Algoritmid ja andmestruktuurid

Ülevaades sellest kuhu oleme jõudnud

Ja kuidas edasi minna

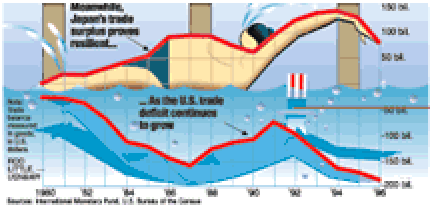
Mis on algoritm?

Probleem

↓

Algoritm

↓



Andmed → Programm → Tulemus

Algoritmide loomise paradigmasid

- jaga ja valitse
  - lahendame väiksemateks ülesanneteks jagamisega
- ahne algoritm
  - lokaalne valikukriteerium annab globaalse tulemuse
  - taandame ülesande ühele väiksemale alamülesandele
- dünaamiline planeerimine
  - paneme alamülesannete lahendustest kokku suurema ülesande
- tagasivõtmisega algoritm, hargne ja kärbi
  - otsime lahendust puust ja kärbime seda niipalju kui suudame
- metaheuristikad
  - võimaldavad leida häid tulemusi ilma headuse garantiita
- lähendavad algoritmid
- paralleelsed algoritmid

Andmestruktuurid

- annavad andmete töötlemiseks algoritmile abstraktse liidese
  - võimaldab kirjutada algoritmi kõrgemal, andmestruktuuri mõistete tasandil
    lisa element (puusse, kuhja), ühenda kaks alamhulka
  - võimaldavad kasutada erinevaid realisatsiooni
    - optimiseeritud erinevate operatsioonide tarvis
    - erinevad valmis realisatsiooni (paketid, teegid)
- peidavad/realiseerivad algoritmi olulise keerukuse
  - enamus programmi mahust võib olla andmestruktuuride realisatsioon
- realiseeritakse lihtsamate andmestruktuuride baasil

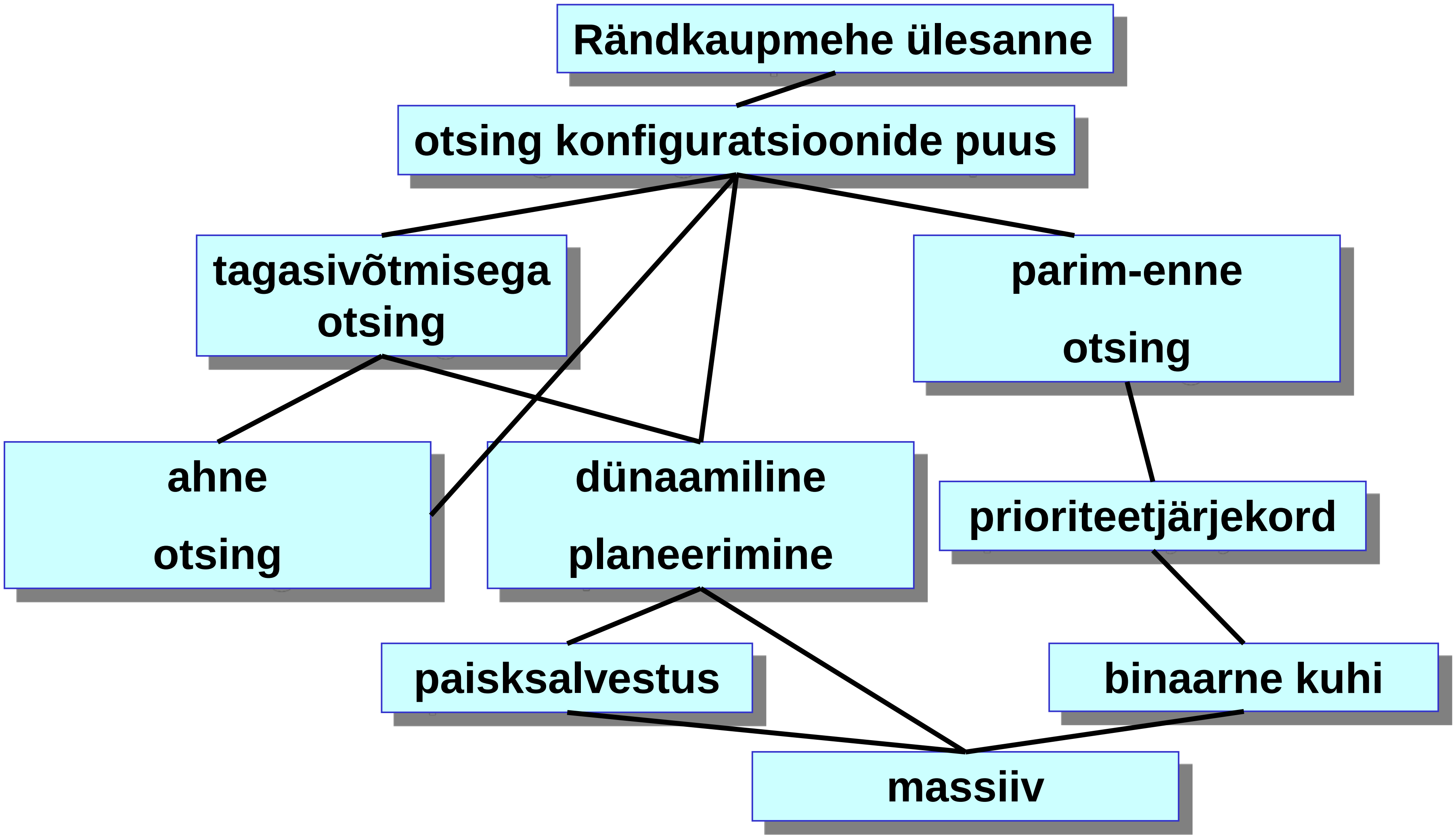
Kompositsiooniline lähenemine

- Algoritmi tehakse abstraktsema andmestruktuuri terminites
  - algoritm on arusaadavam
  - kasutatakse efektiivseid andmestruktuuride realisatsioone
- Algoritmis kasutatakse teadaolevaid häid algoritme
  - minimaalse katva puu leidmine lähendavas TSP-s
  - sügavuti otsing graafil topoloogiliseks sorteerimiseks
- Andmestruktuuride realiseerimiseks kasutatakse lihtsamaid andmestruktuure

  prioriteetjärjekord - kuhi - binaarpuu - massiiv

  mittelõikuvad alamhulgad - tagurpidid lingitud puu - massiiv

Mõtte areng optimiseerimisprobleemis



Mis kasu on algoritmidest ja andmestruktuuridest?

- Üldised algoritmide klassid
  - võimaldavad mõista probleemi olemust
  - võimaldavad probleemile ise algoritmi leida
- Andmestruktuurid
  - võimaldavad mõelda abstraktsemalt
  - kirjutada koodi kõrgema tasemel
  - muuta käitumist andmestruktuuri realisatsiooni ja tüübi muutmisega (sügavuti - laiuti - parim enne otsing)
  - lihtsustada algoritmi kasutades sobilikku olemasolevat andmestruktuuri implementatsiooni

Algoritmide keerukus

- Algoritmi keerukus on põhioperatsiooni(de) arvu sõltuvusfunktsioon K(*n*) sisendi(te) suurusest *n*.
  - *Põhioperatsioon* ei ole üheselt defineeritav
    - Üldiselt midagi mis on riistvara tasandil tehtav piiratud arvu sammudega
  - *Sisendi suurus* võib olla defineeritud erinevalt
    - Sisendandmete maht (massiivi, faili, andmebaasi suurus)
    - Sisendparameetri väärtus
    - Sisendparameetri suurus (bittide/baitide arv)
- Tavaliselt hinnatakse halvima juhu (sisendi) keerukust

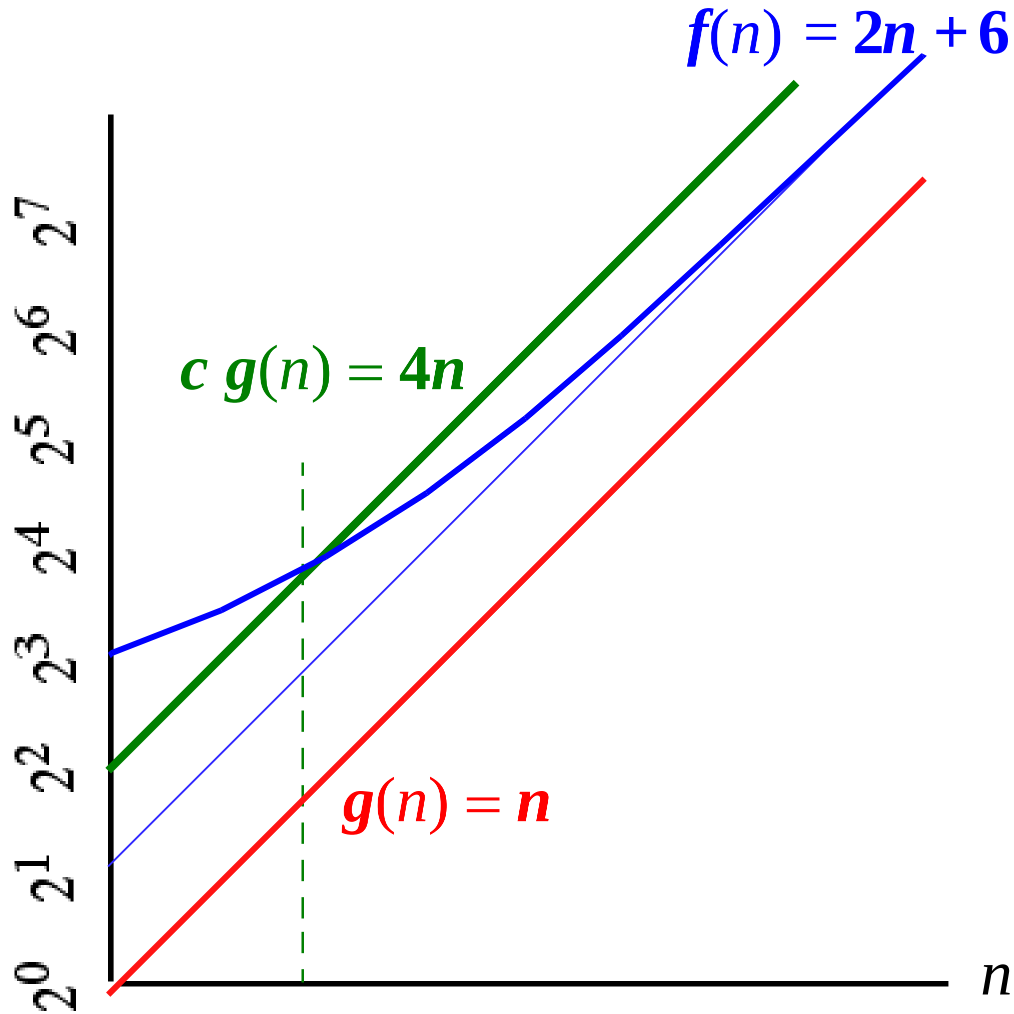
Asümptootiline keerukus

Funktsioonide ***f(n)*** ja ***g(n)*** jaoks on olemas konstandid ***c*** ja *N*, nii et:

$f(n) \leq c\, g(n)$, kui $n \geq N$

järeldus:

2n+6 is O(n).



