge en random edge $e=(u, v)$, og contracter den, altså laver en ny sammensat node med u og v. Alle edges som også var mellem u og v bliver slettet.

Super nodes gemmer information omkring hvilke nodes den indeholder fra G, og så contracter den rekursivt indtil der kun er to super notes tilbage, som skaber en opdeling af notes fra G

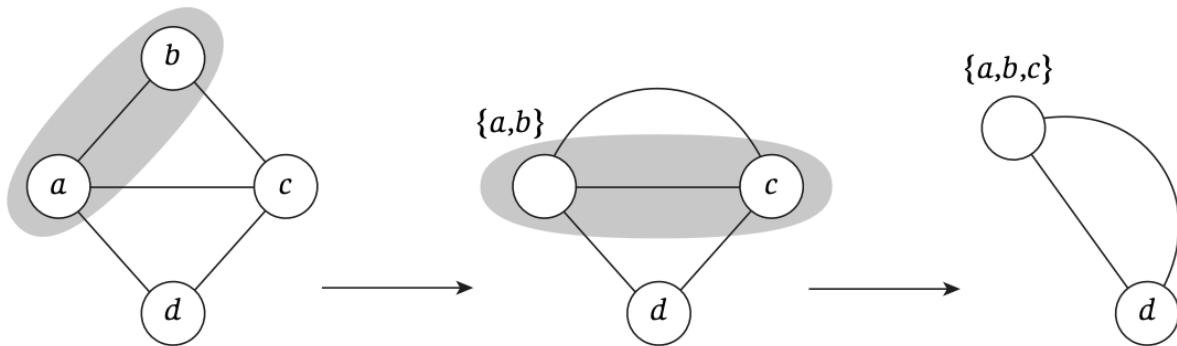


Figure 13.1 The Contraction Algorithm applied to a four-node input graph.

Dette er monte carlo algoritm, så der er en chance for at den ikke finder global min-cut.

(13.5) *The Contraction Algorithm returns a global min-cut of G with probability at least $1/\binom{n}{2}$.*

Proof:

Global min-cut af G, (A,B) har størrelsen k. Altså k edges som har den ene ende i A og den anden i B. F er set med de k edges

Vi vil give en lower bound på chancen for at algoritmen returnerer (A, B).

Hvis vi kigger på første step. Hvis en edge i F bliver contracted så har algoritmen fejlet, så hvis en edge som ikke er i F bliver contracted er chancen for at returnerer (A, B) der stadig.

Så vi vil gerne finde en upper bound på chancen for at en edge i F bliver contracted, og for det skal vi bruge en lower bound på E.

Hvis en node v har en grad på mindre end k, så ville cuttet ($\{v\}$, $V - \{v\}$) have en størrelse mindre end k hvilket modsiger at (A, B) er global min-cut, hvilket betyder at alle nodes i G har en grad på mindst k. Så $|E| \geq \frac{1}{2}kn$, Så chancen for at en edge i F bliver contracted er højest:

$$\frac{k}{\frac{1}{2}kn} = \frac{2}{n}$$

En upper bound for sandsynligheden for at en edge i F bliver contracted efter j iterationer:

$$\frac{k}{\frac{1}{2}k(n-j)} = \frac{2}{n+j}$$

So vi vil gerne finde lower bound på at i løbet af ingen af iterationerne at en edge i F er blevet contracted. E beskriver det event at en edge i F ikke blev contracted:

$$\begin{aligned} & \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdots \Pr[\mathcal{E}_{j+1} | \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_j] \cdots \Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_{n-3}] \\ & \geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-j}\right) \cdots \left(1 - \frac{2}{3}\right) \\ & = \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \cdots \left(\frac{2}{4}\right) \left(\frac{1}{3}\right) \\ & = \frac{2}{n(n-1)} = \binom{n}{2}^{-1}. \blacksquare \end{aligned}$$

Så chancen for at algoritmen fejler efter en gang er tæt på 1. Vi kan køre algoritmen C(n, 2) gange for at en upper bound på at fejle på 1/e

Coupon Collector

Vi har n coupons.

1 coupon per pakke og uniformly random kort fra hver pakke.

Hvor mange pakker skal vi købe før vi har alle kort?

Lad X være random variable som beskriver antallet af pakker vi skal købe for at få alle coupons.

Lad X_j være antallet af pakker der skal til for at få den j 'ne nye coupon.

Vi kan beskrive X som summen af alle random variables for alle j 'er:

$$X = X_0 + X_1 + X_2 + \dots + X_{n-1}$$

Så vi skal finde X_j for hvert j .

