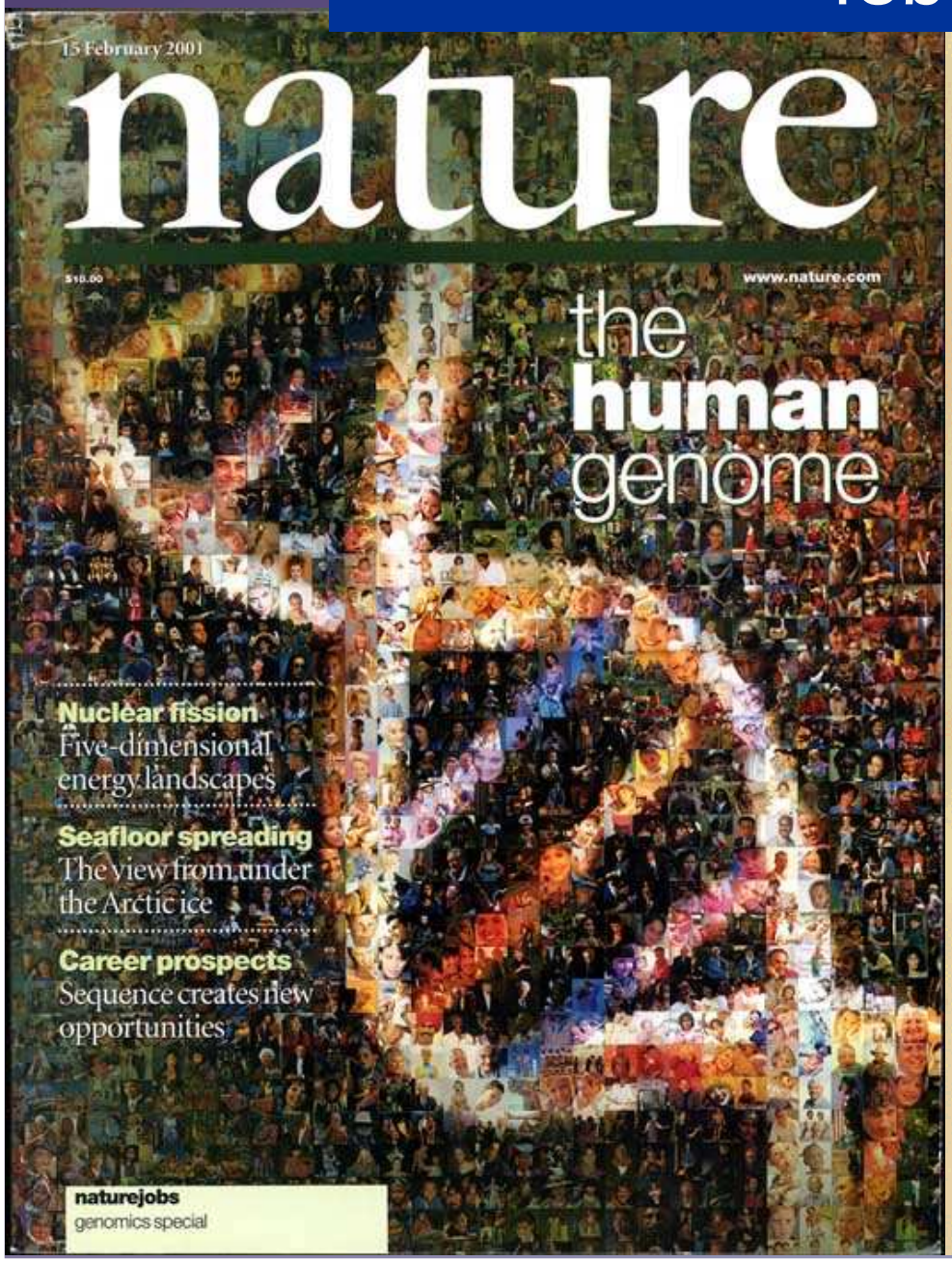
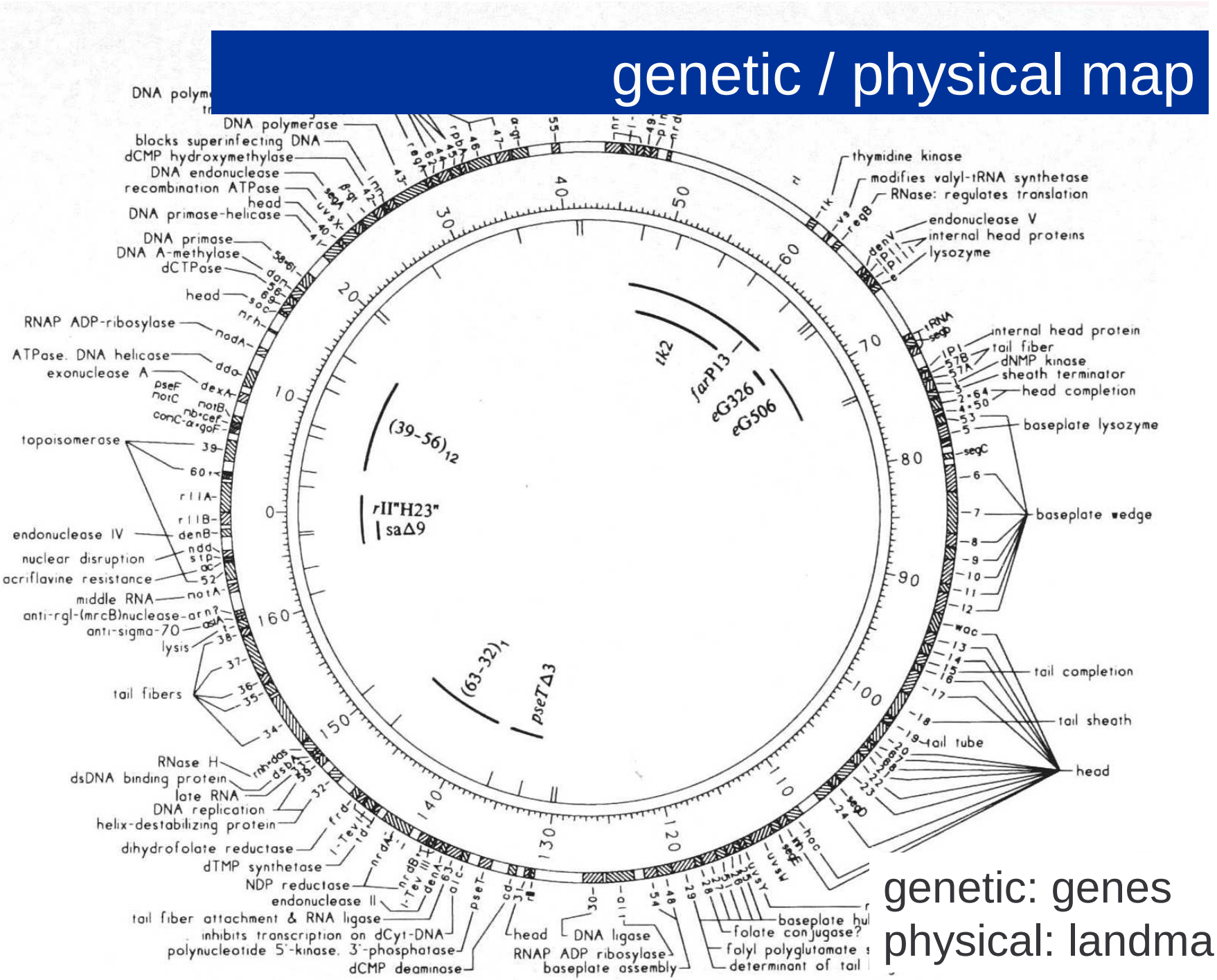
feb’01 - human genome