Lahtine probleem

- Arvutuslik keerukus
  - Käsitleb probleemi
  - Annab keerukuse alumise raja - sellest piirist väiksema keerukusega algoritmi olla ei saa
- Algoritmi keerukus
  - Analüüsib algoritmi
  - Annab keerukuse ülemise raja - probleemi keerukus ei ole suurem kui seda lahendava algoritmi keerukus
- Probleeme, kus keerukuse alumine ja ülemine raja ei lange kokku nimetatakse lahtisteks probleemideks

Lahtine probleem

$O(n!)$	$O(n!)$ algoritm P lahendamiseks	ülemine raja
$O(K^n)$	$O(K^n)$ algoritm P lahendamiseks	
?	Probleemi P olemuslik keerukus	
$O(n^{12})$	Tõestus, et P vajab vähemalt $O(n^{12})$ keerukust	alumine raja
$O(n^3)$	Tõestus, et P vajab vähemalt $O(n^3)$ keerukust	

NP täielik probleem

- Tähtis ülesannete klass
  - sisaldab palju praktikas olulisi probleeme
  - raske leida lahendust - eksponentsiaalse keerukusega
  - lihtne lahendust kontrollida - polünomiaalse keerukusega
  - probleemid on omavahel seotud - kui ühe jaoks leitakse polünomiaalne lahendusalgoritm, siis on see olemas kõigi jaoks
- Mida teha?
  - defineerida probleem ümber, nii et leiduks vähem-keerukas algoritm (tihti piisab mingist lisakitsendusest)
  - kasutada heuristikaid, mis võimaldavad paljudel praktilistel juhtudel ülesande ära lahendada
  - kasutada lähendavat algoritmi

Kuidas saab veel arvutada

- Juhuslikkuse kasutamine – viskame münti
- Meta-heuristiline lähenemine – matkime loodust
- Paralleelne arvutamine – teeme seda kambaga
- Kvantarvutamine, molekularvutamine – looduse saladuste ärakasutamine, massiivne paralleelsus

Juhuslikkuse kasutamine

On mitmed üldiseid meetodeid probleemide ligikaudseks lahendamiseks või kiirendamiseks, mis tuginevad juhuslikkusele

- deterministlike algoritmide juhuslikustamine
- Monte Carlo meetod, simuleerimine
- metaheuristikate kasutamine NP täielike ülesannete lahendamiseks

Algoritmide juhuslikustamine

- Kasutatakse juhul kui
  - algoritmi keskmine ja halvima juhu keerukus on erinevad
  - halvima juhu realiseerumine on keskmisest tõenäolisem
- Algoritmi tuuakse sisse juhuslikkuse moment, nii et halvima juhtumi realiseerumine poleks sõltuv sisendandmetest vaid juhuslikkusest
- Iga algoritmi täitmine samade sisendandmete korral võib võtta erineva aja

Juhuslikustatud Quicksort

- Valime teljeks juhusliku elemendi
  - esimese elemendi teljeks valimisel realiseerub halvima juhtumi keerukus, kui andmed on juba sorteeritud
- Sorteerime kõik teljest väiksemad temast vasakule ja suuremad paremale
- Sorteerime teljest paremale ja vasakule jääva massiivi

Algarvulisuse test

- Kas N on algarv?
- Eksponentsiaalse keerukusega N suuruse (bittide arus suhtes)
- Miller-Rabin test on võimalik kontrollida testide arvu väljendava sisendparameetri *s* alusel
  - kas arv on kordarv
  - kas arv on algarv tõenäosusega 2^{-s}, ehk
    iga testi tsükliga väheneb algarvuks olemise tõenäosus 2 korda
    s = 50 piisab igaks praktiliseks rakenduseks

Monte Carlo meetod

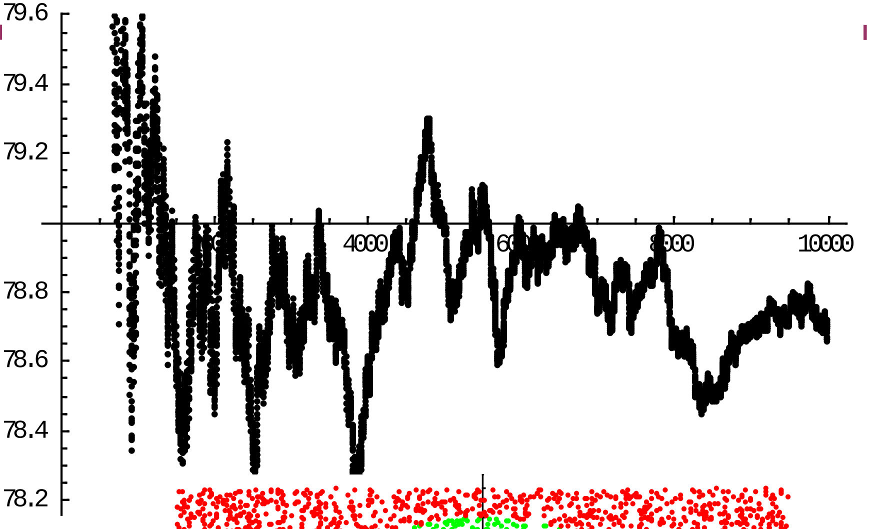
- Stohastiline simuleerimismeetod paljude matemaatiliselt keeruliste probleemide lahenduse numbrilise hinnangu saamiseks
  - kasutab (pseudo)juhuslikke arve probleemi paljukordseks läbisimuleerimiseks

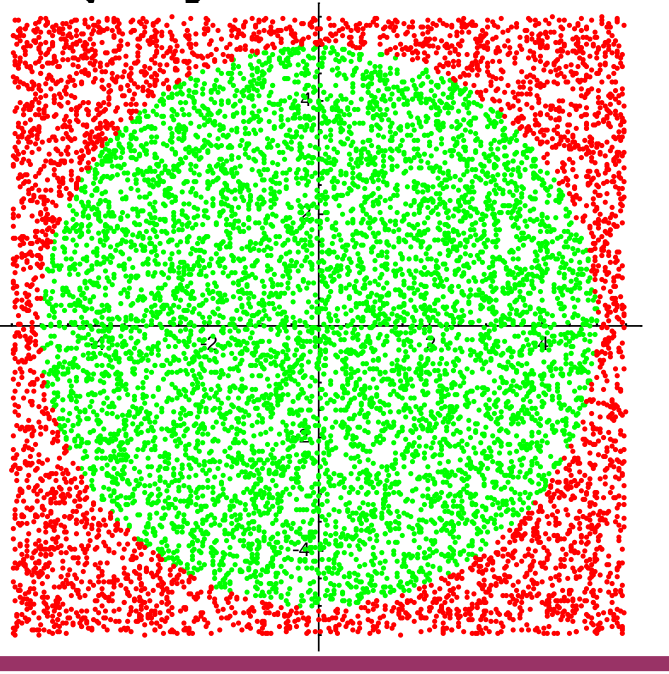
Näiteid

- http://www.plu.edu/~heathdj/java/stats/MonteCar.html
- Selgitatakse objekti sisendite statistiline jaotus
  - Genereeritakse juhuslikke sisendeid vastavalt leitud jaotusele
  - Loendatakse tulemusi
  - Arvutatakse hälve, et hinnata tulemuse täpsust