Chancen for at vi får en ny coupon efter j coupons er:

$$p_j = \frac{n-j}{n},$$

Så expected value for random variable ved j er:

$$E[X_j] = \frac{1}{p_j} = \frac{n}{n-j}$$

For at finde expected value for X , kan vi bruge linearity of expectation:

$$E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{j=0}^{n-1} \frac{1}{n-j} = n \sum_{i=1}^n \frac{1}{i} = nH(n)$$

Randomized Approximation for MAX 3-SAT

Et set af clauses som har tre værdier hver. Værdierne er i mængde:

$X = \{x_1, x_2, x_3, \dots, x_n\}$. Skal vi finde ud af om der eksistere en satisfying truth assignment? Altså en assignment af alle værdier i X for at alle clauses bliver true.

Hvis det ikke kan findes, laver vi problemet om til optimering. Hvilken assignment giver flest true's.

Simple algorithm:

Vi assigner hvert enkelt x i X til enten 1 eller 0, med $p = 1/2$. Lad Z være random variable som beskriver hvor mange satisfyed clauses vi har. Z kan splittes op til mindre dele:

$$Z = Z_1 + Z_2 + \dots + Z_k, \text{ hvor } Z_i = 1 \text{ hvis } C_i \text{ er satisfyed.}$$

Her kan vi udregne the expected value for hvert enkelt Z_i . Der er tre værdier i den Clause og for at den ikke skal være satisfyed skal de tre værdier assignes tre specifikke værdier. Og da de er independent af hinanden er chancen for at en clause ikke er satisfyed $(\frac{1}{2})^3$

Derfor er clause C_i satisfyed med chance $1 - \frac{1}{8} = \frac{7}{8}$, hvilket betyder:

$$E[Z_i] = p(C_i) = \frac{7}{8}$$

Ved hjælp af linearity of expectation, kan vi få en udtryk for alle k clauses:

$$E[Z] = E[Z_1] + E[Z_2] + E[Z_3] + \dots + E[Z_k] = k \frac{7}{8}$$

(13.14) Consider a 3-SAT formula, where each clause has three different variables. The expected number of clauses satisfied by a random assignment is within an approximation factor $\frac{7}{8}$ of optimal.

(13.15) For every instance of 3-SAT, there is a truth assignment that satisfies at least a $\frac{7}{8}$ fraction of all clauses.

Der må eksistere en assignment som satfyer mindst den expected mængde af clauses.

Finding the Median

Vi vil gerne finde medianen af et array, uden at skulle sortere arrayet og tage det midterste element. Hvis vi vælger medianen som splitter så får vi en køretid på $O(n)$, men det er lidt cirkulært da det er den vi gerne vil finde til at starte med.

Hvis vi altid vælger det mindste element, så vil vores subarray kun falde med 1 hver gang og så ville køretiden være $\Theta(n^2)$

Vi vil gerne vælge en random splitter så subarrayet, vi rekursivt kalder, bliver mindre med en konstant mængde, hvilket vil give os lineær køretid.

Vi vil gerne sørge for at den splitter ligger centralt, altså ca. en kvart af elementerne er på den ene side, så mindst en fjerde af elementerne bliver smidt væk hvert rekursivt kald.

Chancen for at vi vælger en central splitter er derfor $\frac{1}{2}$. Vi kan derfor konkludere at expected antal gange vi køre hvert stadie er 2.

X er random variable på hvor mange steps vi skal bruge i alt.

$X = X_1 + X_2 + X_3 + \dots$, hvor X_j er det antal steps der skal bruges i stadie j af algoritmen

Så når algoritmen er i stadie j , så er der højest $n(\frac{3}{4})^j$ elementer i arrayet. Så antallet af steps i en iteration i stadie j er højest $cn(\frac{3}{4})^j$, for en konstant c . Vi ved at expected antal iterationer i stadie j er højst 2, så vi ved:

$$E[X_j] \leq 2cn(\frac{3}{4})^j$$

Ved at bruge linearity of expectation kan vi bound running time for alle stadier:

$$E[X] = \sum_j E[X_j] \leq \sum_j 2cn(\frac{3}{4})^j = 2cn \sum_j (\frac{3}{4})^j \leq 8cn$$

Den konvergerer og derfor ved vi:

(13.18) *The expected running time of Select(n, k) is $O(n)$.*

Quicksort

Her har vi samme problem som før, hvis vi altid vælger det mindste element som splitter vil vi få en køretid på $\Theta(n^2)$, så vi vil, som før, gerne vælge en splitter som ligger centralt, altså hvor, når vi opdeler i subarrays, så har hver subarray mindst en fjerdedel af elementerne i.