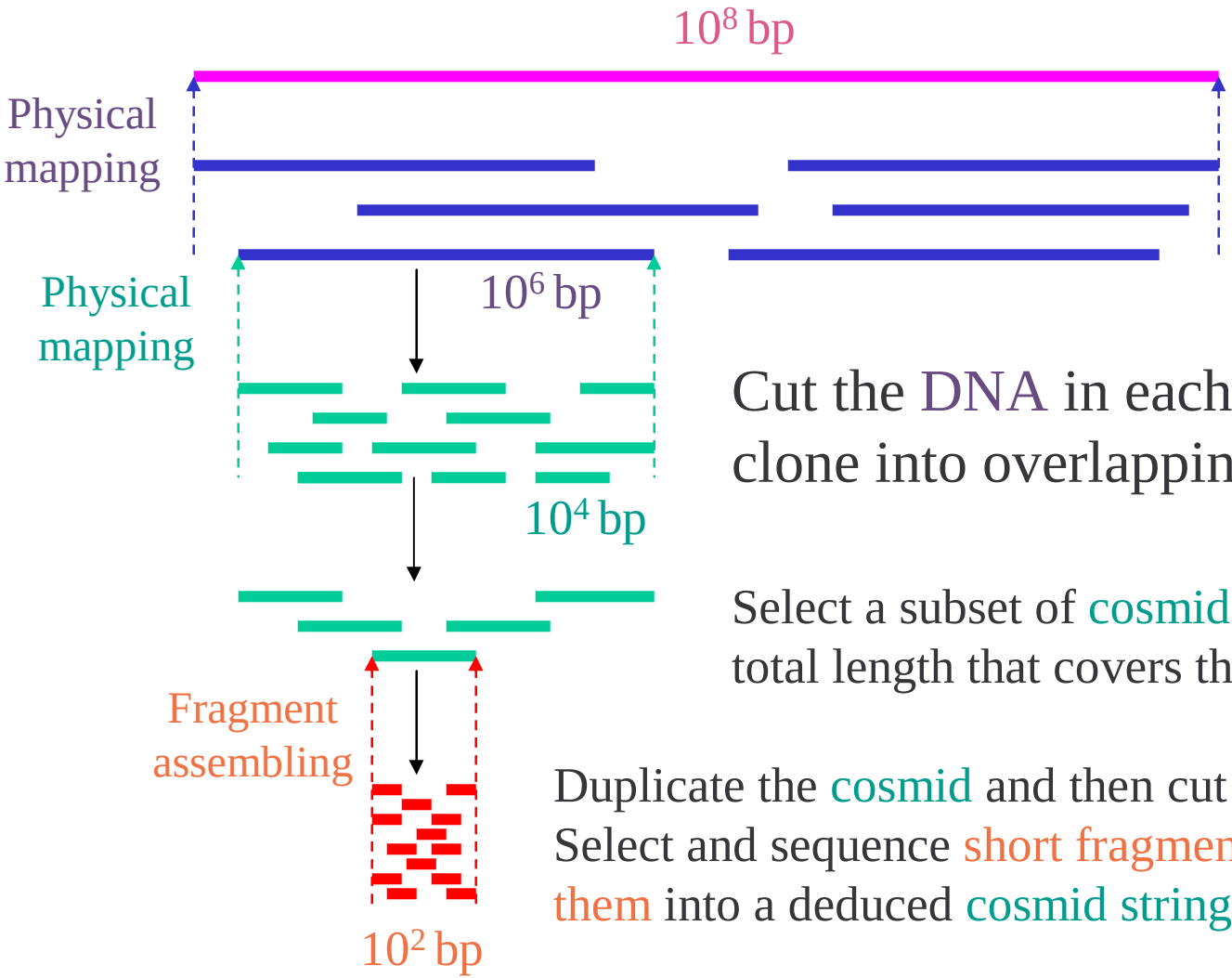


genetic / physical map



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.

overview

physical mapping

location of ‘markers’

- **restriction mapping**

  cutting sites enzymes

  - ✓ double digest problem (NP complete)
  - ✓ partial digest problem
- **hybridization mapping**

  ‘clones’ and ‘probes’