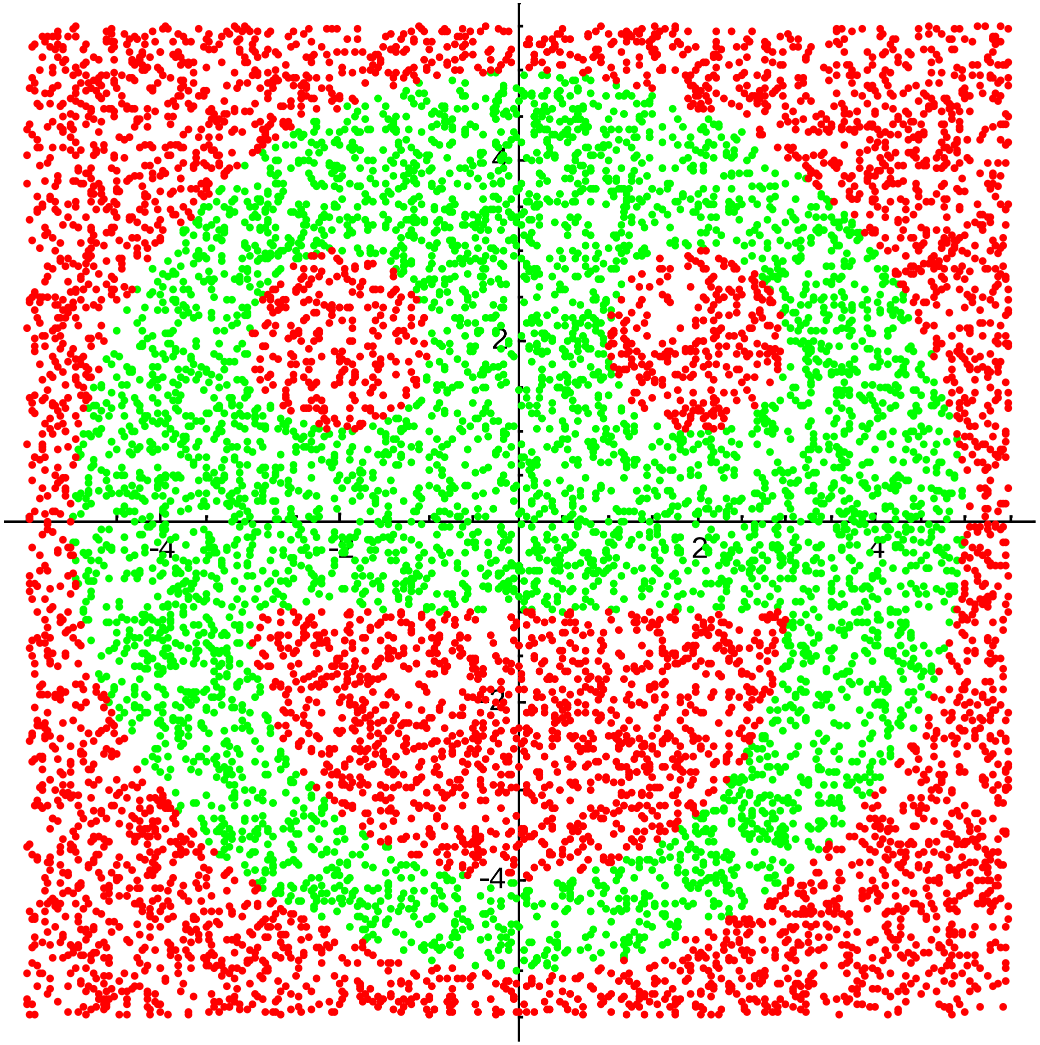
Example: the area of a circle

- Sample points randomly from square surrounding circle of radius 5
- 10,000 sample points
- Acceptance criterion: inside circle
- Actual area: 78.54
- Calculated area: 78.66





Example: more complicated shapes



Monte Carlo kasutamine kärpimise efektiivsuse hindamiseks

- Annab hinnangu kärbitud otsingupuu suurusele
- Iga Monte Carlo simulatsioon on annab puu suuruse hinnangu liikudes mööda üht haru nii sügavale kui võimalik.
- Piisava hulga simulatsioonide keskmisena saame hinnang kogu puu suurusele.
- On rakendatav kui
  - kõigil tasemetel kasutatakse sama *promising* funktsiooni
  - sama taseme sõlmedel on ühepalju järglasi

m_i lootustandvate järglaste arv tasemel i

t_i järglaste arv tasemel i

$$1 + t_0 + m_0\, t_1 + m_0\, m_1\, t_2 + \dots + m_0 \dots m_{i-1}\, t_i$$

Metaheuristikad

- On mitmeid üldisi metaheuristikaid
  - geneetilised algoritmid (*genetic algorithms*)
  - simuleeritud lõõmutamine (*simulated annealing*)
  - tabudega otsing (*tabu search*)
  - optimiseerimine sipelgakolooniaga (*ant colony optimization*)
- Aitavad leida optimaalsele lahendusele lähedast lahendust olukorras, kus optimaalse lahenduse leidmine pole arvutusmahu tõttu realistlik
- Ei anna hinnangut tulemuse “headusele”,
  - tulemuse osas pole kindlust kui palju see erineb optimaalsest
  - optimaalse tulemuse leidmisel, ei peatu

Metaheuristikad

Koosnevad tavaliselt kahest osast

- lokaalse optimumi poole liikumine
- hajutamine, et mitte takerduda lokaalsesse optimumi

Nende vahel tuleb leida tasakaal, et mitte:

- koonduda viletsasse lokaalsesse optimumi
- taanduda juhuslikule otsingule

Geneetilise algoritmi idee

- Genereeritaks lahenduseks vormiliselt sobilik (juhuslik) isendite (kromosoomide) hulk, ehk põlvkond.
- Kontrollitakse iga isendi sobivust lahenduseks ja antakse neile sobilikkuse hinnang
- Ühe põlvkonna isendeid kasutatakse järgneva põlvkonna loomiseks, püüdes seda teha nii, et need oleks keskmiselt paremad
- Jätkatakse kuni arvatakse olevat leitud lahend või piisavalt hea lahend.

Geneetilise algoritmi üldkuju

1. Genereeri juhuslik *n* isendiga põlvkond (*n* genoomi)
2. Hinda iga isendi sobilikkust *s*(*n*)
3. Loo uus põlvkond kasutades järgnevaid samme
   a) vali kaks vanemat - suurema tõenäosusega parema sobilikkusega isendid
   b) ristamistõenäosusega rista järglaste tekitamiseks vanemate kromosoomid, vastasel juhul kopeeri
   c) muteerimistõenäosusega muuda mõne geeni väärtust
   d) lisa tulemus uude generatsiooni
4. Kasuta järgnevalt uut põlvkonda
5. Vastavalt lõpetamistingimusele jätka punktist 2 või lõpeta

Järglaste loomine

- Kromosoomi kodeerimine

  Näiteks on kromosoomiks (probleemi argumendiks) on üks täht ja number 0-7.

  Kodeerime tähe 5 bitiga, sümboli 3 bitiga:

  `A5 = 00001101`

  `F2 = 00110010`

- Ristamine

  lõikame gromosoomid juhuslikust kohast pooleks ja vahetame vanemate kromosoomide tagumised osad

  `vanemad:  00001101 (A5) 00110010 (F2)`

  `järglane: 00101101 (E5)`

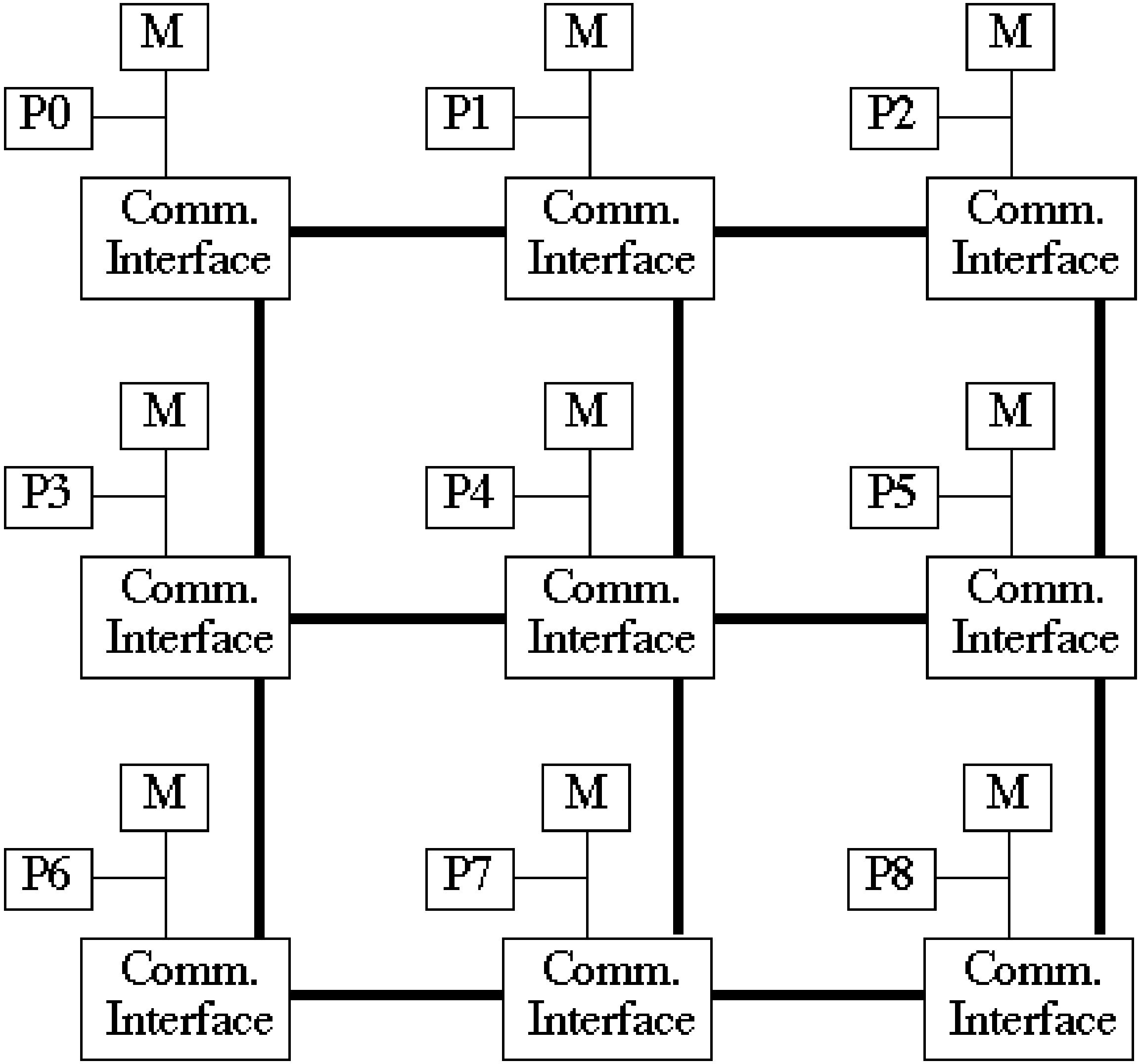
- Muteerimine

  `00101101 (E5)    00101111 (E7)`

Metaheuristilised algoritmid

- Annavad tihti enamvähem hea lahenduse keerulisele probleemile
- Ei anna kunagi kindlust, et leitud lahendus on optimaalne optimaalsusülesande lahendamisel
- NP täieliku ülesande lahendamisel saadud “jah” vastuse korral on meil lahend käes
- Võimaldavad mitte süveneda probleemi matemaatilisse keerukusse
- Pole imerohuks - efektiivsus sõltub oluliselt kasutatavast kodeeringust ja operaatoritest