Hvis vi heldigvis valgte medianen hver rekursive kald ville vi få en upper bound køretid $O(n \log n)$.

Her kigger vi på expected running time, og vi vil gerne vise at den kan blive bound med $O(n \log n)$.

Hver eneste sub-problem j i algoritmen har højst $n(\frac{3}{4})^j$ elementer, men flere en $n(\frac{3}{4})^{j+1}$ elementer. Tiden det tager at splitte et array S op i to og tjekke om det er en central splitter er $O(|S|)$, uden rekursive kald, så for vores subproblem j ville det være $O(n(\frac{3}{4})^j)$ uden de rekursive kald.

Antallet af subproblems vi kan opdele j i er $(\frac{4}{3})^{j+1}$, så ved at bruge linearity of expectation for alle subproblemer j så ved vi at køretiden for j er upper boundet $O(n)$. Antallet af forskellige typer a subproblemer er bounded ved $\log_{\frac{4}{3}}n = O(\log n)$, hvilket giver os vores ønsket bound:

(13.21) *The expected running time of Modified Quicksort is $O(n \log n)$.*

Chernoff Bounds

Vi vil gerne finde ud af om en random variable er i nærheden af expected value.

Så hvis vi har en random variable X , som er summen af en masse independent trials, som kan være enten 0 eller 1.

$$X = X_0 + X_1 + X_2 + \dots, \text{ hvor } p(X_i = 1) = p_i$$

By linearity of expectations så ved vi at:

$$E[X] = \sum_{i=1}^n p_i$$

Fordi de er uafhængige ville det intuitivt give mening at de flukturerer og vil "cancel" hinanden ud og derfor komme med tæt på expectation med høj sandsynlighed. Her kigger vi på to scenarier, at X afviger over $E[X]$ eller X afviger under $E[X]$. Dette kalder vi Chernoff Bounds.

(13.42) Let X, X_1, X_2, \dots, X_n be defined as above, and assume that $\mu \geq E[X]$. Then, for any $\delta > 0$, we have

$$\Pr [X > (1 + \delta)\mu] < \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu.$$

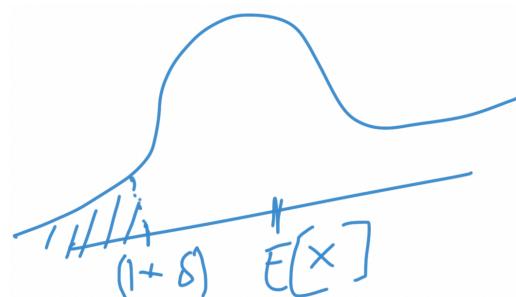
En anden form af chernoff bounds:

(13.43) Let X, X_1, X_2, \dots, X_n and μ be defined as above. Then for any $1 > \delta > 0$, we have

$$\Pr [X < (1 - \delta)\mu] < e^{-\frac{1}{2}\mu\delta^2}.$$

Så vi kigger på chancen for at vores random variable afviger fra gennemsnittet med en eller anden værdi.

Vi kan vende uligheden i venstre side for at få upper eller lower bound



Randomized Algorithms - CLRS

The Hiring Problem

Der bliver sendt en ny kandidat ind hver dag, og hvis kandidaten er bedre end den nuværende assistent, så bliver assistenten erstattet. Dette koster dog hiring fee.

c_i er interview cost, og c_h er hiring cost. Interview cost er lavt mens hiring cost er høj.

Hvis m er antallet af personer hyret i løbet af algoritmen, så ville vi forvente en cost på:

$O(n \cdot c_i + m \cdot c_h)$, som upper bound. Det er hvis vi har interviewet alle og hyret de bedste hver gang. mc_h er det eneste der ændre sig fra run til run.

Worst case ville være hvis vi altid hyrede den næste som kom ind. Altså kandidaterne var sorteret i stigende rækkefølge.

Så ville hiring cost ville ende ud i: $O(nc_h)$

Hvad forventer vi for costen, da vi aldrig kan regne med at de kommer i rækkefølge?

Vi bruger probabilistic analysis til at finde køretiden af algoritmer, eller i dette tilfælde, cost af at køre algoritmen.

Hvis vi rankgere hver kandidat, så får vi en liste af ranks, som er en permuteret liste af vores originale liste af kandidater. Så når vi siger at vores kandidater bliver interviewet tilfældigt så siger vi også at hver eneste permutation af ranklisten er mulig, altså $n!$ forskellige permutationer.

