How to apply de Bruijn graphs to genome assembly

Phillip E C Compeau, Pavel A Pevzner & Glenn Tesler

A mathematical concept known as a de Bruijn graph turns the formidable challenge of assembling a contiguous genome from billions of short sequencing reads into a tractable computational problem.

The development of algorithmic ideas for next-generation sequencing can be traced back 300 years to the Prussian city of Königsberg (present-day Kaliningrad, Russia), where seven bridges joined the four parts of the city located on opposing banks of the Pregel River and two river islands (**Fig. 1a**). At the time, Königsberg's residents enjoyed strolling through their city, and they wondered if every part of the city could be visited by walking across each of the seven bridges exactly once and returning to one's starting location. The solution came in 1735, when the great mathematician Leonhard Euler[1] made a conceptual breakthrough that would solve this 'Bridges of Königsberg problem'. Euler's first insight was to represent each landmass as a point (called a node) and each bridge as a line segment (called an edge) connecting the appropriate two points. This creates a graph—a network of nodes connected by edges (**Fig. 1b**). By describing a procedure for determining whether an arbitrary graph contains a path that visits every edge exactly once and returns to where it started, Euler not only resolved the Bridges of Königsberg problem but also effectively launched the entire branch of mathematics known today as graph theory[2].

Since Euler's original description, the use of graph theory has turned out to have many additional practical applications, most of which have greater scientific importance than the development of walking itineraries. Specifically, Euler's ideas were subsequently adapted by Dutch mathematician Nicolaas de Bruijn to find a cyclic sequence of letters taken from a given alphabet for which every possible word of a certain length (*k*) appears as a string of consecutive characters in the cyclic sequence exactly once (**Box 1** and **Fig. 2**). Application of the de Bruijn graph has also proven invaluable in the field of molecular biology where researchers are faced with the problem of assembling billions of short sequencing reads into a single genome. In the following article, we describe the problems faced when constructing a genome and how the de Bruijn graph approach can be applied to assemble short-read sequences.

Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

Phillip E. C. Compeau and Glenn Tesler are in the Department of Mathematics, University of California San Diego, La Jolla, California, USA, and Pavel A. Pevzner is in the Department of Computer Science and

Problems with alignment-based assembly

To illustrate why graphs are useful for TGCAATG) sequenced from a small circular genome, ATGGCGTGCA (**Fig. 3a**). Current next-generation sequencing methods produce reads that vary in length, but the most popular technology generates ~100-nucleotide reads. A straightforward method for assembling reads into longer contiguous sequences—and the one used for assembling the human genome[3,4] in 2001 as well as for all other projects based on Sanger sequencing—uses a graph in which each read is represented by a node and overlap between reads is represented by an arrow (called a 'directed edge') joining two reads. For instance, two nodes representing reads may be connected with a directed edge if the reads overlap by at least five nucleotides (**Fig. 3b**).

Visualizing an ant walking along the edges of this graph provides an aid for understanding a broad class of algorithms used to derive insights from graphs. In the case of genome assembly, the ant's path traces a series of overlapping reads, and thus represents a can-

Specifically, if the ant fol-

GCGT → GGCGTGC →
CAATG → CAATGGC →

P.Compeau, P.Pevzner & G.Tesler:
How to apply de Bruijn graphs to genome assembly.
Nature Biotechnology 29 (nov. 2011) 879-911.

Leiden Center For Natural Computing

Molecular Computing
(Rozenberg)
Evolutionary Algorithms
Neural Networks
(Bäck, Kok, Blockeel)

www.lcnc.nl

Last updated on 9 December 2002

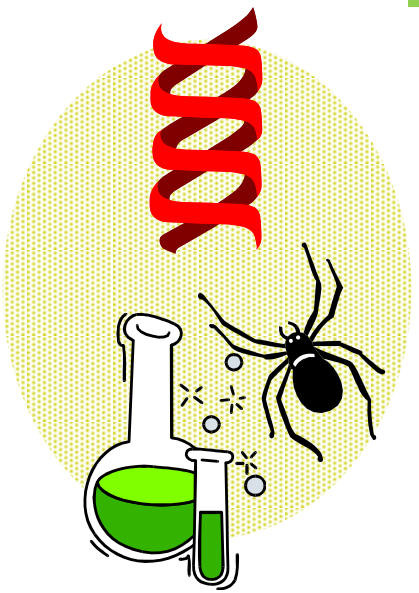
natural computation

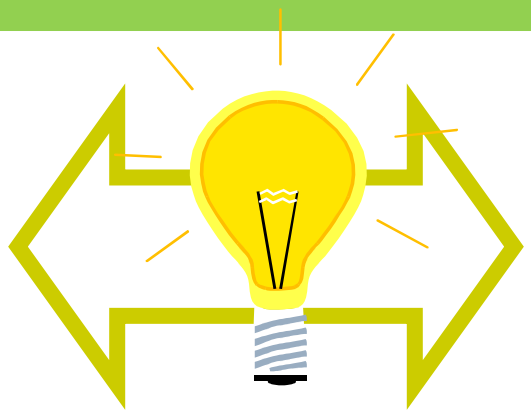
'sorting' DNA / self assembly

understanding nature
as a computational process

neural netw & genetic alg

bio inspired computing





bio hardware

DNA computing

bio-informatics

comp mol biol / beeldanalyse

Nicolaas Govert "Dick" de Bruijn

(9 July 1918 – 17 February 2012)

http://www.win.tue.nl/automath/images/photos/deBruijn.jpg

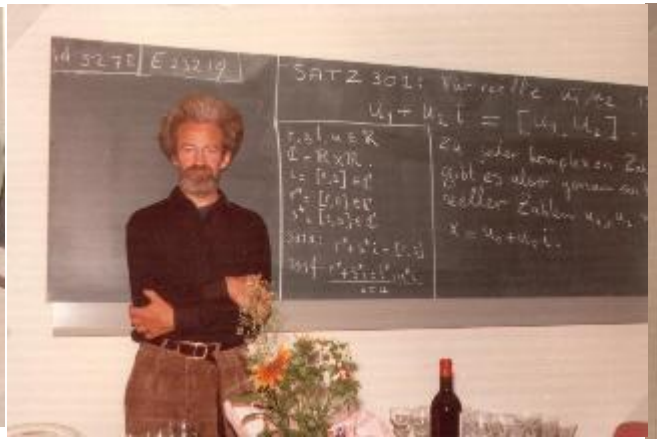
werk

- De Bruijn sequence (1946)
- BEST Theorem (1951)
- Automath (1967)

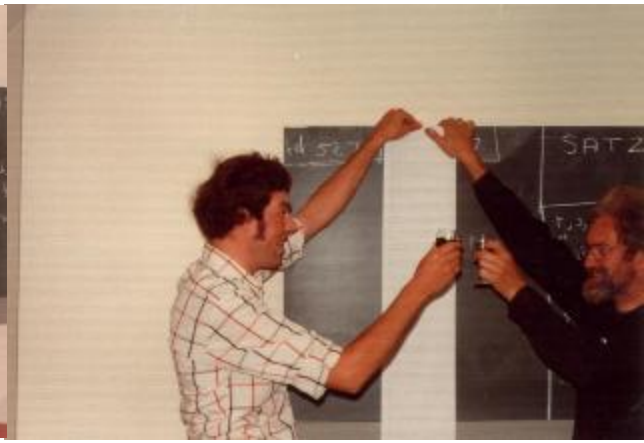
Automath Archive

The last theorem of Edmund Landau's book 'Grundlagen der Analyses' is entered...