Parallel computer



PRAM
Parallel Random-Access Machine

- Shared Memory – SMT (Simultaneous Multithreading)
  - Each processor may have local memory to store local results
- MIMD – Multiple Instruction Multiple Data
- SIMD – Single Instruction Multiple Data
  - Model is MIMD, but algorithms tend to be SIMD
  - Each active processor executes same instruction
- SIMT - Single Instruction Multiple Thread (CUDA)
- Synchronous
- Memory Access
  - There are different PRAM models depending on the possibility of the concurrent read/write

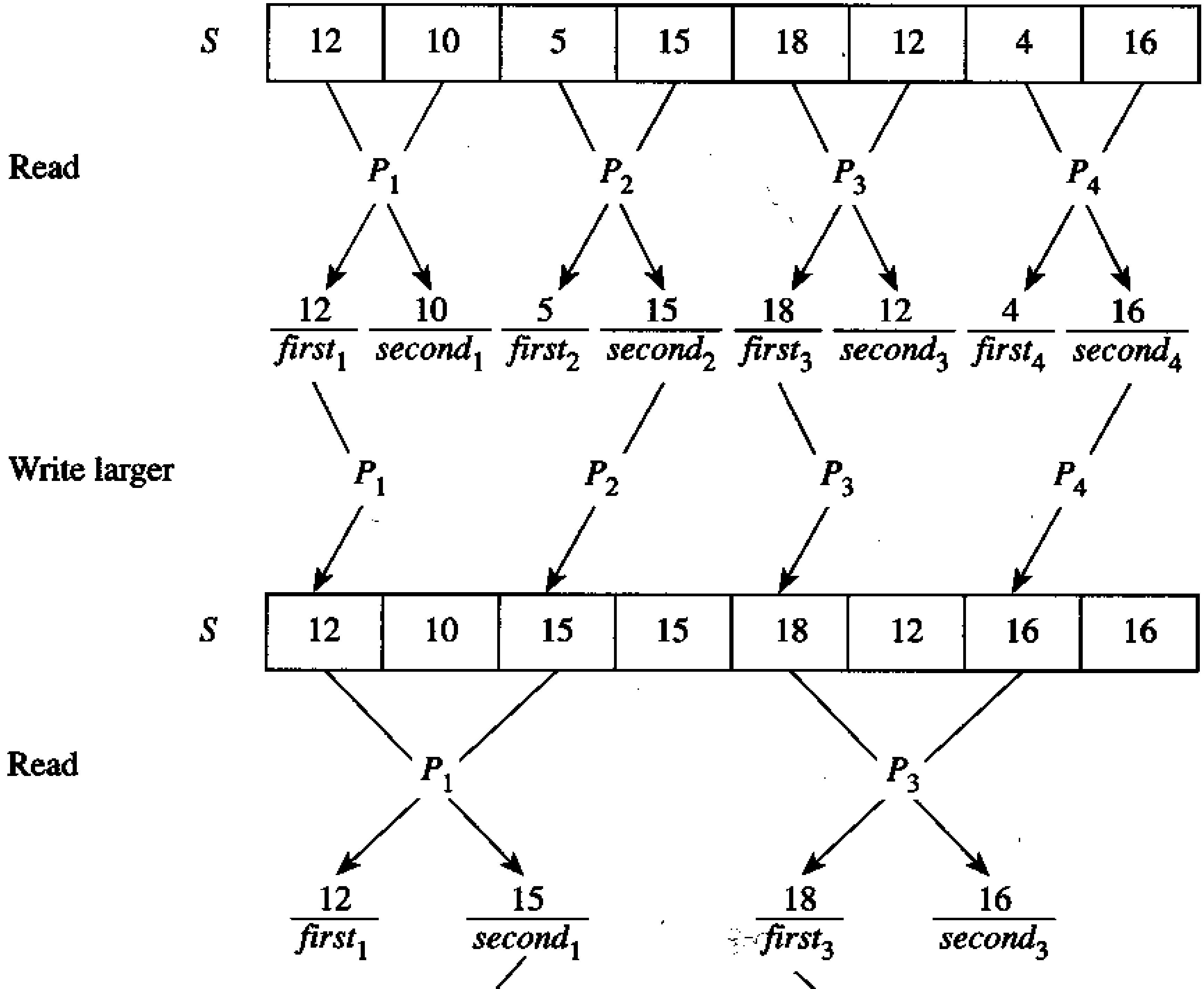
Memory Access

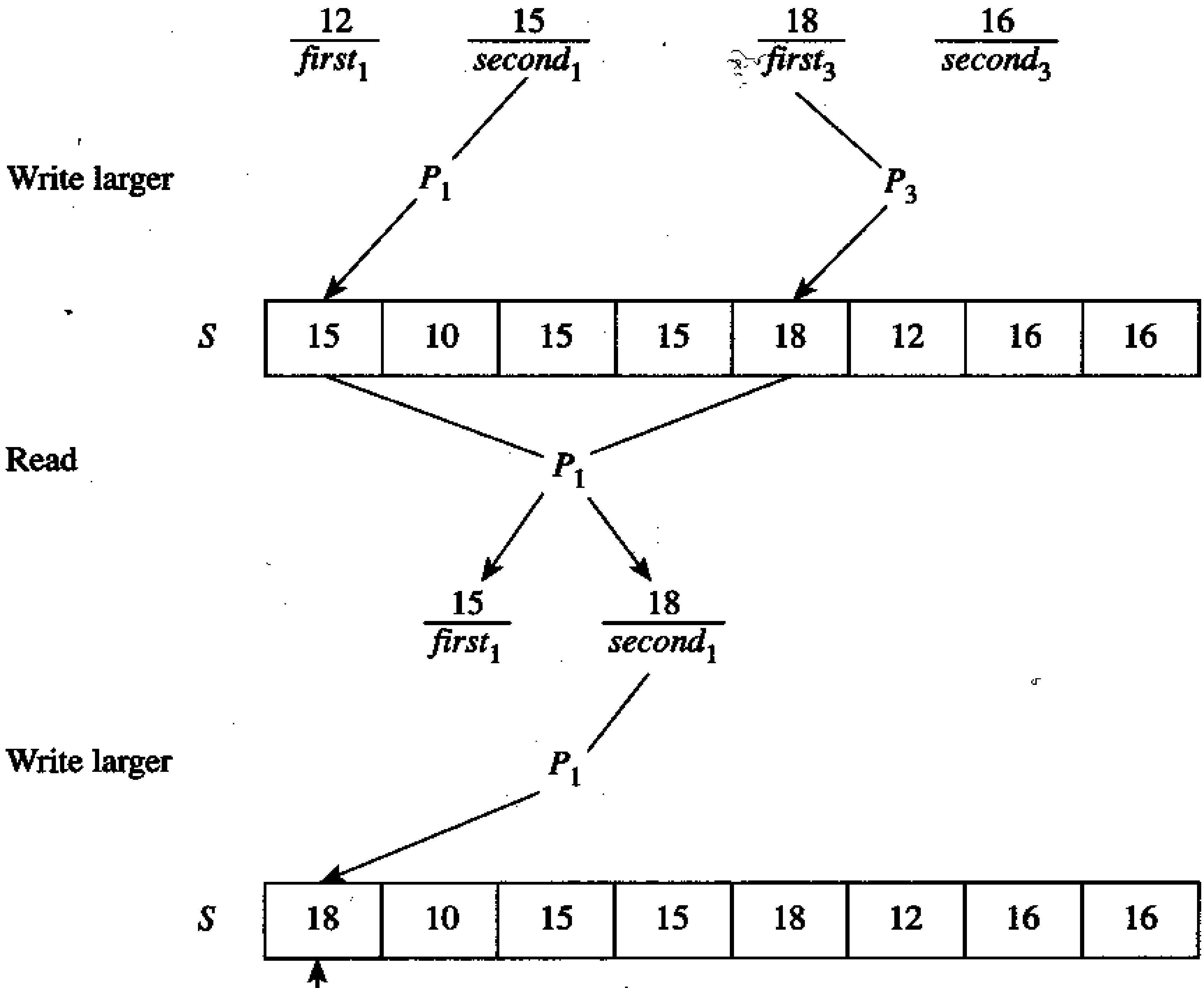
- Exclusive Read Exclusive Write (EREW)
  - no simultaneous access to a single shared-memory location
- Concurrent Read Exclusive Write (CREW)
  - simultaneous reads of a single shared-memory location are allowed
- Concurrent Read Concurrent Write (CRCW)
  - simultaneous reads and writes are allowed on a single shared-memory location are allowed