Vi bruger average-case running time når inputs er random, men expected running time, når algoritmen selv bruger randomness. Så hvis vi får en liste af kandidaterne inden, og får algoritmen til at vælge en randomt, kan vi snakke om expected running time. Det gør det til en **Randomized Algoritme**

For at analysere randomized algorithms bruger vi indicator random variables.

Hiring problem med randomized input:

Lad X være random variable som beskriver hvor mange gange vi skal hyre en ny assistent.

$$\text{Lad } X_i = \begin{cases} 1 & \text{hvis } i \text{ er hyret} \\ 0 & \text{o.w.} \end{cases}$$

Så: $X = X_1 + X_2 + \dots + X_n$

$$E[X_i] = p(X_i = 1)$$

Når vi når den i'ende kandidat, så er der lige stor chance for at nogle af de første i personer er den mest kvalificeret. Derfor har kandidat i chance $1/i$ for at være mest kvalificeret.

$$p(X_i = 1) = \frac{1}{i}, \text{ og derfor:}$$

$$E[X_i] = p(X_i = 1) = \frac{1}{i}$$

Så for at finde $E[X]$, bruger vi linearity of expectation:

$$\begin{aligned} E[X] &= E \left[\sum_{i=1}^n X_i \right] \\ &= \sum_{i=1}^n E[X_i] \\ &= \sum_{i=1}^n \frac{1}{i} \\ &= \ln(n) + O(1) \end{aligned}$$

Pga:

Harmonic series

For positive integers n , the n th **harmonic number** is

$$\begin{aligned} H_n &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} \\ &= \sum_{k=1}^n \frac{1}{k} \\ &= \ln n + O(1). \end{aligned}$$

Det betyder at vi egentlig kun forventer at hyre $\ln(n)$ mennesker ud af n når vi kører algoritmen. Her har vi fundet average-case running time, da det ikke er en randomized algorithm, men vi assumer randomized input

Randomized Algorithms

Hvis vi ikke kender til distributionen af inputs. Så i stedet for at regne med at vores input er tilfældigt, så kan algoritmen tilfældigt permuttere inputtet for at sikre et random input.

Du garanterer nærmest at inputtet ikke er dårligt ordered. Selv hvis inputtet var ordered worst-case, ville en randomized algoritme højest sandsynligt ikke køre langsommere af det.

Her finder vi expected running time, da vi er garanteret et randomized input.

Randomly Permuting Arrays

Permute by sorting og randomized in place

Permute by sorting laver en random rankgeling af elementerne og sorterer dem efter rankgelingen.

Randomized Quicksort

Hvis vi har X til at være antallet af comparisons i Quicksort, og n er antallet af elementer vi sorterer, så vil upper bound køretiden være $O(n + X)$.

For at bevise det:

Man laver to set af tal. Et set hvor tal er i ascending order, et array.

Så definerer vi settet Z_{ij} som sættet med alle elementer mellem i og j .

Her prøver vi at finde ud af hvornår algoritmen compare i og j .

X_{ij} er indicator random variable, og er 1 hvis i og j blev compared.

Så kan vi definere X som summen af alle indicator random variables:

$$X = \sum_{i=0}^{n-1} \sum_{j=i+1}^n X_{ij}$$

Og med linearity of expectation:

$$\begin{aligned} E[X] &= \sum_{i=0}^{n-1} \sum_{j=i+1}^n E[X_{ij}] \\ &= \sum_{i=0}^{n-1} \sum_{j=i+1}^n p(X_{ij} = 1) \end{aligned}$$

Så skal vi kun finde $p(X_{ij} = 1)$.

$$\begin{aligned} \Pr\{z_i \text{ is compared to } z_j\} &= \Pr\{z_i \text{ or } z_j \text{ is first pivot chosen from } Z_{ij}\} \\ &= \Pr\{z_i \text{ is first pivot chosen from } Z_{ij}\} \\ &\quad + \Pr\{z_j \text{ is first pivot chosen from } Z_{ij}\} \\ &= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\ &= \frac{2}{j-i+1}. \end{aligned}$$

Så kan vi udregne expected running time:

$$\begin{aligned}
 E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} \\
 &= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
 &< \sum_{i=1}^{n-1} \sum_{k=1}^n \frac{2}{k} \\
 &= \sum_{i=1}^{n-1} O(\lg n) \\
 &= O(n \lg n).
 \end{aligned}$$

Universal Hashing

Hvis vi har en hash funktion, som brugeren kender, kan vi risikerer at få et input, som har en masse collision. Det vi gør for at undgå at inputtet bestemmer hvor god vores hash funktion er, er at vi følger en hash funktion tilfældigt fra en familie af hash funktioner.