From left to right: (probably) Bram Kornaat, Bert Jutting, Ids Zandleven, Roel de Vrijer, prof. de Bruijn



The last theorem ('Satz 301') of Landau's book explained by Bert Jutting



The situation is tensed, something does not appear to go well... It appears to be succesful after all. Pressing the enter button and..... The print with the proof of the last theorem!

GRUNDLAGEN DER ANALYSIS

(DAS RECHNEN MIT GANZEN, RATIONALEN, IRRATIONALEN, KOMPLEXEN ZAHLEN)

ERGÄNZUNG ZU DEN LEHRBÜCHERN DER DIFFERENTIAL- UND INTEGRALRECHNUNG

VON

EDMUND LANDAU
PROFESSOR
AN DER UNIVERSITÄT GÖTTINGEN

1930
AKADEMISCHE VERLAGSGESELLSCHAFT M. B. H.
LEIPZIG

erkennt man an den ausgerechneten Ausdrücken

$$(\mathfrak{x}\mathfrak{y})\mathfrak{z} = \mathfrak{x}(\mathfrak{y}\mathfrak{z}).$$

Satz 227 (distributives Gesetz):

$$\mathfrak{x}(\mathfrak{y}+\mathfrak{z}) = \mathfrak{x}\mathfrak{y}+\mathfrak{x}\mathfrak{z}.$$

Beweis:

$$\begin{aligned}
&[\Xi_1, \Xi_2]([H_1, H_2]+[Z_1, Z_2]) = [\Xi_1, \Xi_2][H_1+Z_1, H_2+Z_2]\\
&= [\Xi_1(H_1+Z_1)-\Xi_2(H_2+Z_2),\ \Xi_1(H_2+Z_2)+\Xi_2(H_1+Z_1)]\\
&= [(\Xi_1 H_1+\Xi_1 Z_1)+(-(\Xi_2 H_2)+(-(\Xi_2 Z_2))),\ (\Xi_1 H_2+\Xi_1 Z_2)+(\Xi_2 H_1+\Xi_2 Z_1)]\\
&= [(\Xi_1 H_1-\Xi_2 H_2)+(\Xi_1 Z_1-\Xi_2 Z_2),\ (\Xi_1 H_2+\Xi_2 H_1)+(\Xi_1 Z_2+\Xi_2 Z_1)]\\
&= [\Xi_1 H_1-\Xi_2 H_2,\ \Xi_1 H_2+\Xi_2 H_1]+[\Xi_1 Z_1-\Xi_2 Z_2,\ \Xi_1 Z_2+\Xi_2 Z_1]\\
&= [\Xi_1, \Xi_2][H_1, H_2]+[\Xi_1, \Xi_2][Z_1, Z_2].
\end{aligned}$$

Satz 228: $\mathfrak{x}(\mathfrak{y}-\mathfrak{z}) = \mathfrak{x}\mathfrak{y}-\mathfrak{x}\mathfrak{z}.$

Beweis: $\mathfrak{x}(\mathfrak{y}-\mathfrak{z}) = \mathfrak{x}(\mathfrak{y}+(-\mathfrak{z})) = \mathfrak{x}\mathfrak{y}+\mathfrak{x}(-\mathfrak{z}) = \mathfrak{x}\mathfrak{y}+(-(\mathfrak{x}\mathfrak{z}))$
$= \mathfrak{x}\mathfrak{y}-\mathfrak{x}\mathfrak{z}.$

Satz 229: *Die Gleichung*

$$\mathfrak{y}\mathfrak{u} = \mathfrak{x},$$

wo $\mathfrak{x}$, $\mathfrak{y}$ *gegeben sind und*

$$\mathfrak{y} \neq \mathfrak{n}$$

ist, hat genau eine Lösung $\mathfrak{u}$.

Beweis: 1) Es gibt höchstens eine Lösung; denn aus

$$\mathfrak{y}\mathfrak{u}_1 = \mathfrak{x} = \mathfrak{y}\mathfrak{u}_2$$

folgt

$$\mathfrak{n} = \mathfrak{y}\mathfrak{u}_1-\mathfrak{y}\mathfrak{u}_2 = \mathfrak{y}(\mathfrak{u}_1-\mathfrak{u}_2),$$

Example of a text in the formal language AUTOMATH, L.S. van Benthem Jutting, August 1970

0	x	:=	—	Nat	
x	y	:=	—	Nat	
y	n	:=	—	x≠y	
n	i	:=	—	{x}s={y}s	
i	r1	:=	{i}{y}{x}Ax4	x=y	
i	r2	:=	MP(x=y,CON,r1,n)	CON	
n	Th1	:=	[t,{x}s={y}s]r2(t)	{x}s≠{y}s	*theorem:* $x \neq y \vdash x' \neq y'$
x	prop1	:=	{x}s≠x	type	
0	r3	:=	{1}Ax3	prop1(1)	
x	p	:=	—	prop1(x)	
p	r4	:=	Th1({x}s,x,p)	prop1({x}s)	
x	Th2	:=	lemma([t,Nat]prop1(t),r3,[t,Nat][u,prop1(t)]r4(t,u),x)	{x}s≠x	*theorem:* $x' \neq x$
x	prop2	:=	x=1 ∨ ∃([t,Nat]x={t}s)	type	
0	r5	:=	OR1(1=1,∃([t,Nat]1={t}s),REF(Nat,1))	prop2(1)	
x	r6	:=	SOME1(Nat,[t,Nat]{x}s={t}s,x,REF(Nat,{x}s))	∃([t,Nat]{x}s={t}s)	
x	r7	:=	OR2({x}s=1,∃([t,Nat]{x}s={t}s,r6)	prop2({x}s)	

http://www.win.tue.nl/automath/archive/pdf/aut012.pdf

werk

- De Bruijn sequence (1946)
- BEST Theorem (1951)
- Automath (1967)

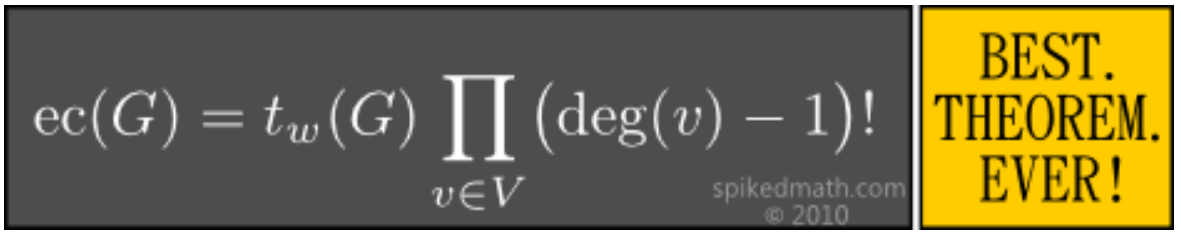
BEST

- BEST Theorem (1951)