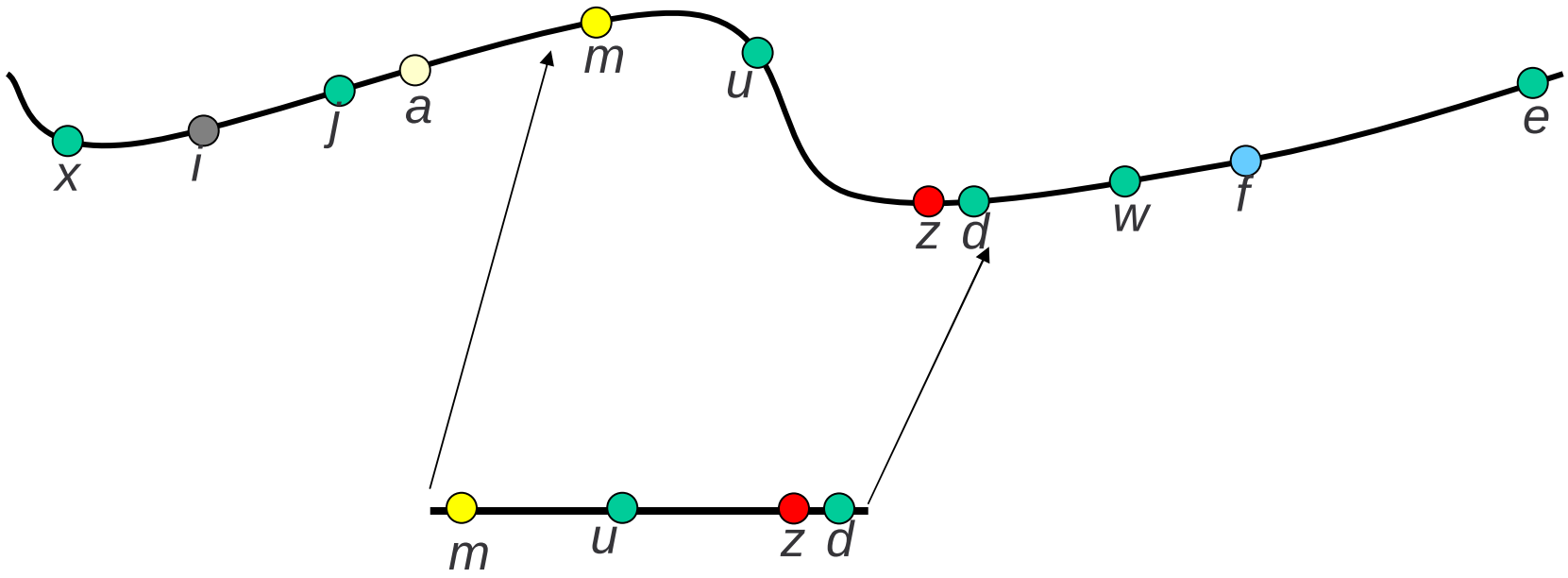
  - ✓ non-unique probes (NP hard)
  - ✓ unique probes (P time) 

fragment assembly

full sequence from fragments

- ✓ shortest superstring
- ✓ overlap graph

using a physical map



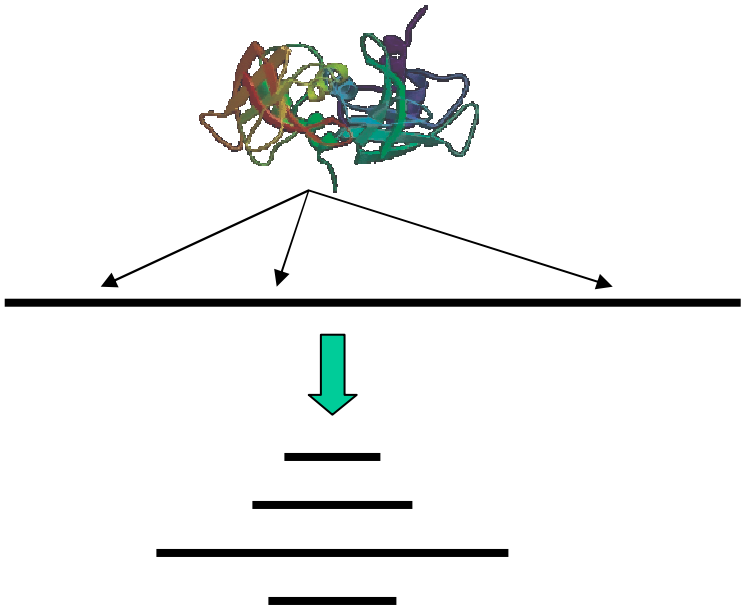
markers: short sequences
- restriction sites
- hybridization sites