Conventions of the parallel algorithms

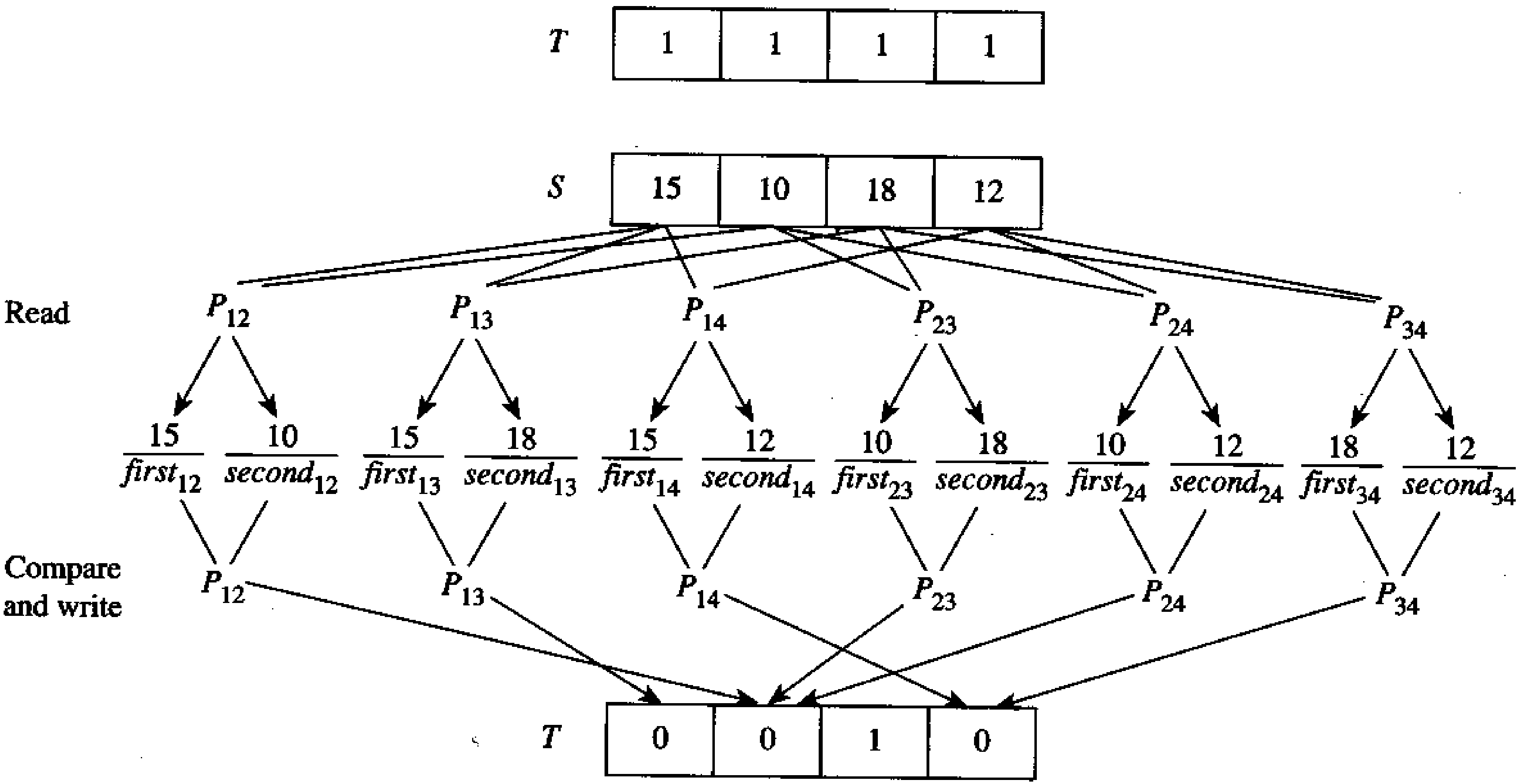
- Each processor can know it's own index

  p = index of this processor;
- A variable can be in
  - shared memory - accessible to all processors
  - private memory - each processor has a local copy of the variable
- Shared memory is accessed only for reading and writing (not for operations)
- Processors do the steps, reads and writes simultaneously
- There is unlimited number of procs available


S
12 10 5 15 18 12 4 16
Read
P1 P2 P3 P4
12 first1
10 second1
5 first2
15 second2
18 first3
12 second3
4 first4
16 second4
Write larger
P1 P2 P3 P4
S
12 10 15 15 18 12 16 16
Read
P1 P3
12 first1
15 second1
18 first3
16 second3


12
first1
15
second1
18
first3
16
second3
Write larger
P1
P3
S
15 10 15 15 18 12 16 16
Read
P1
15
first1
18
second1
Write larger
P1
S
18 10 15 15 18 12 16 16

CRCW PRAM algorithm for finding the largest key



Complexity of the parallel (finding) algorithms

Algorithm	*Time complexity*	*Size (Processors)*
Sequential	O(n)	1
CREW PRAM	O(lg n)	n/2
CRCW PRAM	O(1)	n(n-1)/2

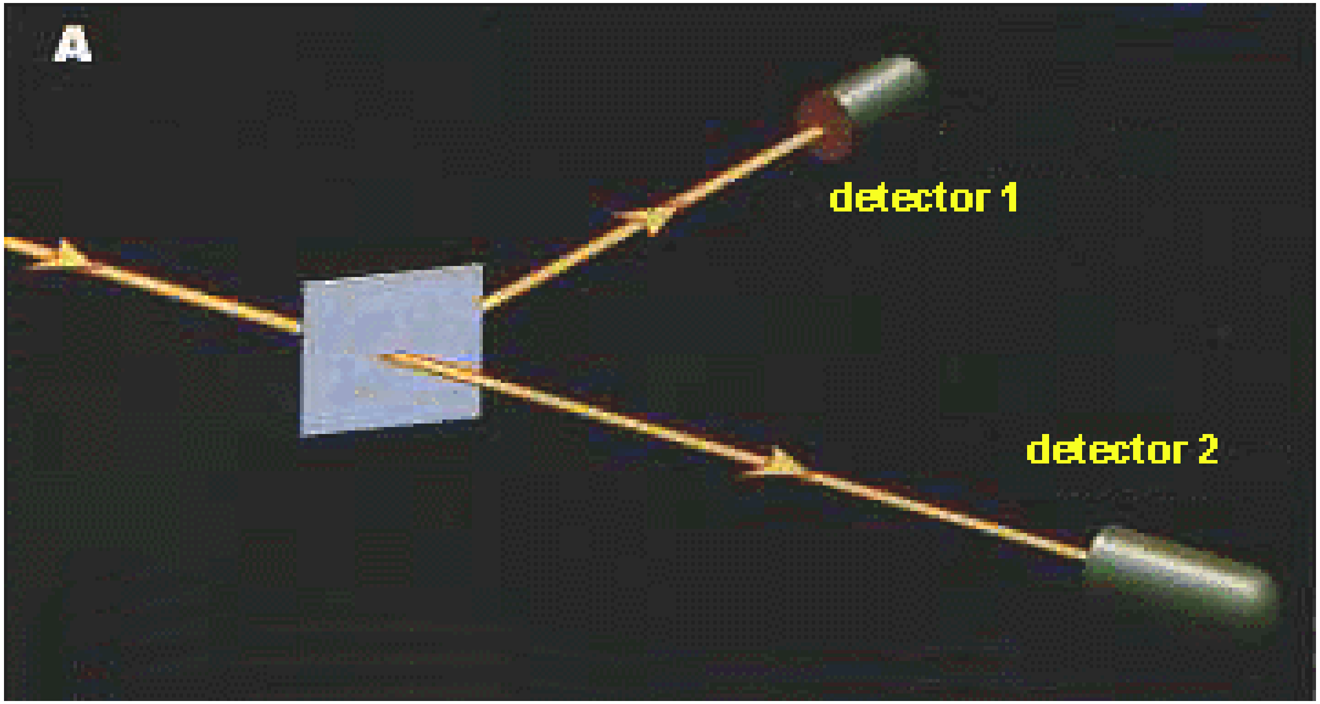
- Order of magnitude improvement can be achieved only when the number of processors is unlimited.
- Product complexity *Time* $\times$ *Size* cannot be better than the complexity of the best sequential algorithm.
- Sequential space complexity ~ parallel time complexity
  - Sequential-PSPACE = Parallel-PTIME

Kvantarvutamine

- Tugineb kvantmehaanika nähtustel
  - Kvantolek on tõenäosuslik
  - Kvantoleku mõõtmisel olek muutub
  - Põimolekus kvantnähtused toimuvad sünkroonselt ja hetkeliselt sõltumata vahemaast

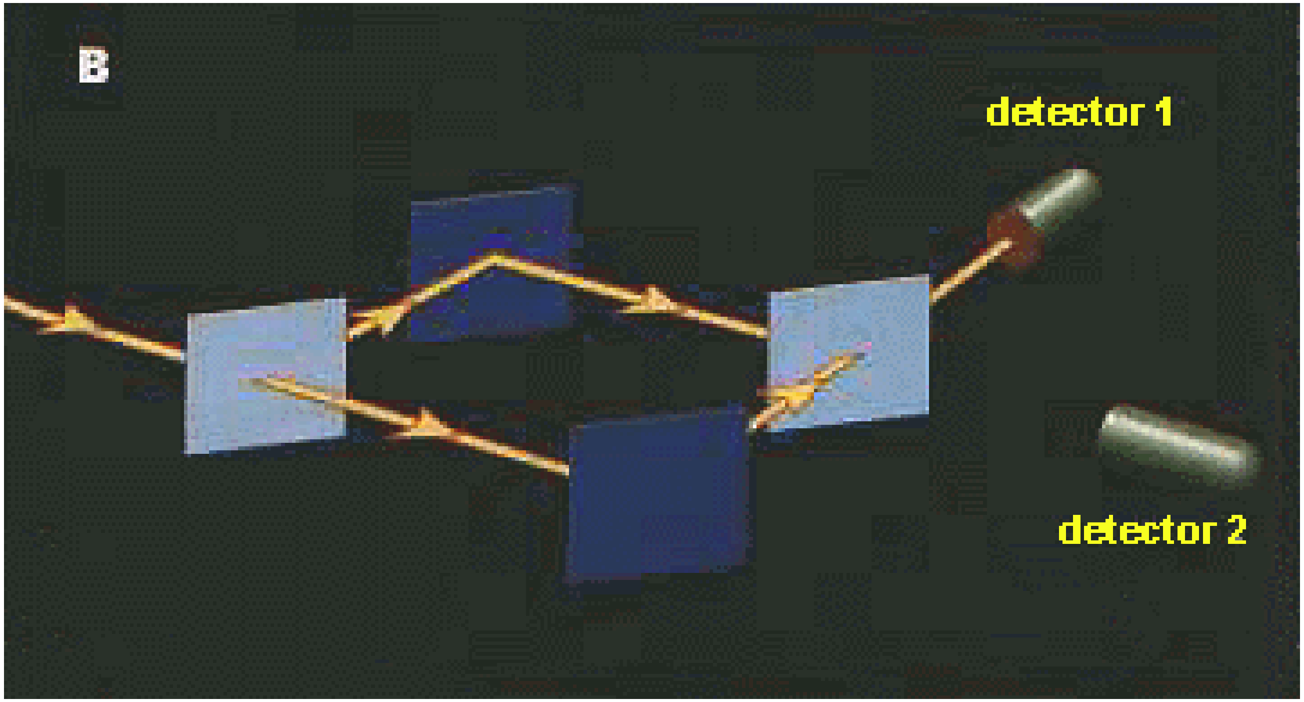
Lihtne kvanteffekt

- Laseme 1 footoni läbi poolläbipaistva (50%) peegli
- Mõlemad andurid registreerivad footoni 50% tõenäosusega