de Bruijn, van Aardenne-Ehrenfest, Smith and Tutte

euler cycles

http://spikedmath.com/327.html



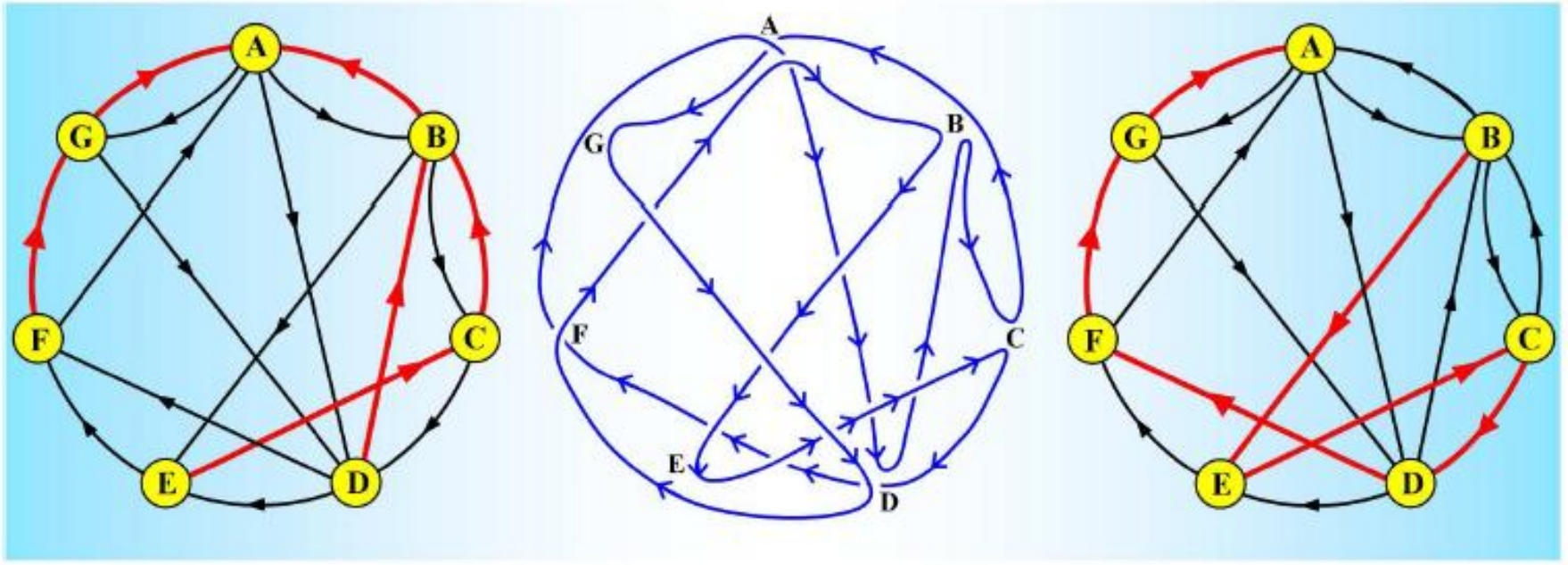
spanning trees
‘into’ w

Theorem of the Day

The BEST Theorem *Let $G = (V, E)$ be a directed graph in which, for each vertex v in V, the indegree and outdegree have the same value, $d(v)$, say. Then G has a directed Euler tour: a closed walk which passes each edge exactly once; let $\varepsilon(G)$ denote the number of such tours. Then, for any fixed vertex x,*

$$\varepsilon(G) = t_x \prod_{v \in V} (d(v) - 1)!$$

where t_x denotes the number of those spanning trees of G in which every vertex has a directed path to x.



speciaal geval

d(v)=2

http://myweb.lsbu.ac.uk/~whittyr/MathSci/TheoremOfTheDay/CombinatorialTheory/BEST/TotDBEST.pdf

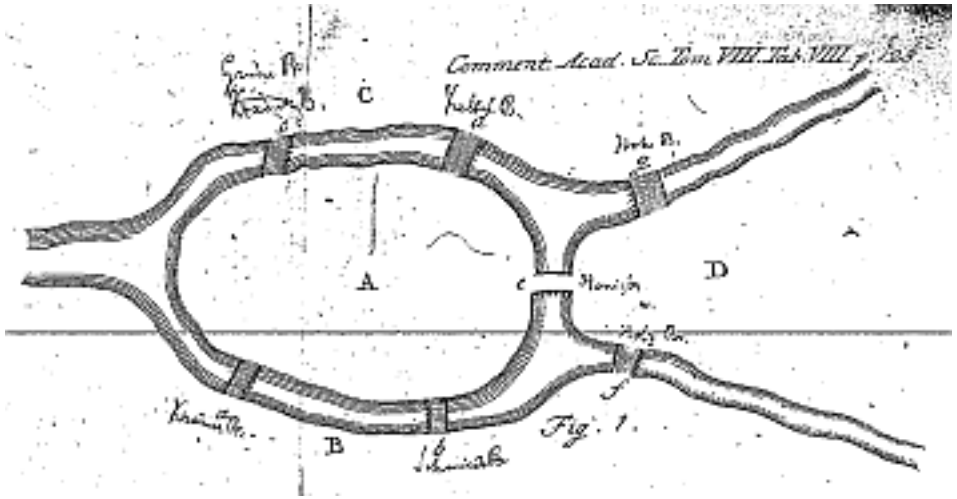
werk

- De Bruijn sequence (1946)
- BEST Theorem (1951)
- Automath (1967)

Königsbergen

L. Euler
Solutio problematis ad geometriam situs pertinentis,
Comment. Academiae Sci. I. Petropolitanae 8 (1736) 128-140

SOLVTIO PROBLEMATIS
AD
GEOMETRIAM SITVS
PERTINENTIS.
AVCTORE
Leonh. Eulero.



http://www.math.dartmouth.edu/~euler/docs/originals/E053.pdf

§8.5 rondwandeling

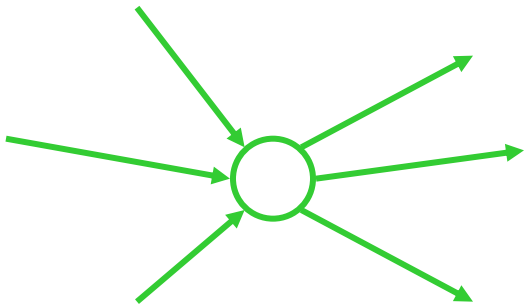
Een *Euler trail* is een gesloten wandeling die elke lijn precies één keer bevat. 'traversable'

trail 'all edges distinct'

- zeven bruggenprobleem van Königsbergen

Theorem 8.3

Een samenhangende graaf heeft een Euler trail desdals elk punt even graad heeft.



http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Euler & Hamilton

- Euler graaf
  Leonhard Euler (1736) Königsberger bruggen
  elke *lijn* precies een keer ‘trail’

  - eenvoudige karakterisatie Theorem 8.3
  - efficient te herkennen

- Hamilton graaf
  William Rowan Hamilton (1858) Icosian Game
  elke *knoop* precies een keer
  ‘handelsreiziger’

Ore (1960) A graph with n vertices (n>3) is Hamiltonian if, for each pair of non-adjacent vertices, the sum of their degrees is n or greater

  - geen karakterisatie
  - NP compleet

VK 17 mrt 2012

WISKUNDEMEISJES

Wat er zo bijzonder is aan 00001111011001010000

Vorige maand overleed de Nederlandse wiskundige N.G. de Bruijn. Ik ontdekte zijn speelse wiskunde jaren geleden, toen een vriend mijn hulp vroeg bij het kraken van een code. Hij wilde als grap de boodschap op iemands telefoonbeantwoorder veranderen. Zijn slachtoffer had een destijds hypermodern apparaat, dat je vanaf elke telefoon op afstand kon bedienen, mits je de geheime viercijferige code invoerde. Je mocht daarbij net zoveel cijfers intoetsen als je wilde. Dus als de juiste code 4567 was, dan kwam je met 1234567 of 4444567 in het systeem. Mijn vriend vroeg zich af wat de snelste manier was om de 10.000 mogelijke codes te proberen. Domweg alle mogelijkheden achter elkaar intoetsen gaf een reeks van 40.000 cijfers. Wist ik een wiskundige truc om het sneller te doen?