landmarks on the genome

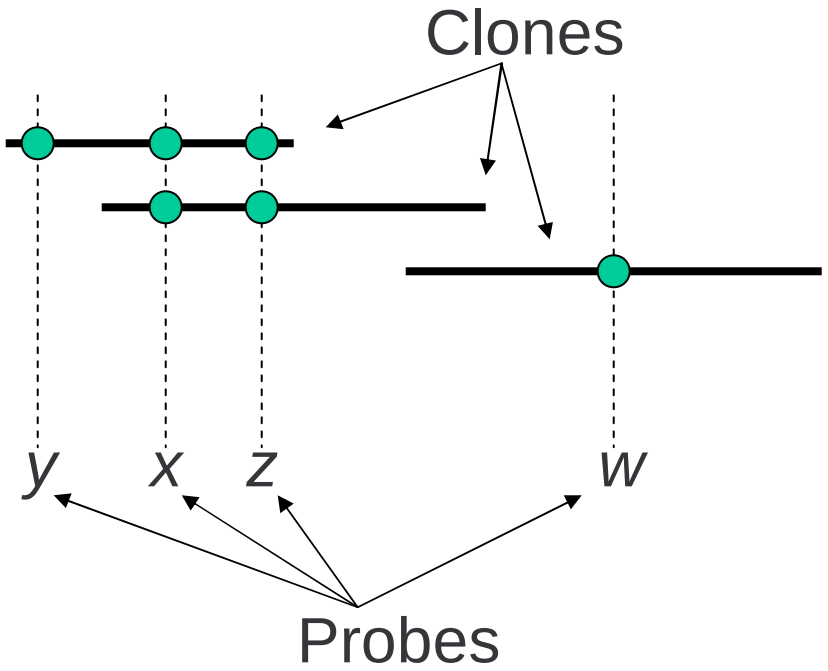
order or location of sequence landmarks

restriction mapping

BamH1 – GGATCC



hybridization mapping

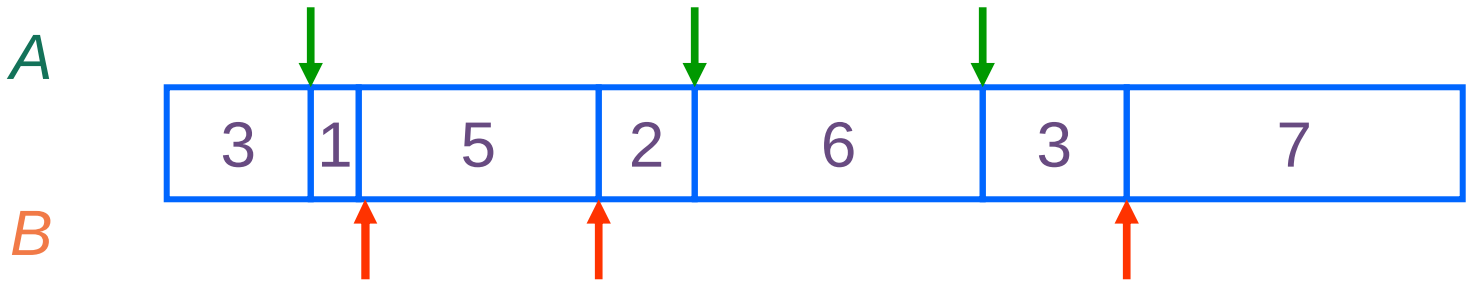


RESTRICTION MAPPING

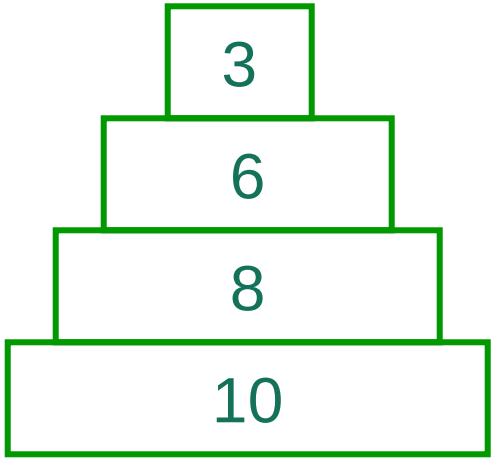
- double digest problem
- partial digest problem

(pictures only)