U er univers af keys, som: $U = \{0, 1, 2, 3, 4, \dots, m - 1\}$

H er family of hash functions.

Det vi gerne vil opnå med universal hash funktions, for to distinct keys:

$$p(h(k_1) = h(k_2)) \leq \frac{1}{m}$$

Eller vi ønsker at antallet af hash funktioner som gør at to keys mapper til det samme er højest $|H| / m$

Theorem 11.3

Suppose that a hash function h is chosen randomly from a universal collection of hash functions and has been used to hash n keys into a table T of size m , using chaining to resolve collisions. If key k is not in the table, then the expected length $E[n_{h(k)}]$ of the list that key k hashes to is at most the load factor $\alpha = n/m$. If key k is in the table, then the expected length $E[n_{h(k)}]$ of the list containing key k is at most $1 + \alpha$.

Som siger at hvis key, k , ikke er i table, så er expected antal af elementer i en table entry højest $\alpha = n/m$, og hvis k er i table, er expected antal af elementer i table entry højest $1 + \alpha$

Proof:

For hver eneste par af distinct keys, k_i og k_j , laver vi en indicator random variable:

$$X_{ij} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{o.w.} \end{cases}$$

Så definerer vi en random variable som beskriver alle de keys andre en k_i der mapper til samme entry som k_i :

$$Y_i = \sum_{k_j \in T, k_j \neq k_i} X_{ij}$$

Vi ved at $p(X_{ij} = 1) \leq \frac{1}{m}$, pga definitionen af universal collection of hash function i bogen.

Derfor:

$$\begin{aligned} E[Y_i] &= E \left[\sum_{k_j \in T, k_j \neq k_i} X_{ij} \right] \\ &= \sum_{k_j \in T, k_j \neq k_i} E[X_{ij}] && \text{By linearity of expectation} \\ &\leq \sum_{k_j \in T, k_j \neq k_i} \frac{1}{m} && \text{By definition of universal collection of hash functions} \end{aligned}$$

Her kigger vi på om key, k_i , allerede er i T .

Hvis den ikke er: Så er Y_i antallet af elementer på entry i i table. Hvilket vil sige at alle andre keys som er i T ikke er k_i .

$$\text{Derfor: } E[Y_i] \leq \frac{n}{m} = \alpha$$

Hvis k allerede er i T: Her bliver vi nødt at til plusse en, da k_i ikke hører med i Y_i . Derfor er antallet af elementer i entry i: $Y_i + 1$. Vi ved også at alle andre keys i table har størrelsen $|n - 1|$. Derfor:

$$E[Y_i] \leq \frac{n-1}{m} + 1 = 1 + \alpha - \frac{1}{m} < \alpha + 1$$

Dette betyder at vi ikke kan give et input som forværre køretiden. Kun tilfældighed bestemmer om vi ender med worst-case

Køretiden for n operationer af enten INSERT, DELETE eller SEARCH. DELETE og SEARCH tager konstant tid, og siden der er n operationer og m table entries, Så ved vi at der er $O(m)$ insertions, og n er $O(m)$, hvilket betyder at $\alpha = O(1)$. Derfor vil hele sekvensen af operationer blive bounded under $O(n)$

Designing universal class of hash functions

Når man skal vælge en god hashing family eller class, skal man først vælge et primtal p.

Det kræves at hele universet af keys skal dækkes ind af den række af tal fra 0 til p-1, altså: $0 \leq k < p$, og siden vi går ud fra at der er flere keys end hash table entries så holder det at: $p > m$.

Vi definere to ranges: $\mathbb{Z}_p = \{0,1,2,3,\dots,p - 1\}$, og $\mathbb{Z}'_p = \{1,2,3,\dots,p - 1\}$

Nu definerer vi hash funktionen h_{ab} , som går fra hvilken som helst $a \in \mathbb{Z}'_p$ til hvilken som helst $b \in \mathbb{Z}_p$:

$$h_{ab} = ((ak + b) \text{ mod } p) \text{ mod } m$$

Så for at definere en klasse eller familie af hash funktioner ud fra dette:

$$H_{pm} = \{h_{ab} : a \in \mathbb{Z}'_p \text{ og } b \in \mathbb{Z}_p\}$$

Hver eneste hash funktion i H_{pm} mapper fra \mathbb{Z}_p til et entry i tabellen med m entries. Uanset hvor stor m er, og siden vi har $p-1$ valg for a og p valg for b, betyder det at vi har $p(p - 1)$ hash funktioner i H_{pm} .