Ik begon eerst met een eenvoudiger probleem: een zo kort mogelijke rij zoeken met alle viercijferige codes van alleen enen en nullen. In dat geval zijn er zestien verschillende mogelijkheden, en alle mogelijkheden na elkaar proberen geeft dan een reeks van 64 cijfers. Maar ik zag al snel dat het sneller kan doordat codes elkaar mogen overlappen. Toets bijvoorbeeld 000011 en je probeert met zes cijfers meteen drie codes: 0000, 0001 en 0011.

N.G. de Bruijn
Foto Bart van Overbeeke

Als je die overlap optimaal gebruikt, dan zit elk cijfer dat je intoetst in vier verschillende combinaties, behalve de drie cijfers aan het begin en einde. In het beste geval zou je daarom zestien combinaties in negentien cijfers kunnen proppen. Na een tijdje prutsen op een enveloop vond ik het volgende rijtje: 0000111101100101000. Liefhebbers mogen controleren dat in deze negentien cijfers inderdaad alle zestien mogelijke codes zitten.

Voor de telefoonbeantwoorder vermoedde ik dat op een zelfde manier alle tienduizend codes in slechts 10.003 cijfers moesten passen. Maar daar was met pen en papier geen beginnen aan. Toen wees een vriendelijke wiskundige me erop dat N.G. de Bruijn dit soort rijen al uitgebreid had geanalyseerd. Ze dragen nu zelfs zijn naam. De Bruijn bewees dat er altijd een rijtje bestaat waarin elke combinatie precies één keer voorkomt. Het maken van zo'n rij is nog best lastig, maar met wat programmeerwerk vond ik voor mijn vriend inderdaad een reeks van 10.003 cijfers met alle codes voor de telefoonbeantwoorder.

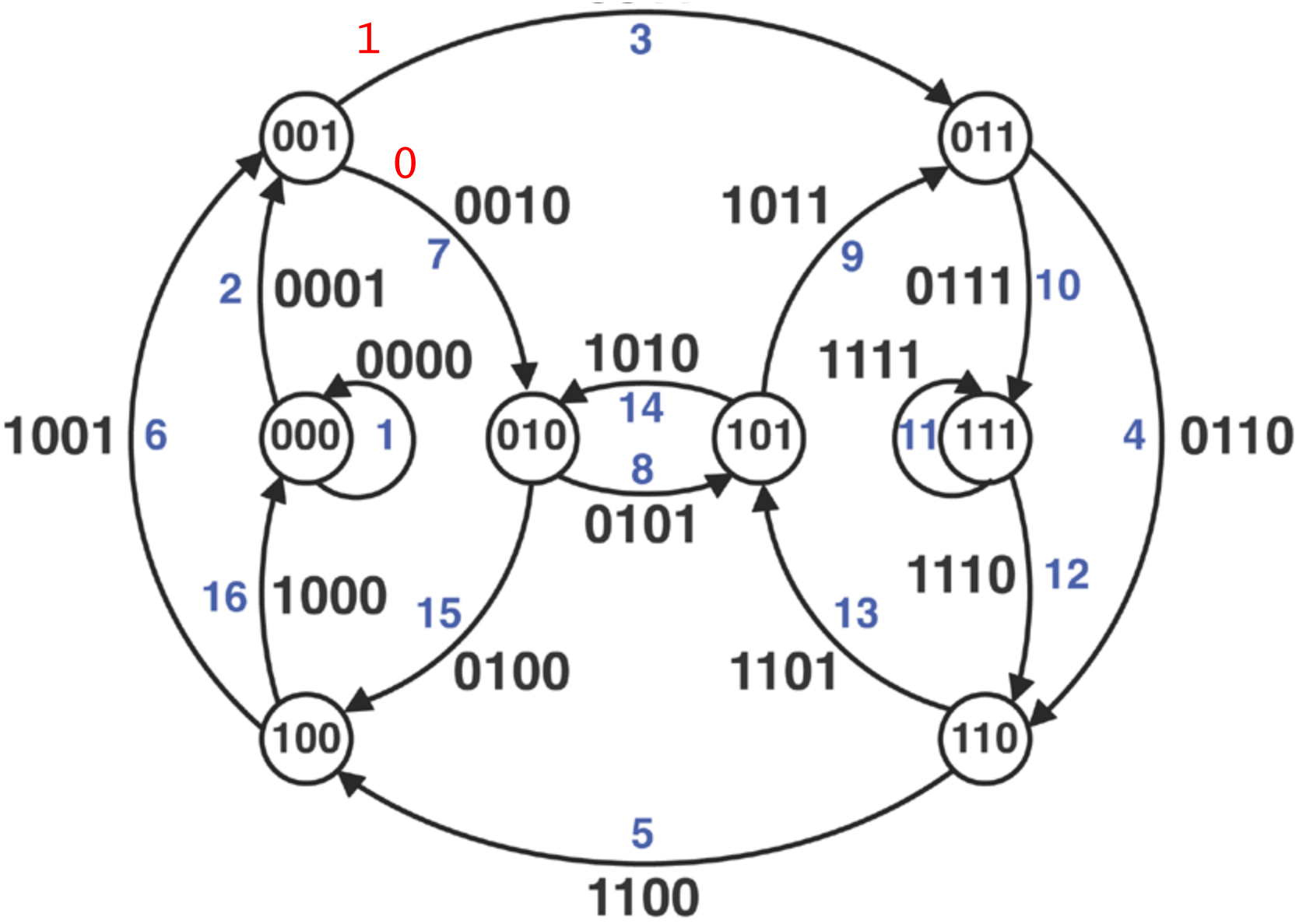
> In het Sanskriet bestond tweeduizend jaar geleden al een De Bruijn-rij

De Bruijn-rijen duiken op onverwachte plekken op. In het Sanskriet was er tweeduizend jaar geleden al een als ezelsbruggetje om namen te onthouden. Tegenwoordig zijn de rijtjes nuttig bij dna-analyse en datacompressie. De mooiste toepassing – naast het kraken van die telefoonbeantwoorder – kwam ik laatst tegen in een goocheltruc. Je kunt er in een pak speelkaarten voor zorgen dat elke zes opeenvolgende kaarten een ander kleurenpatroon hebben (bijvoorbeeld *zrzzrz*, waar *z* een zwarte kaart is en *r* een rode). Een slimme goochelaar laat het dek couperen door een vrijwilliger, vraagt daarna zes mensen om steeds de bovenste kaart te pakken en laat degenen met een rode kaart hun hand opsteken. Uit die minimale informatie kan hij dankzij het unieke zwart-rood-patroon dan precies zeggen welke kaarten de vrijwilligers hebben.