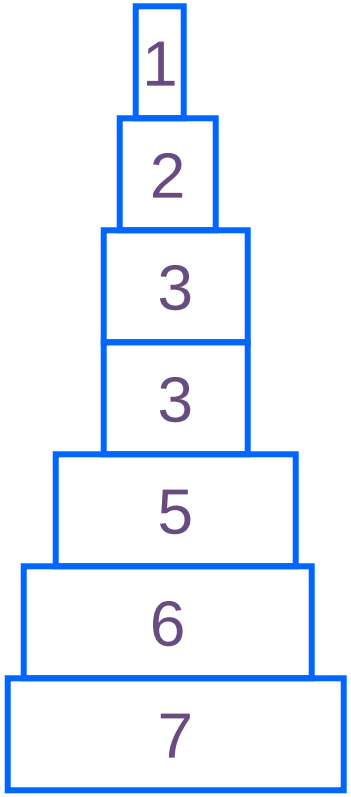
double digest problem



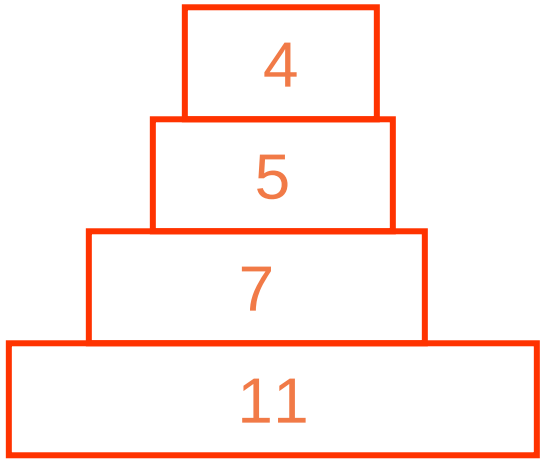
long segments:
unknown sequences



enzyme A {3,6,8,10}

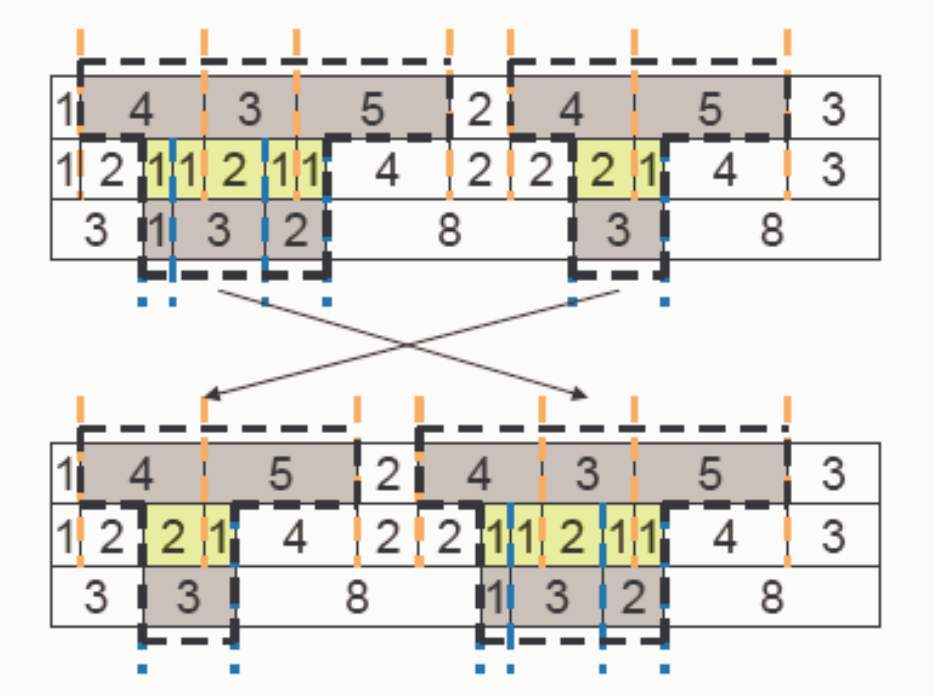


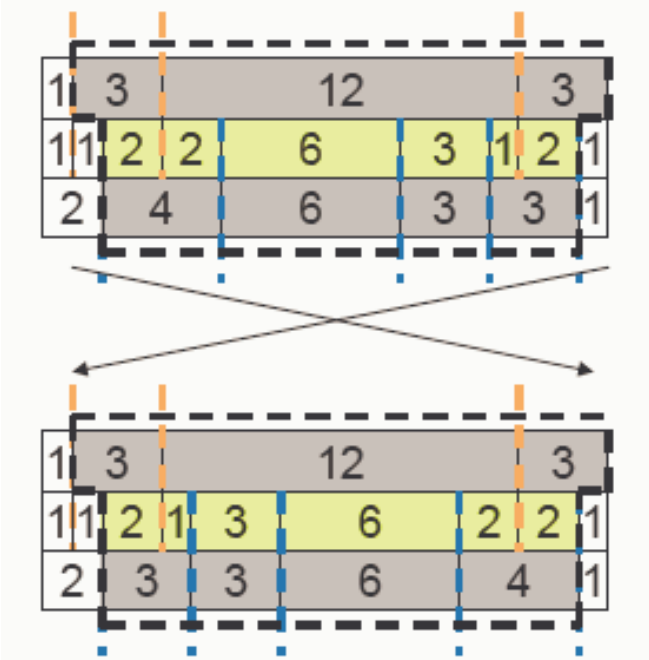
A+B {1,2,3,3,6,7}



enzyme B {4,5,7,11}

cassette exchange / reflection





solution not unique
characterization: interdependence of solutions

reduction from set partition

proving NP completeness (decision version)

- X = { 1, 3, 5, 6, 9 }
- S = 24

A = X

B = { 12, 12 }

A+B = X

set partition (two parts)

restriction

9	3	5	6	1
12		12		
9	3	5	6	1

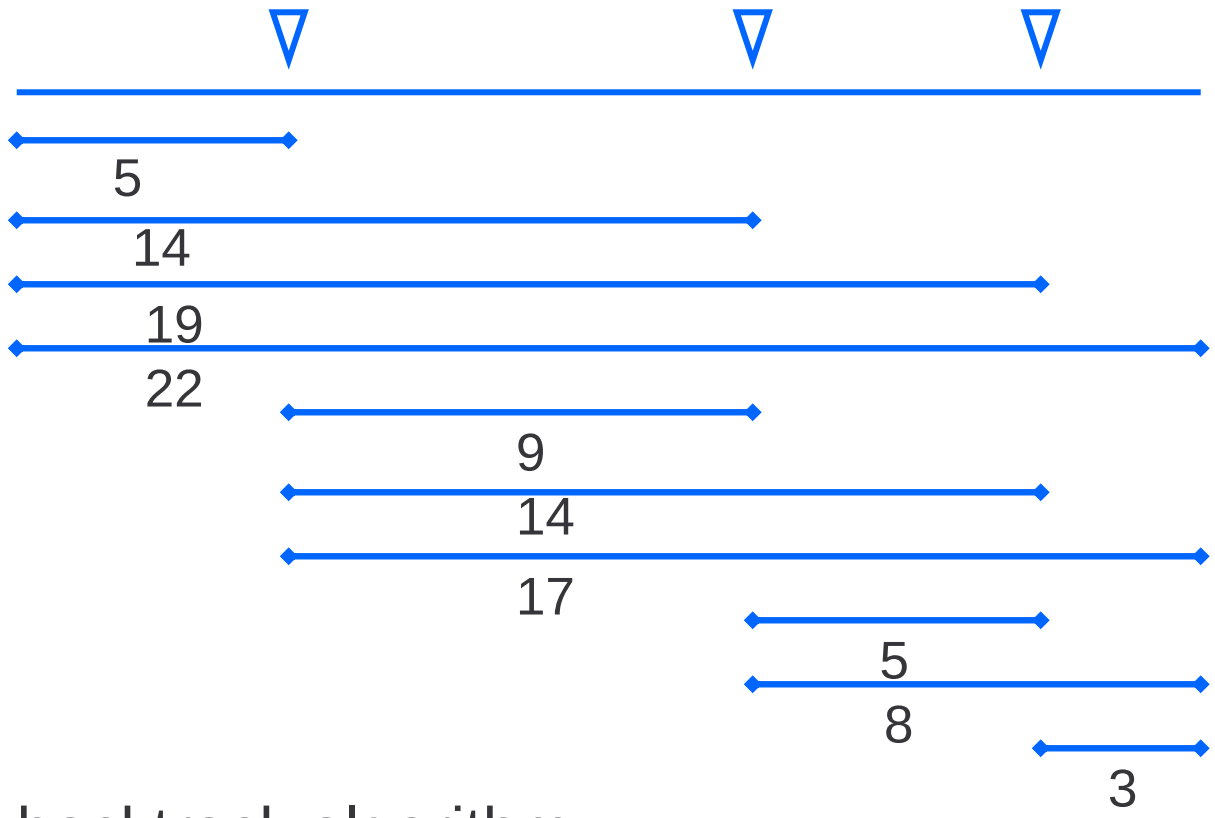
is there a partition ?

is there a restriction ?

partial digest problem

varying duration restriction experiments

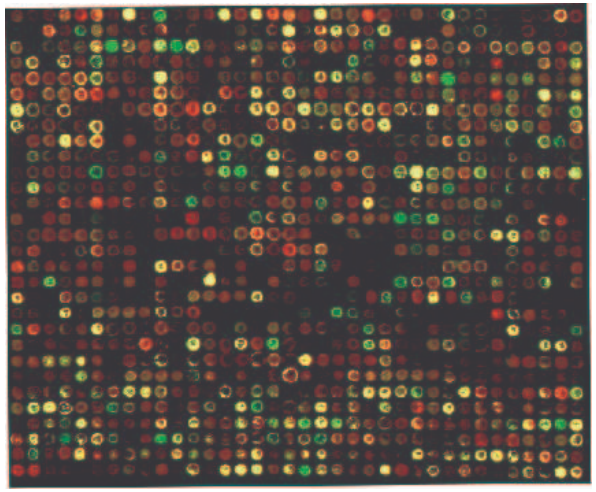
(multi)-set { 3, 5, 5, 8, 9, 14, 14, 17, 19, 22 }