Network Flows

Flow netværk defineres som et set af vertices og et set af edges $G = (V, E)$ og er en directed graf. Hver eneste edge har en non-negative capacity, Ingen self-loops eller parallel edges.

Flow i et netværk følger disse to rules:

$$0 \leq f(u, v) \leq c(u, v), \text{ for hvilke som helst } u, v \in V \quad \text{- Capacity Constraint}$$

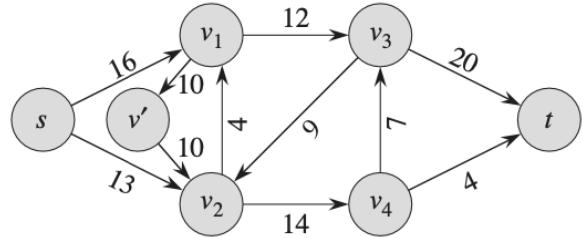
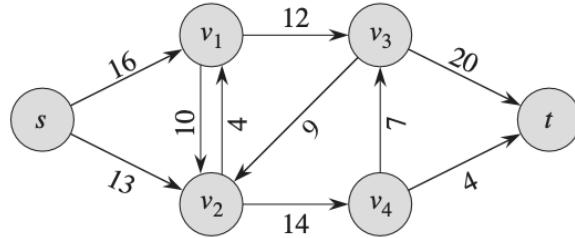
$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u), \text{ for alle } u \in V - \{s, t\} \quad \text{- flow conservation}$$

For at beskrive total flow for et givet flow i et flow netværk bruger vi denne formel.

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

Selvom, lige nu, er det sidste led altid 0, da der ikke går noget ind i sourcen, putter vi den stadig på, da vi skal bruge det senere når vi snakker om at finde maximum flow i et netværk.

Avoiding Antiparallel:



For networks med flere sources og sinks, laver vi en supersource og en supersink som har en edge til alle respektive sources og sinks, som har en capacity på ∞ .

Ford-Fulkerson

En måde at finde maksimum flow i et givet netværk.

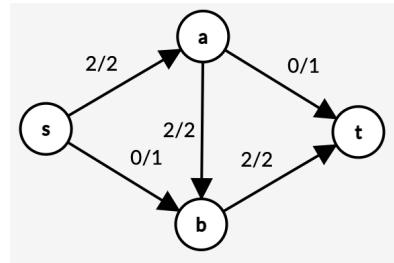
Risidual Networks

Residual network bliver defineret som G_f .

Hvis $f(u, v) > 0$, så lav en edge (v, u) hvor $c_f(v, u) = f(u, v)$

Hvis $f(u, v) < c(u, v)$ lav en edge (u, v) hvor $c_f(u, v) = c(u, v) - f(u, v)$

Man laver en risidual graf ud fra et flow i en graf, G.



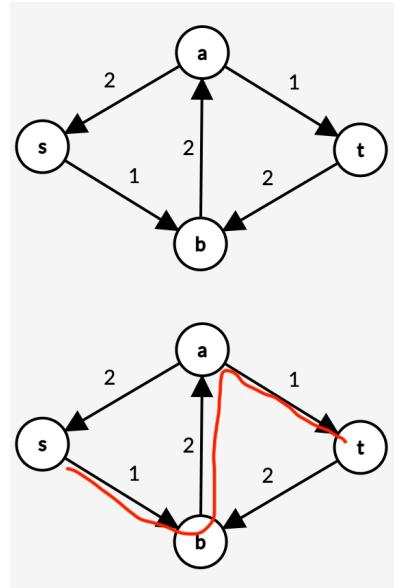
Augmenting Path

Augmenting path er path fra s til t i G_f .

Hvis ingen augmenting path eksisterer i G_f , så har vi allerede fundet max flow.

Vi finder max capacity af vores path ved at finde det laveste capacity af de edges som udgør augmenting path.

Så skal vi increase flowet med min capacity over augmenting path, i G. Da der ikke eksistere en edge (b, a) i G, så ved vi at vi skal trække min capacity fra flowet i (a, b) i G.



Cuts

Et cut (S, \bar{S}) opdeler settet V op i to sub-sets. Hvor $S \subset V$ og $\bar{S} \subset V - S$.

Det er et s-t cut hvis $s \in S, t \in \bar{S}$

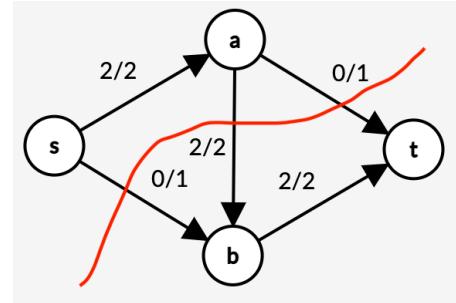
Capaciteten af et cut defineres som $c(S, \bar{S}) = \sum_{v \in S, u \in \bar{S}} c(v, u)$

Min-cut er defineret som det mindst mulige kapacitet blandt alle s-t cuts i en graf.