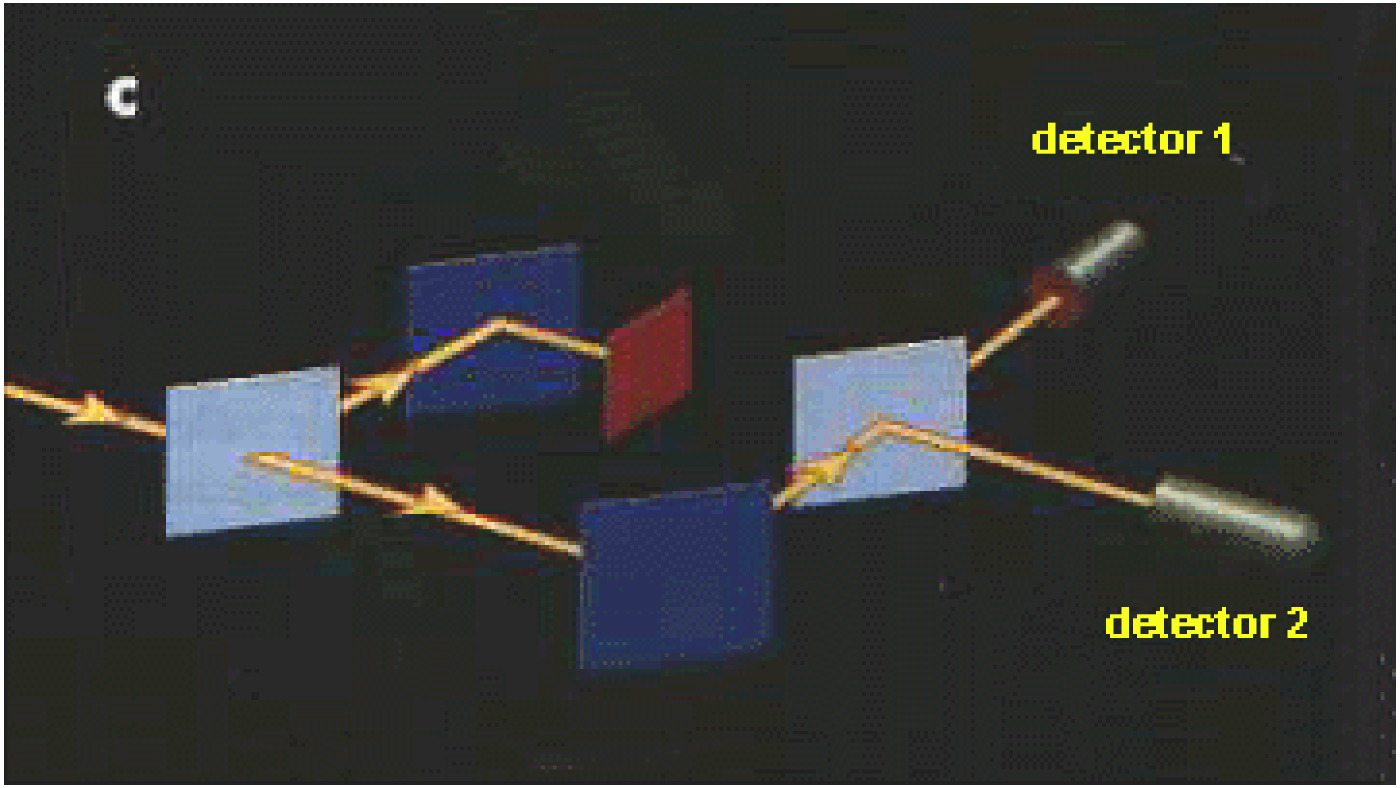
Lihtne kvanteffekt

- Lisame teise pooläbipaistva peegli ja ühendame süsteemid peeglite abil
- Andur 1 registreerib footoni 100% ja andur 2 0% tõenäosusega ?!?



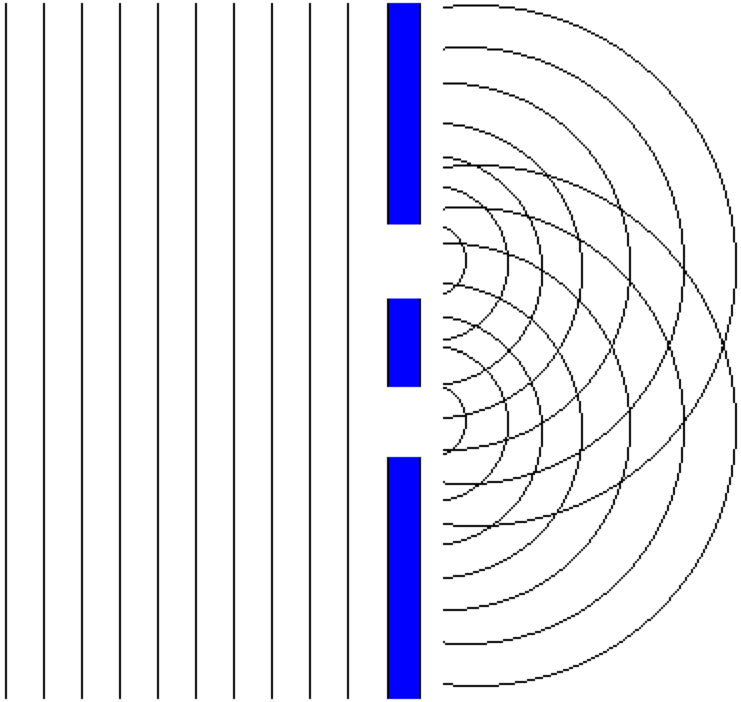
Lihtne kvanteffekt

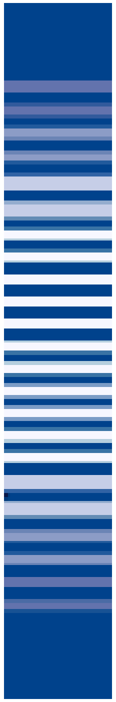
- Katkestades ükskõik kumma tee registreerivad mõlemad andurid footoni 50% tõenäosusega