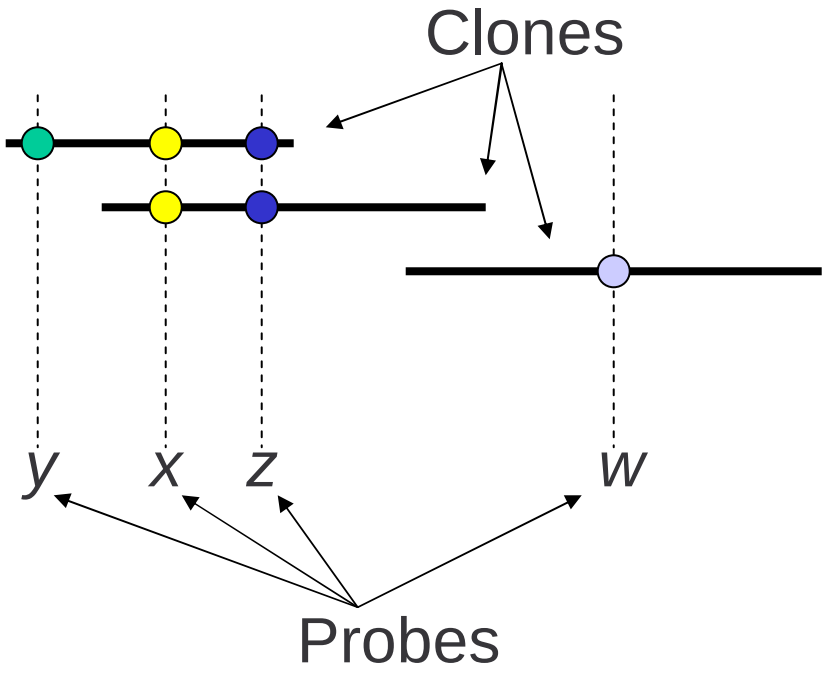


backtrack algorithm

worst case exponential time

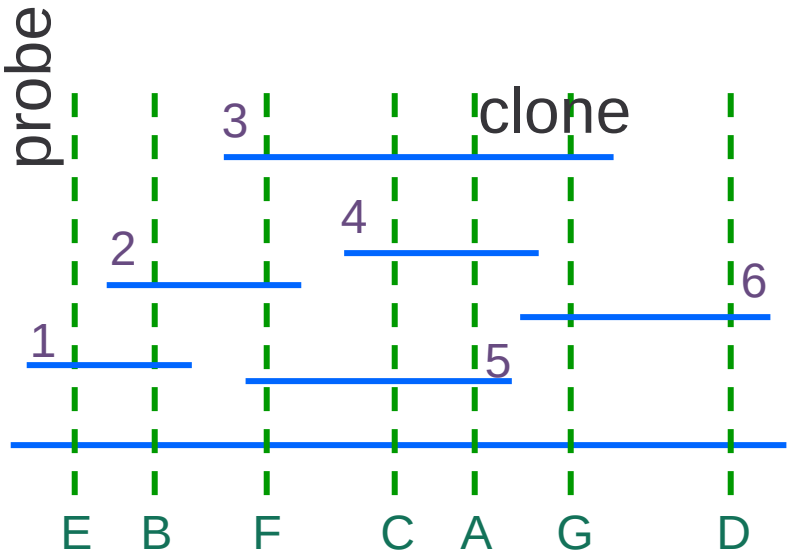
HYBRIDIZATION MAPPING





here: each probe *unique position* on genome

unique probe mapping



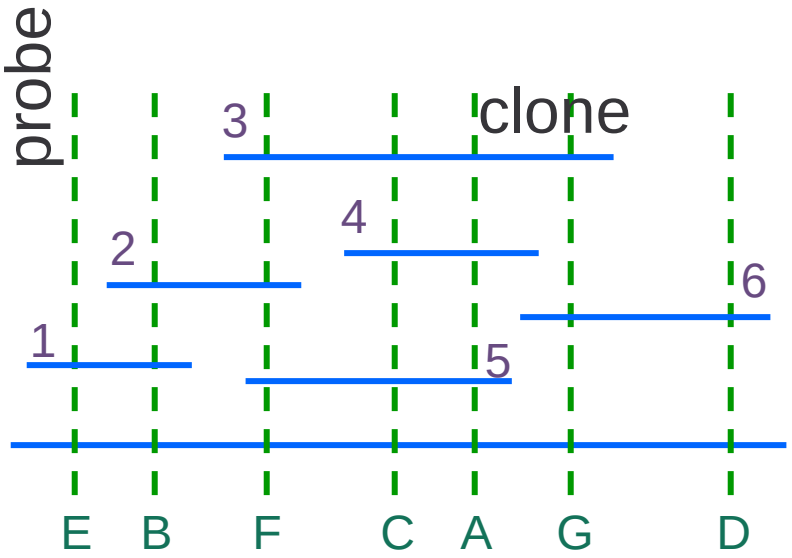
6 clones 1,2,…,6
7 probes A,B,…,G

1 : { B, E }
2 : { B, F }
3 : { A, C, F, G }
4 : { A, C }
5 : { A, C, F }
6 : { D, G }

matrix representation

	A	B	C	D	E	F	G
1		1			1		
2		1				1	
3	1		1			1	1
4	1		1				
5	1		1			1	
6				1			1

reordering of probes



clones contain consecutive probes

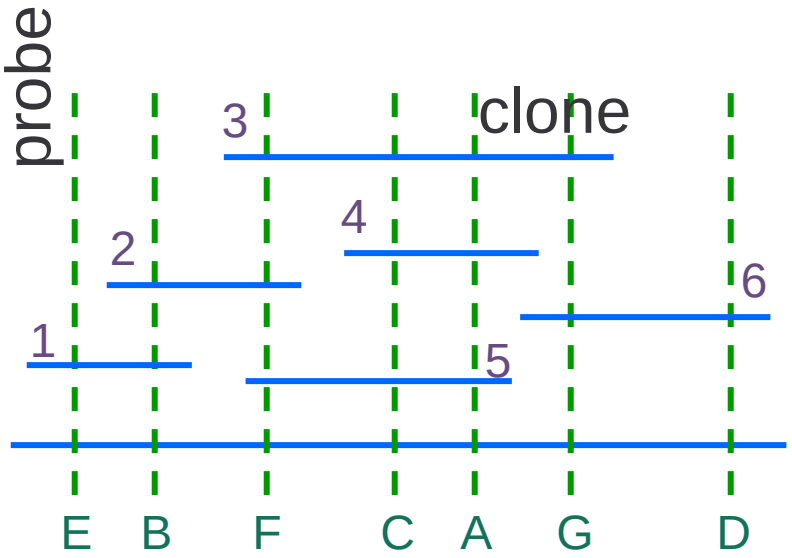
ordering

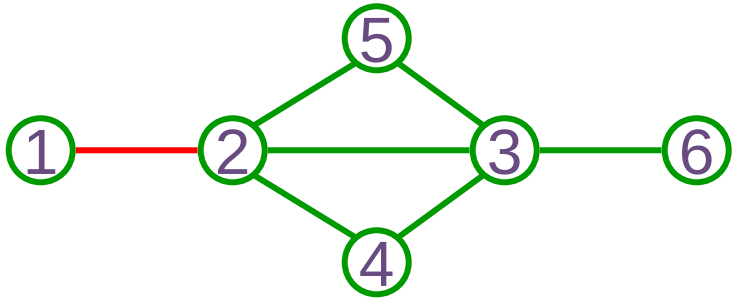
	D	G	C	A	F	B	E
1						1	1
2					1	1	
3		1	1	1	1		
4			1	1			
5			1	1	1		
6	1	1					

	A	B	C	D	E	F	G
1		1			1		
2		1				1	