Eksempel:

$$c(S, \bar{S}) = \{(s, b), (a, b), (a, t)\} = 4$$

Det er altså ikke min-cut vi har fundet



LEMMA: Hvis f er et flow i G , og (S, \bar{S}) er et s-t cut i G . Så gælder det at $|f| \leq c(S, \bar{S})$

Proof:

$$\begin{aligned} |f| &= \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ into } S} f(e) \\ &\leq \sum_{e \text{ out of } S} f(e) \\ &\leq \sum_{e \text{ out of } S} c(e) = c(S, \bar{S}) \quad \text{Capacity rule. } f(e) \leq c(e) \end{aligned}$$

Max flow min cut theorem:

if $|f| = c(S, \bar{S})$ så er f max-flow og (S, \bar{S}) er min-cut.

Proof:

$a : f$ is max flow

$b : G_f$ har ingen augmenting paths

$c : |f| = c(S, \bar{S})$ for et s-t cut.

$a \Rightarrow b \Rightarrow c \Rightarrow a$

$a \Rightarrow b$: Contradiction. Hvis f er et maximum flow i G , og der eksisterer en augmenting path i G_f , så modsiger det vores tidligere statement om at hvis en augmenting path eksisterer, kan vi increase flowet og derfor få et nyt flow som er større end f .

$b \Rightarrow c$: Der er ingen augmenting paths i G_f . Definer et set af vertices: $S = \{v \in V \mid$ der er en path fra s til v i $G_f\}$. Og $T = V - S$, altså resten af vertices. Fordi der ingen paths er fra s til t , så holder det at: $s \in S, t \notin S$.

u og v er i hvert sit sub-set. Hvis $(u, v) \in E$, så må det holde at $f(u, v) = c(u, v)$ ellers ville der eksisterer en path fra s til v i G_f . Samtidigt, hvis der eksisterer $(v, u) \in E$, så skal $f(v, u) = 0$ da der ellers ville eksistere en edge fra u til v i G_f .

Hvis der ikke eksisterer nogen edge mellem u og v . Så ville vi ikke have noget flow fra u til v og derfor ville $c(u, v) = c(S, \bar{S})$.

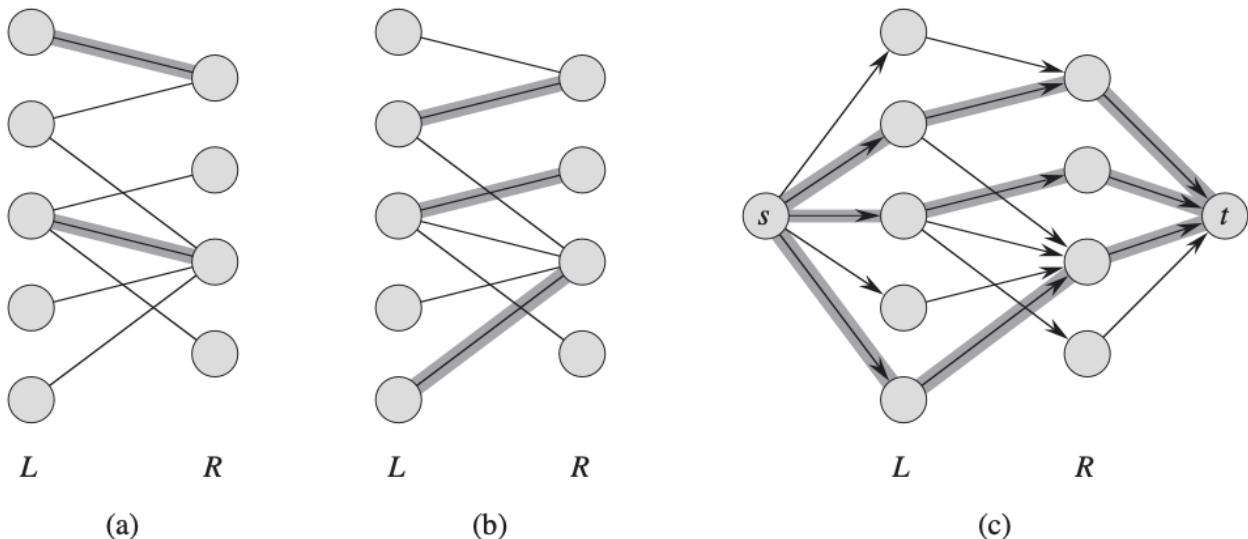
$c \Rightarrow a$: Vi ved at $|f| \leq c(S, \bar{S})$ for alle s-t cut. Derfor så implicerer det at hvis $|f| = c(S, \bar{S})$ så er f maks flow. Det kan aldrig blive større.