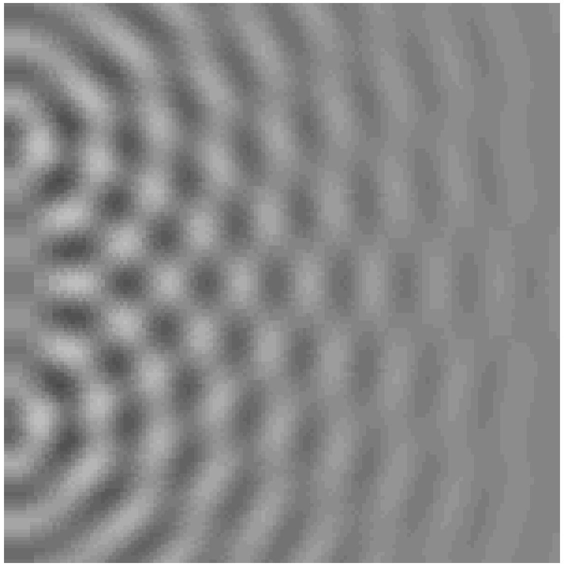


Interferents

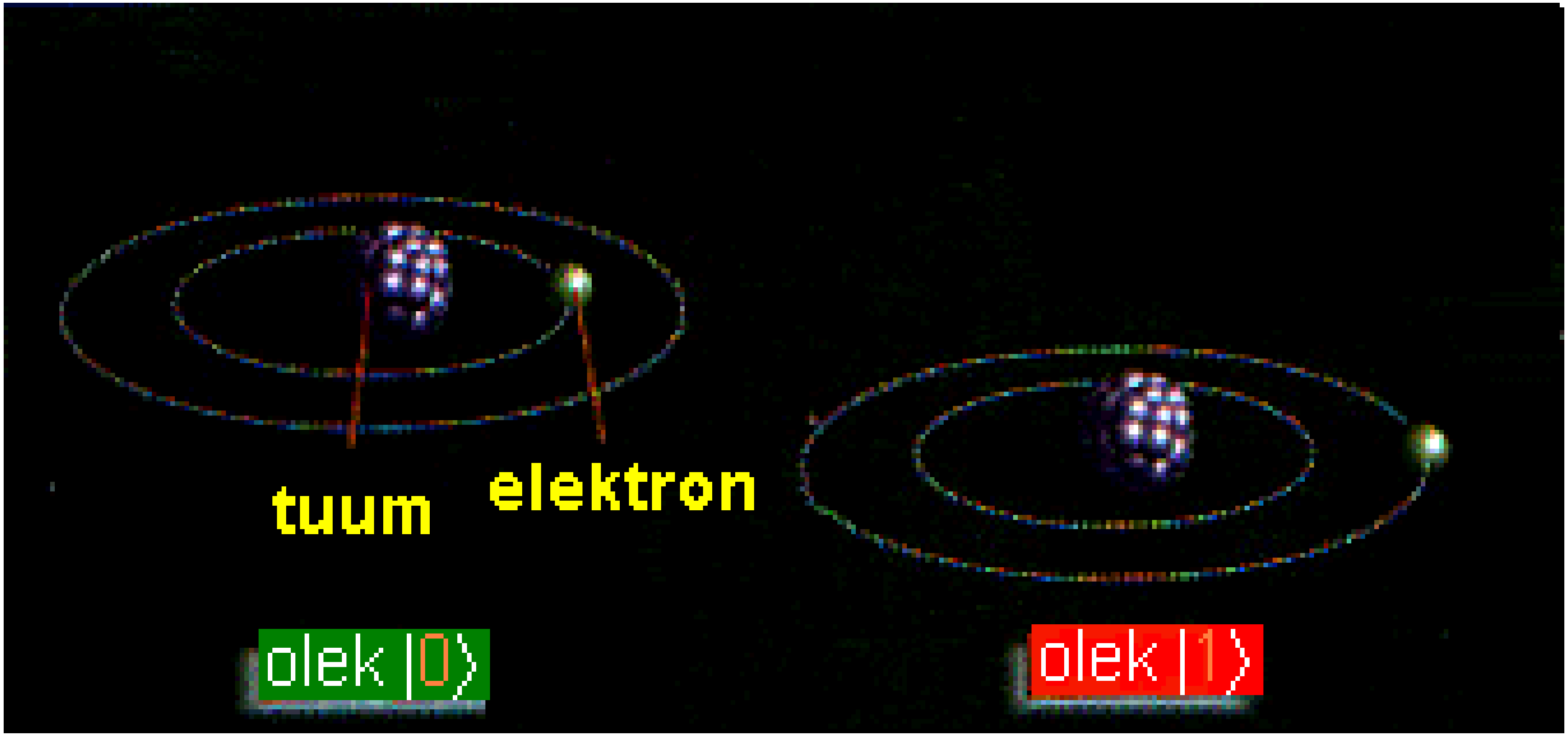
- Tegemist on interferentsiga







Kvantolekud



Kvantbitt

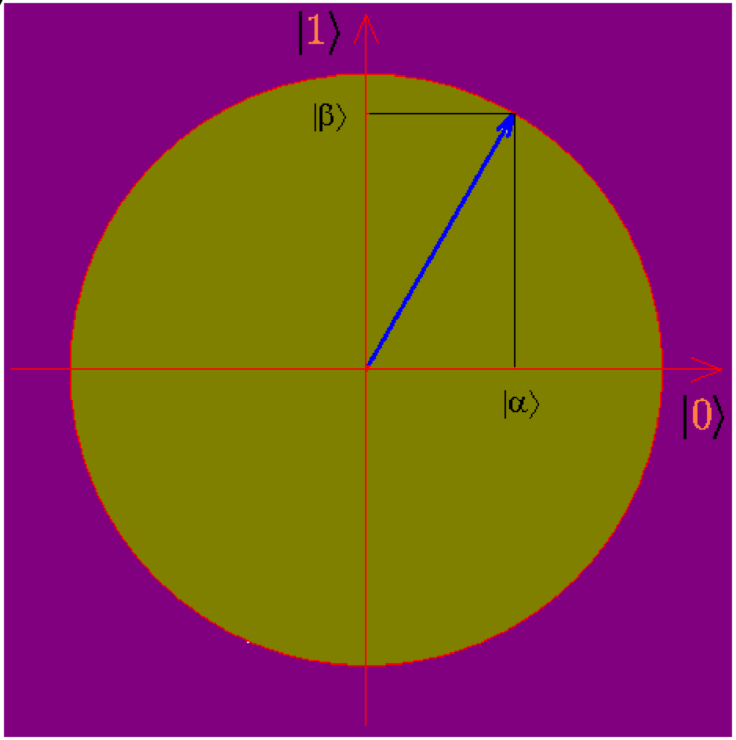
- Matemaatiliselt: $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$
- Siin on $|0\rangle$ ja $|1\rangle$ baasivektorid kompleksruumis ning α ja β kompleksarvud.
- Füüsikaliselt võivad $|0\rangle$ ja $|1\rangle$ olla näiteks aatomi põhi- ja ergastatud seisund, footoni polarisatsioonitasand, spinni suund jm.
- Kvantbitt
  - on korraga “mingi tõenäosusega” mõlemas olekus
  - kannab korraga kahte kompleksväärtust

Kvantbiti mõõtmine

- Kvantbiti $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ mõõtmisel baasi $|0\rangle$, $|1\rangle$ suhtes saame

  tõenäosusega $|\alpha|^2$ tulemuse $|0\rangle$

  tõenäosusega $|\beta|^2$ tulemuse $|1\rangle$
- Normeerimistingimus
  $|\alpha|^2+|\beta|^2=1.$
- Kvantbitt $|\varphi\rangle$ hävib mõõtmisel.
  - Kui mõõtmise tulemus on 0, siis on kõigi järgnevate mõõtmiste tulemus ka 0



√not kvantfunktsioon

$\sqrt{\text{not}}\,(0) = i/\sqrt{2}|0\rangle + 1/\sqrt{2}|1\rangle$

$\sqrt{\text{not}}\,(1) = 1/\sqrt{2}|0\rangle + i/\sqrt{2}|1\rangle$

$0 \rightarrow 0$ ja $1 \rightarrow 1$ tõenäosusamplituudiga $i/\sqrt{2}$

$0 \rightarrow 1$ ja $1 \rightarrow 0$ tõenäosusamplituudiga $1/\sqrt{2}$

not = kaks √not kvantfunktsiooni järjest :

$0 \rightarrow 0$ on summa kahest teest $0 \rightarrow 0 \rightarrow 0$ ja $0 \rightarrow 1 \rightarrow 0$

tõenäosus $\left| (i/\sqrt{2})^2 + (1/\sqrt{2})^2 \right|^2 = \left| -1/2 + 1/2 \right|^2 = 0$

Kvantregister

- Kahebitine kvantregister on nelja vektori superpositsioon:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

- Normeerimistingimus

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

- Üks *n*-bitine kvantregister sisaldab 2^n erinevat kordajat (kompleksarvu).

Kvantparallelism

- Kvantregistri üks teisendus mõjutab kõiki baasivektoreid.
- Enne:

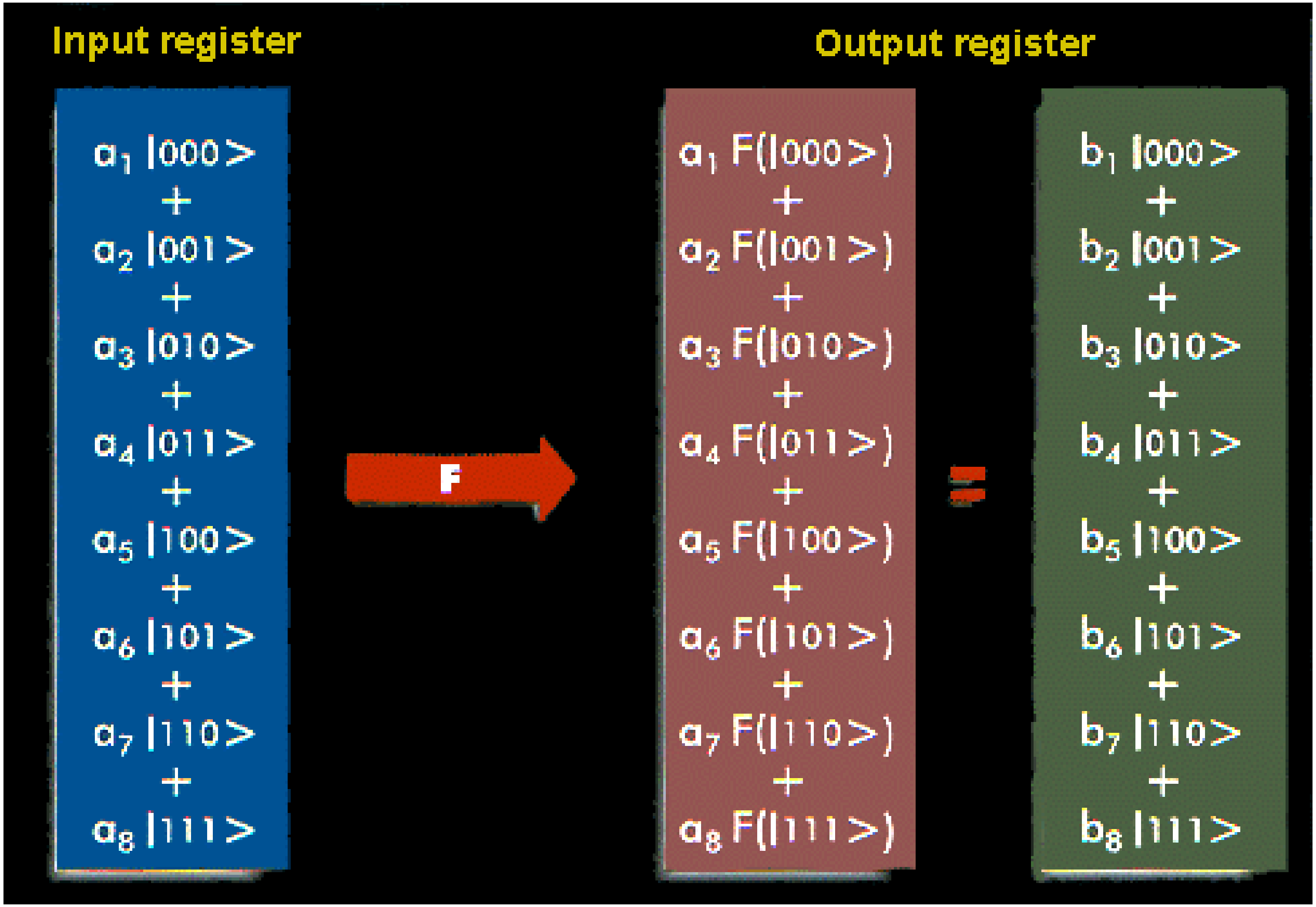
$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