3	1		1			1	1
4	1		1				
5	1		1			1	
6				1			1

interval graphs

no details in this course!





characterization using cliques

{1,2} {2,3,4} {2,3,5} {3,6}

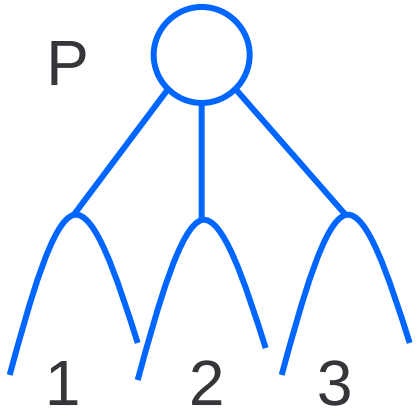
	A	B	C	D	E	F	G
1		1			1		
2		1				1	
3	1		1			1	1
4	1		1				
5	1		1			1	
6				1			1

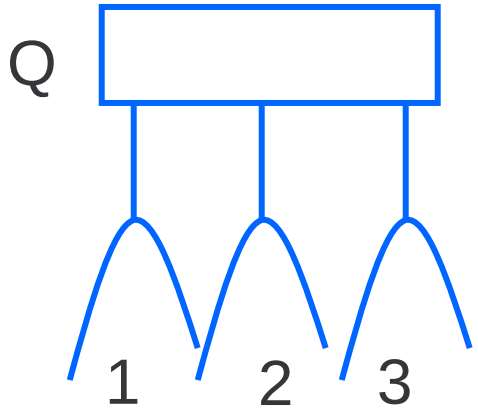
PQ-trees

our focus!

choosing a data structure

representation for permutations



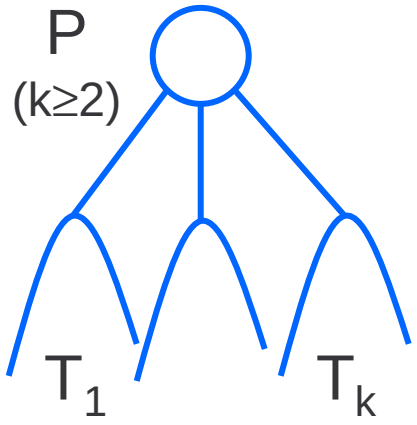


{ 123, 132, 213, 231, 312, 321 }

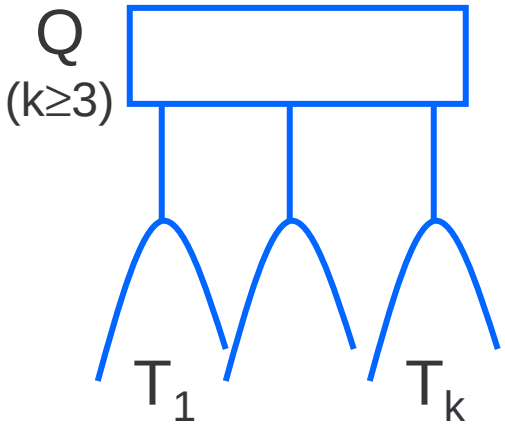
{ 123, 321 }

PQ-trees

data structure to represent all possibilities



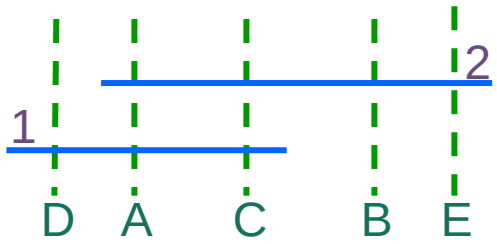
P permutation



Q linear order

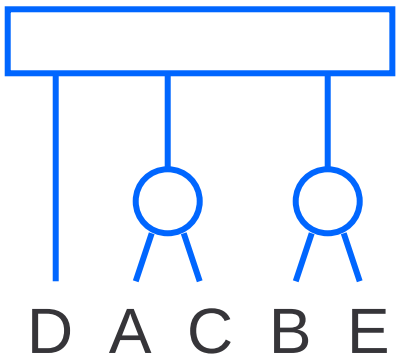
PQ trees
represent possible reorderings
(permutations of probes)