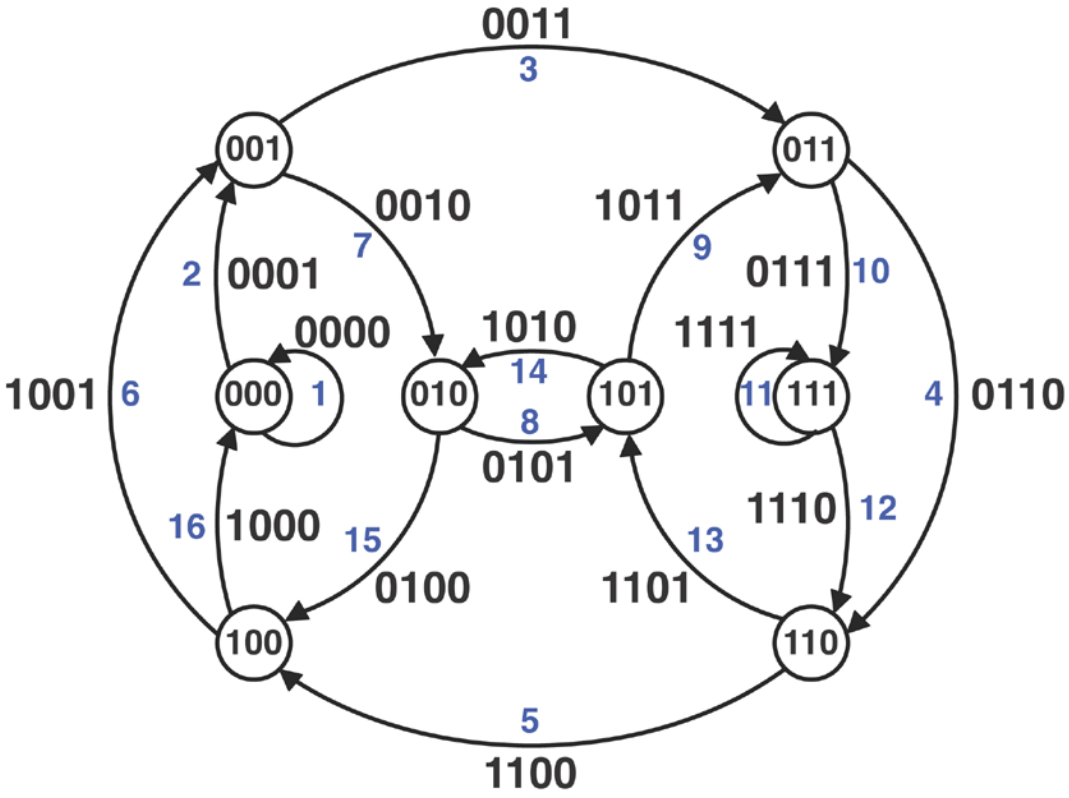
Jammer dat je deze truc alleen goed kunt uitvoeren als je net zo slim bent als N.G. de Bruijn.