- Pärast:

$$\alpha U|00\rangle + \beta U|01\rangle + \gamma U|10\rangle + \delta U|11\rangle$$

- Loogikatehted korraga kõigi argumentidega

Superpositsioon



Arvutamine kvantarvutil

- Algoleku tekitamine: registri sisuks kõigi baasivektorite ühtlane superpositsioon

  $$1/\sqrt{2^n}\ (|0000\rangle + |0001\rangle + \ldots + |1111\rangle)$$

- Tehted unitaarteisendustena: kvantseaduste järgi peavad kõik tehted olema pööratavad
- Mõõtmine: registri sisu mõõdetakse baasil

  $$|0000\rangle, |0001\rangle, \ldots, |1111\rangle$$

- Arvutamise komplitseeritus

  n-qbitine arvuti arvutab 2^n kompleksväärtust, aga need väärtused on seotud

  - ruutude summa = 1
  - teisendused muudavad korraga kõiki väärtusi

Shori algoritm

- Taandab tegurdamise perioodi leidmise ülesandele
- Leiab n-bitise arvu M tegurid O(*n*) keerukusega, kasutades 2*n* qbitist kvantarvutit

Põimolek (*entangled state*)

- Kaks kvantsüsteemi võivad olla ühises olekus, nii et ühe mõõtmine mõjutab ka teist isegi suure vahemaa tagant.
- Näiteks

$$1/\sqrt{2}\ |00\rangle + 1/\sqrt{2}\ |11\rangle$$

- Põimolekute tõttu on kvantsuperpositsiooni kordajad üksteisest sõltuvad.

Kvantkrüptograafia

- Printsiip: pealtkuulamine on võimalik ainult signaali taastamatu rikkumise hinnaga.
- On olemas tõestatavalt turvaline võtmekehtestusprotokoll (BB84)
- Edastada tuleb vaid polariseeritud footoneid
- Katseid korraldatud mitmete kilomeetrite kaugusele

Turvaline kvantkanal

- Edastatakse footoneid polarisatsiooniga 0, 45, 90 ja 135 kraadi
- Vastuvõtja saab mõõta süsteemis 0, 90 või 45, 135
- Saatja saadab suvalise polarisatsiooniga footoneid, pidades järjekorra meeles
- Vastuvõtja mõõdab juhuslikult ühes või teises süsteemis, peab meeles tulemused ja avaldab mõõtmiseks kasutatud süsteemide järgnevuse
- Saatja teatab millised mõõtmised olid õiges süsteemis
- Vastuvõtja avaldab alamhulga õigeid bitte, et tuvastada kanali võltsimist

Raskused

- Mõõtmine hävitab kvantsuperpositsiooni
- Dekoherents: vastasmõju keskkonnaga, andmete lugemisel
- Tehnoloogilised raskused

Ideaalne kvantarvuti

- skaleerub füüsiliselt qbittide osas
- qbitid saab algväärtustada suvalisele väärtusele
- operatsioonid on kiiremad kui *dekoherents*
- turing-täielik teisenduste komplekt
- qbitte saab lugeda lihtsalt

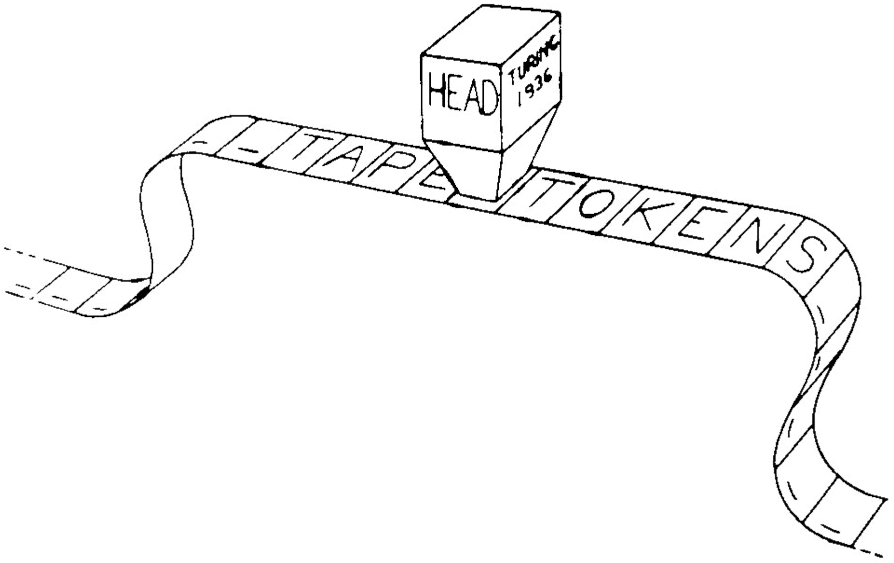
Kvantarvuti v digitaalne arvuti

- Kvantarvuti annab polünomiaalse keerukusega tegurdamis- ja diskreetse logaritmi algoritmi
- Mõlemad algoritmid on NP. Pole suudetud näidata, et nad oleks NPC ja usutakse, et need pole NPC
- Pole suudetud näidata, et kvantarvuti lahendaks polünomiaalses ajas NPC ülesannet, kui pole realiseeritavad mittelineaarsed unitaarteisendused.
- Kvantarvuti ei laienda arvutatavate probleemide hulka - kvantarvutit saab simuleerida Turingi masinal.

Turingi masin

- Minimaalne andmestruktuuride esitus – string
- Minimaalne arvutusmasin – Turingi masin
  - Andmed – lõpmatu pikkusega lint
  - Käsud – loe, kirjuta, liigu paremale, liigu vasakule
  - Programm – lõplik olekumasin

- Arvutatavuse alusformalism



Church-Turing tees

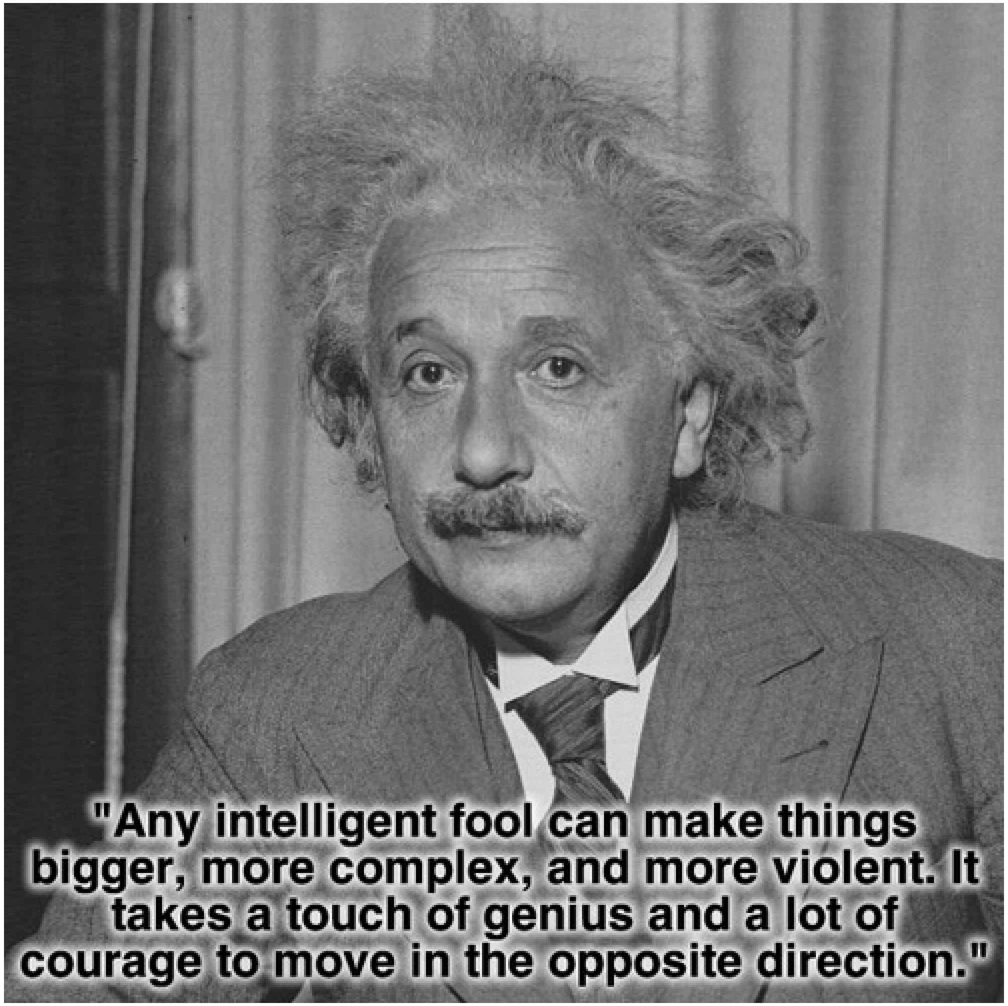
Kõik, mis on efektiivselt (algoritmiliselt) arvutatav, on arvutatav Turingi masinaga (1930)

Alternatiivsed arvutatavusteooriad

- Rekursiivsed funktsioonid (Kleene)
- Lambdaarvutus (Church)

Need on omavahel ekvivalentsed

- Paraleelarvutus, kvantarvutus jt mudelid ei lisa midagi arvutatavate funktsioonide hulka


"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius and a lot of courage to move in the opposite direction."