Edmonds-Karp

Improvement af Ford-Fulkerson, da den udfører breadth-first-search på augmenting paths, så den altid tager den korteste augmenting path først.

Ford-Fulkerson kan komme til at vælge en dårlig augmenting path, og kan endda ikke terminer hvis en dårlig path er valgt.

Maximum bipartite matching



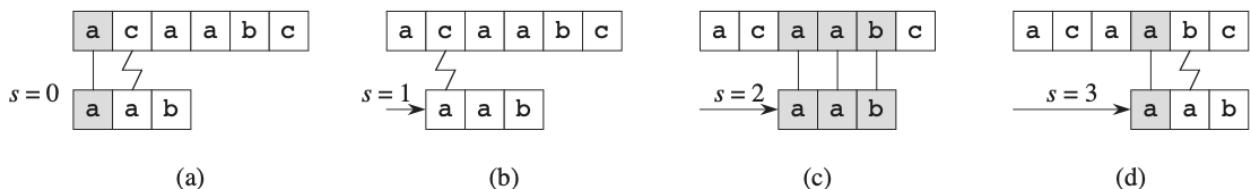
Vi ønsker at finde det bedste mulige matching af elementer på venstre side og elementer på højre. Vi laver det om til en digraph og adder en s og t . Vi sætter alle edges til at have en kapacitet på 1.

Find maximum flow i grafen for at finde det maximum antal matches mellem de to sider.

String Matching

Naive String Matching

Algoritmen går gennem teksten vi gerne vil matche på og tjekker om symbolerne matcher. Så shifter den en gang mod højre og prøver igen.



Antallet positioner den skal compare, er $n-m+1$ hvis man tæler den første position med.

Derfor er worst-case running time $O((n - m + 1)m)$ og jo dårligere more input er, jo mere tight vil den bounde running time. Udover det, så kan vi risikere at det pattern vi vil matche er cirka halvdelen af hele længden teksten, hvilket vil maksimere antallet af comparisons og shifts. Hvis vi havde flere shifts ville vi have færre comparisons og omvendt. Hvis vi derudover havde et input på a^n , altså n a'er i række. Og vi prøver at matche a^m , og $m = \lfloor \frac{n}{2} \rfloor$, så ville køretiden være $\Theta(n^2)$.

Rabin-Karp Algorithm

Basic idea

$$\begin{array}{l} a=1 \\ b=2 \\ c=3 \\ d=4 \end{array}$$

Tekst: aaaaab

Input: aab

$$h(\text{input}) = 1 + 1 + 2$$

$$P_1 = h(\text{aaa}) = 1 + 1 + 1 = 3$$

$$P_2 = \underline{\quad} + \underline{\quad} + \underline{\quad} = (P_1 - 1) + 1 = 3$$

$$P_4 = h(\text{aab}) = 1 + 2 + 1 = 4 \Rightarrow \text{Match - compare}$$

Drawback

tekst: ccacccaaedba

input: dba

$$\begin{array}{l} a:1 \\ b:2 \\ c:3 \\ d:4 \\ e:5 \end{array}$$

$$h(\text{input}) = 4 + 2 + 1 = 7$$

$h(P_1) = 3 + 3 + 1 = 7$ hash match - ikke ens \Rightarrow spurious hits

$$h(P_1) = h(P_2) = h(P_3)$$

Worst-case $\Theta(mn)$, $\Theta(n-m+1)$ hvis ingen spurious hits

actual Rabia korrekt

$$h(P) = P[1] \cdot 10^{m-1} + P[2] \cdot 10^{m-2} + P[3] \cdot 10^{m-3} \Rightarrow \sum_{i=1}^m P[i] \cdot 10^{m-i} \text{ 10 er } (\sum)$$

Tekst ~~ccacccaa~~

$$S_0 = 3 \cdot 10^0 + 3 \cdot 10^1 + 1 \cdot 10^2$$

$$M=3 S_1 = (S_0 - 3 \cdot 10^2) + 10^3 + 3 \cdot 10^0$$

Worst-case: $\Theta(mn)$

Average: $\Theta(n-m+1)$

Modulus

$h(P) \% 2^{32}$ hvis unsigned int \rightarrow højere risiko for spurious hits