Ionica Smeets

0011 de Bruijn graaf



de Bruijn graaf



0000110010111101000

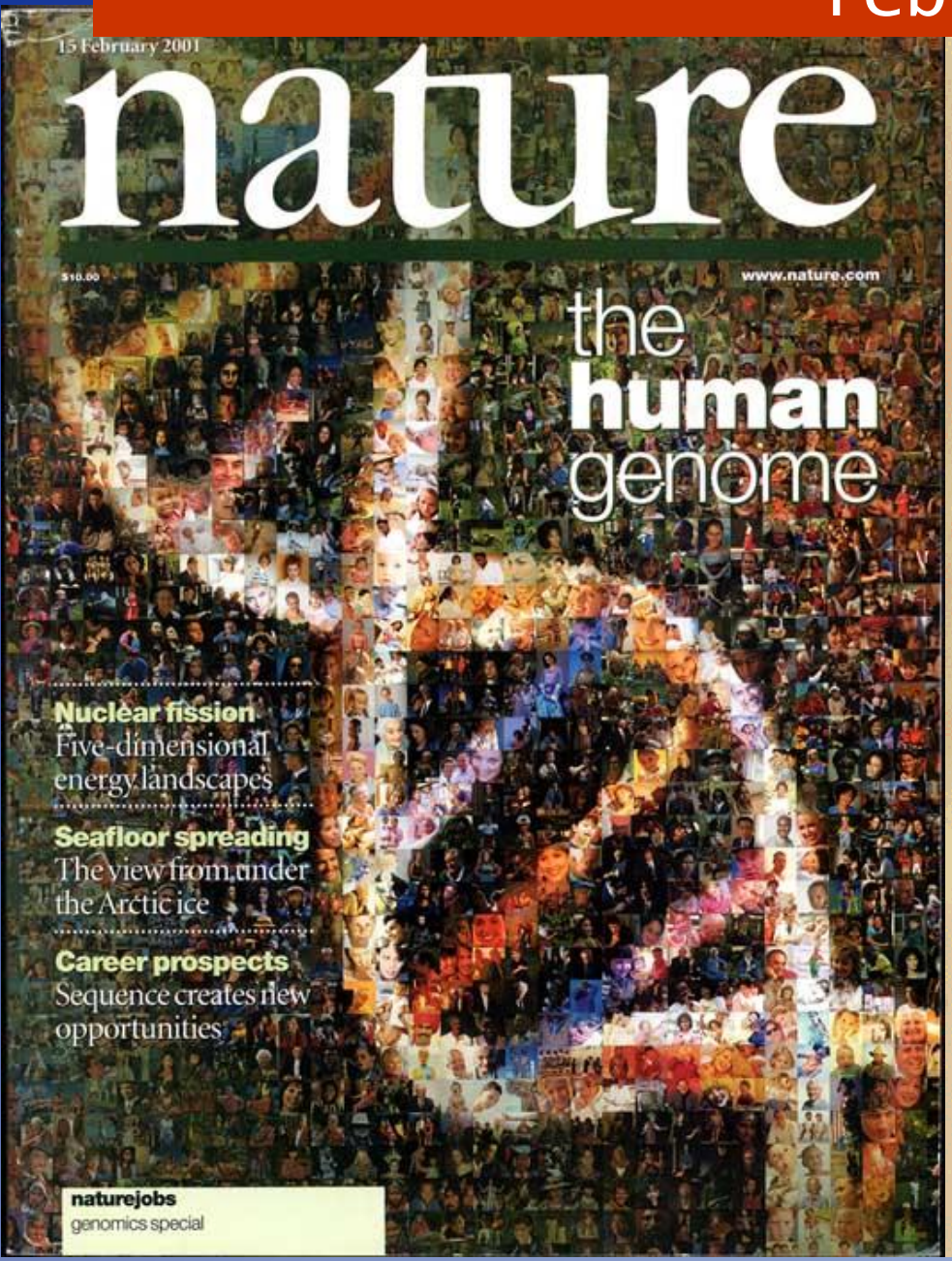
0000111101100101000

de Bruijn rijtjes

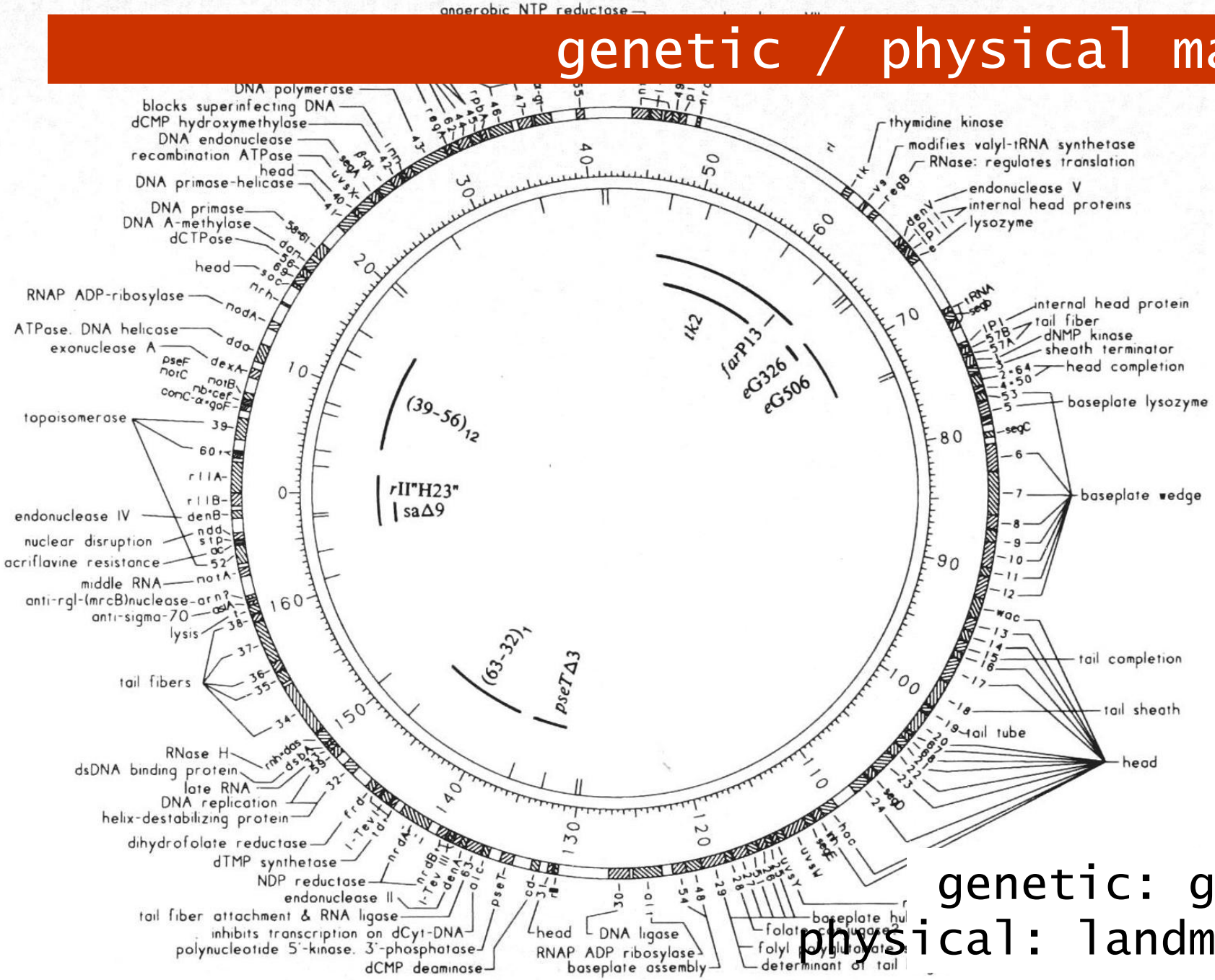
de Bruijn, Nicolaas G. (1946), "A combinatorial problem", Proc. Koninklijke Nederlandse Akademie v. Wetenschappen 49: 758–764, MR0018142, Indagationes Mathematicae 8: 461–467

Flye Sainte-Marie, Camille (1894), "Solution to question nr. 48", L'intermédiaire des Mathématiciens 1: 107–110

feb'01 - human genome

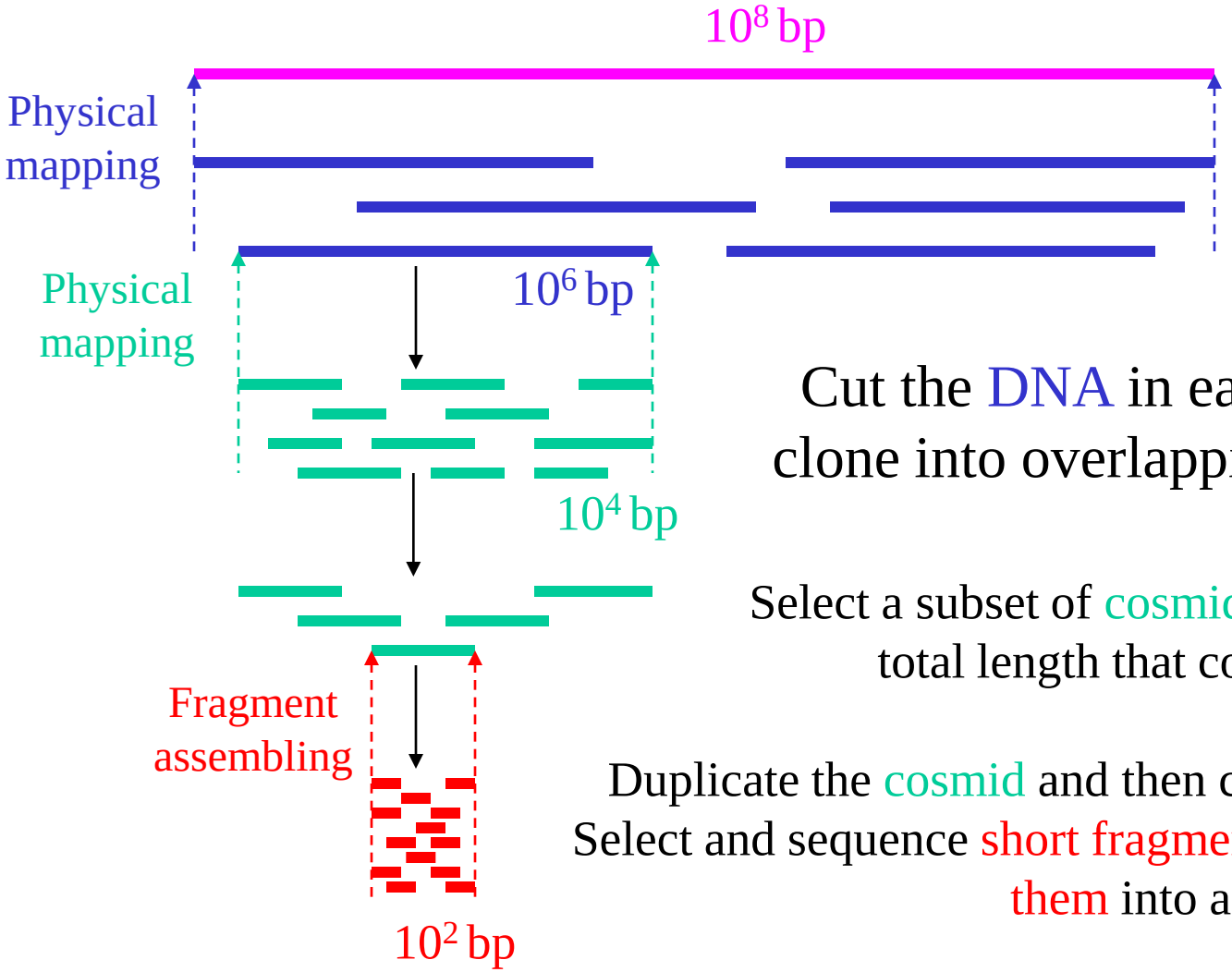


genetic / physical map



genetic: genes
physical: landmarks

physical mapping



C: Full DNA

Cut *C* and clone into overlapping YAC clones.

Cut the DNA in each YAC clone and clone into overlapping cosmid clones.

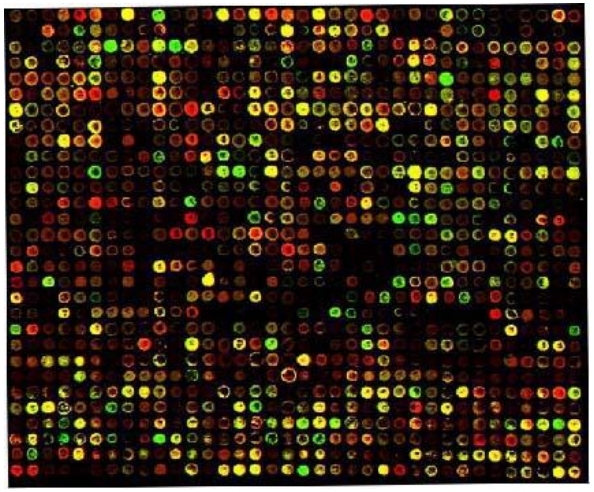
Select a subset of cosmid clones of minimum total length that covers the YAC DNA.

Duplicate the cosmid and then cut the copies randomly. Select and sequence short fragments and then reassemble them into a deduced cosmid string.

sequencing by hybridization

all possible probes of length ℓ
hybridization: determine substrings
reconstruct from (multi-)set of substrings

DNA array



AA	AC	AG	AT
CA	CC	CG	CT
GA	GC	GG	GT
TA	TC	TG	TT

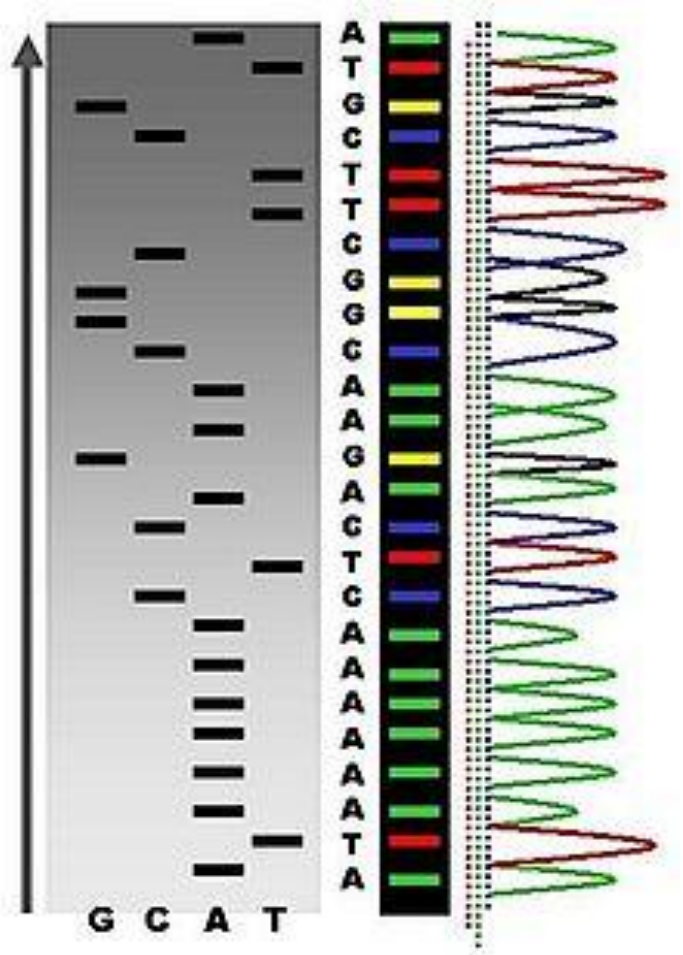
$\ell = 3$

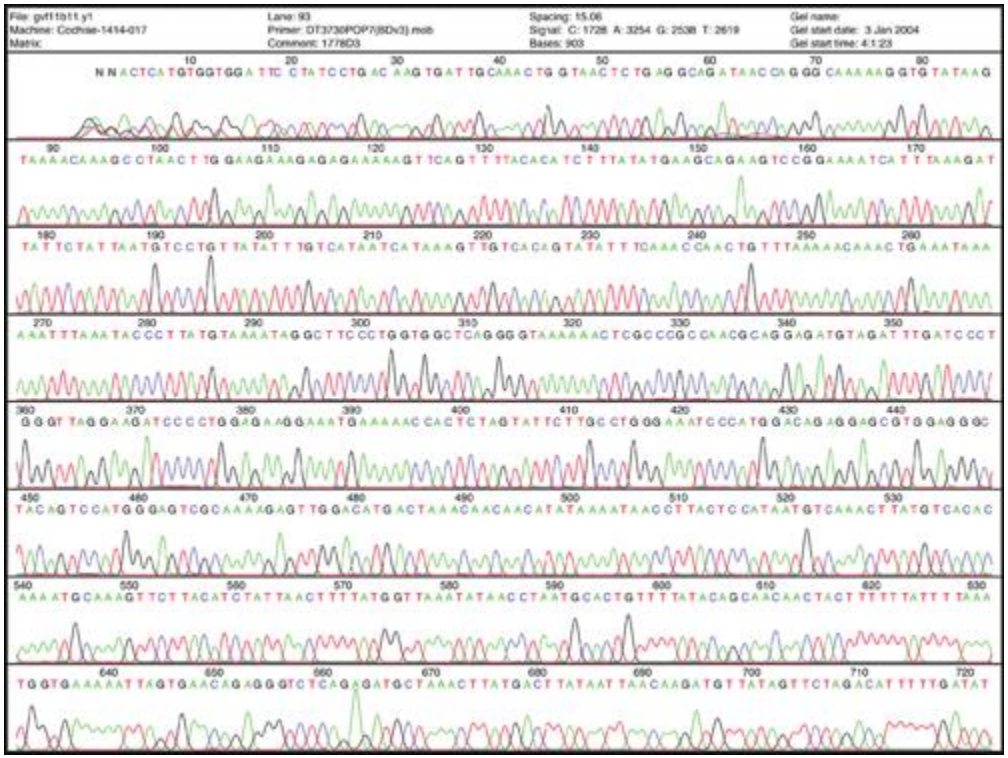
ATTGAC

1990 now ‘short read sequencing’

chain-termination sequencing

(Sanger)
300 – 1000 mer





http://en.wikipedia.org/wiki/DNA_sequencing

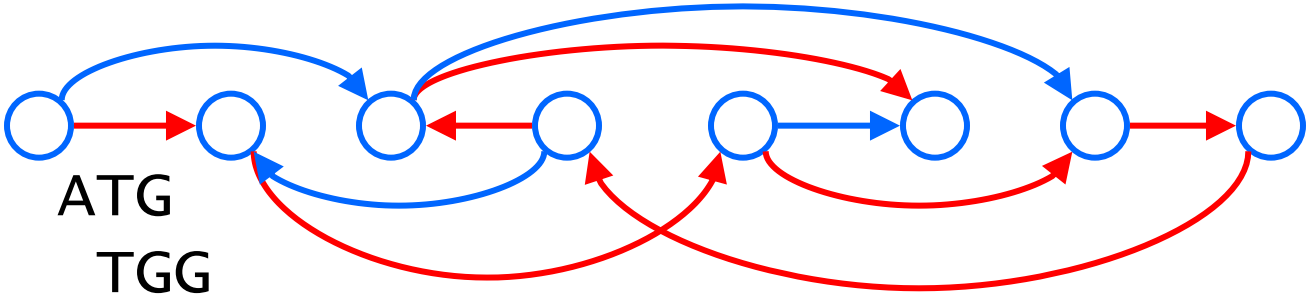
SBH example

as before: overlap graph (not a good choice)

'characteristic triplets'

$\ell = 3$

{ ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT }



ATGGCGTGCA

Hamilton approach: all nodes
(overlap ℓ-1)

triplet=node

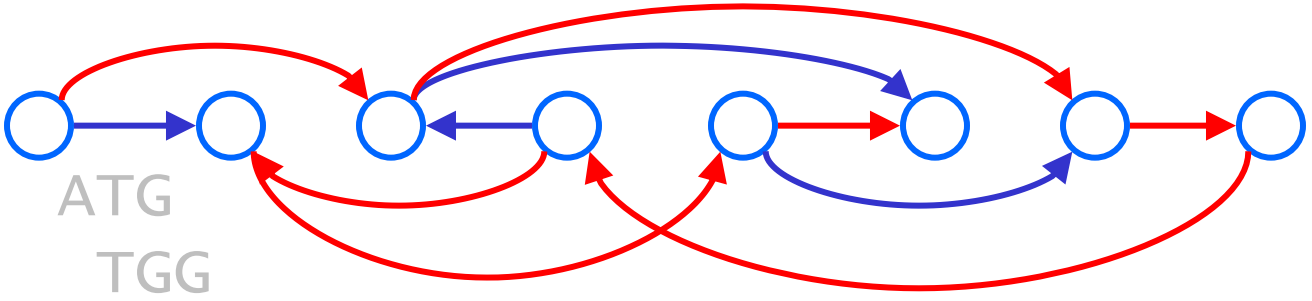
SBH example

as before: overlap graph (not a good choice)

'characteristic triplets'

$\ell = 3$

{ ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT }



ATGCGTGGCA ATGGCGTGCA

another solution

Hamilton approach: all nodes (overlap ℓ-1)

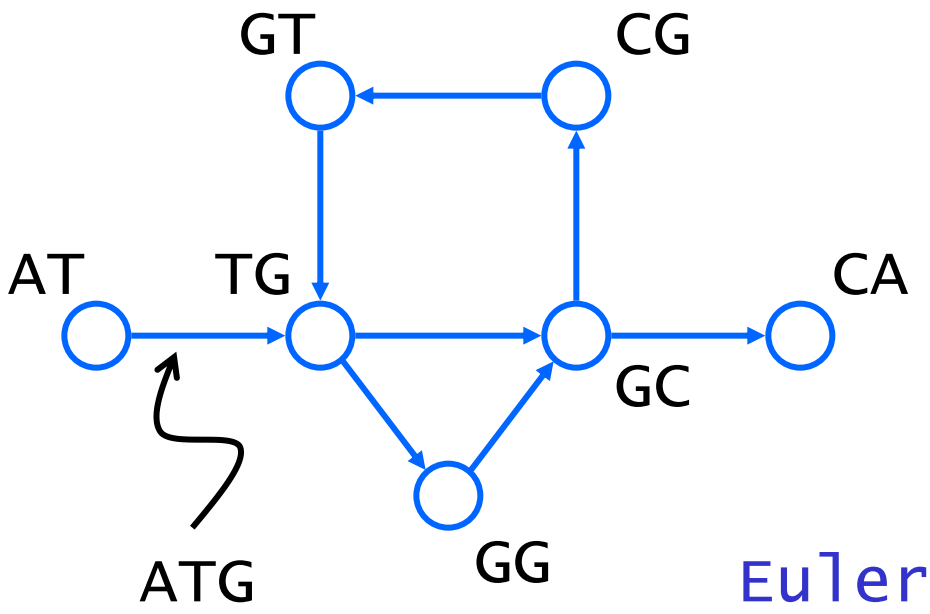
triplet=node

SBH example

we can do better with same problem:

$\ell = 3$

{ ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT }



ATGGCGTGCA

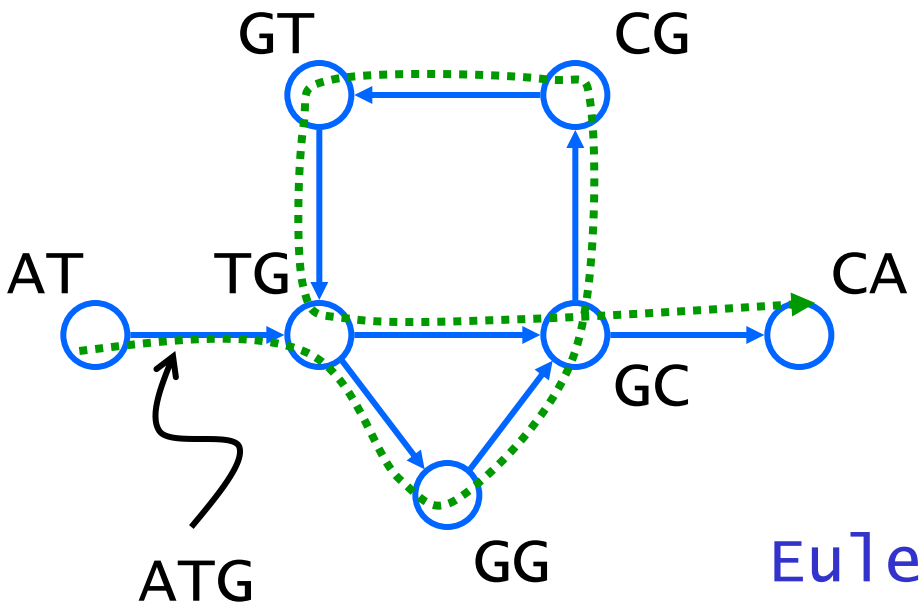
Euler approach: edges
(overlap ℓ-1 = node)
linear ☺

triplet=edge

SBH example

$\ell = 3$

{ ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT }



ATGGCGTGCA

ATGCGTGGCA

Euler approach: edges

even degree nodes
(except start+finish)

DNA assemblers

practical issues

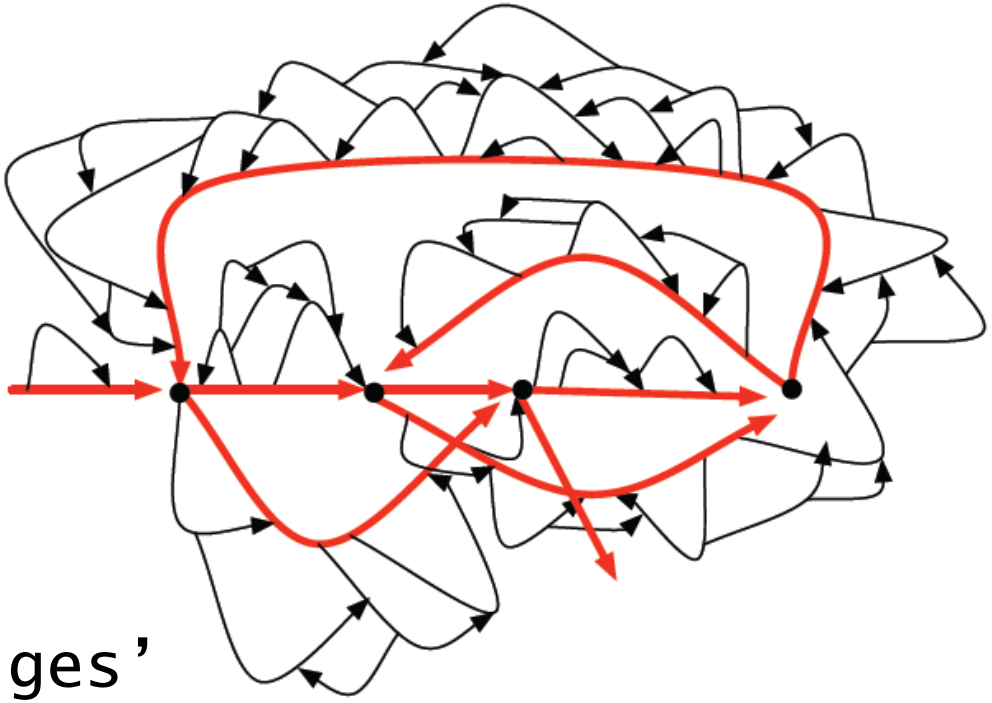
coverage

‘reads’ :
100 mer → 55 mer

errors & repeats
multiplicities, ‘bulges’

paired reads

hardware & software



actg.....ccat

end...

How to apply de Bruijn graphs to genome assembly
P.E.C. Compeau, P.A. Pevzner & G. Tesler
Nature Biotechnology 29 (november 2011)
doi: 10.1038/nbt.2023