example



	A	B	C	D	E
1	1		1	1	
2	1	1	1		1

clones { A, C, D } { A, B, C, E }

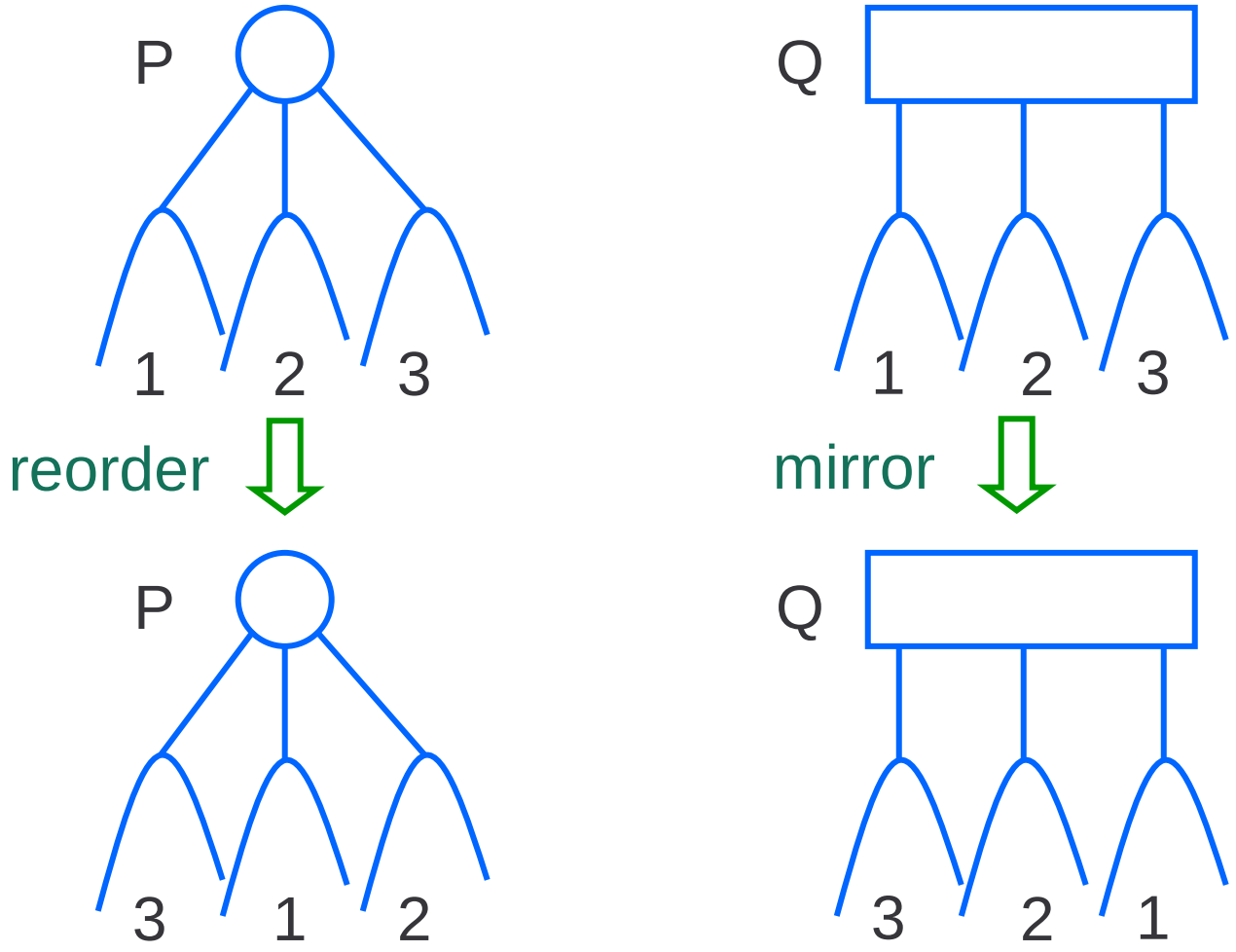


D AC BE
D CA BE
D AC EB
D CA EB

EB CA D
EB AC D
BE CA D
BE AC D

PQ-trees

equivalent representations

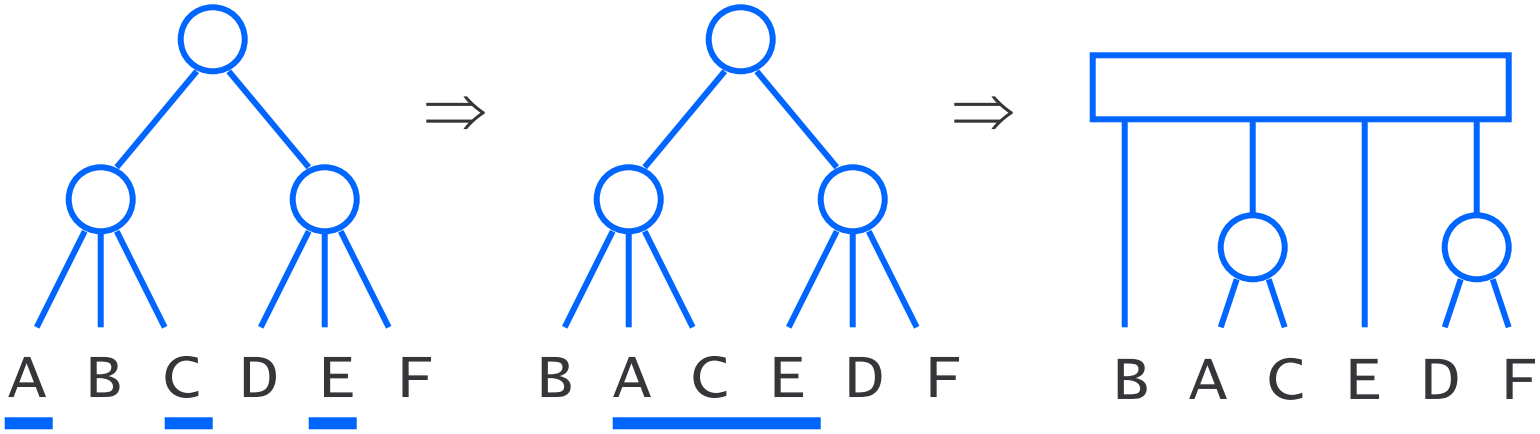


{ 123, 132, 213, 231, 312, 321 }

{ 123, 321 }

example

{A,B,C}
{D,E,F}



S = {A,C,E}

PQ-tree algorithm

reduce(T,S)

T PQ tree ~ set of permutations

S new clone ~ set of (consecutive) probes

add requirement S to tree T

'*keep S together*'

- colour leaves in S
- apply transformations

  reorder to get consecutive leaves
- apply replacement rules (*bottom-up*)

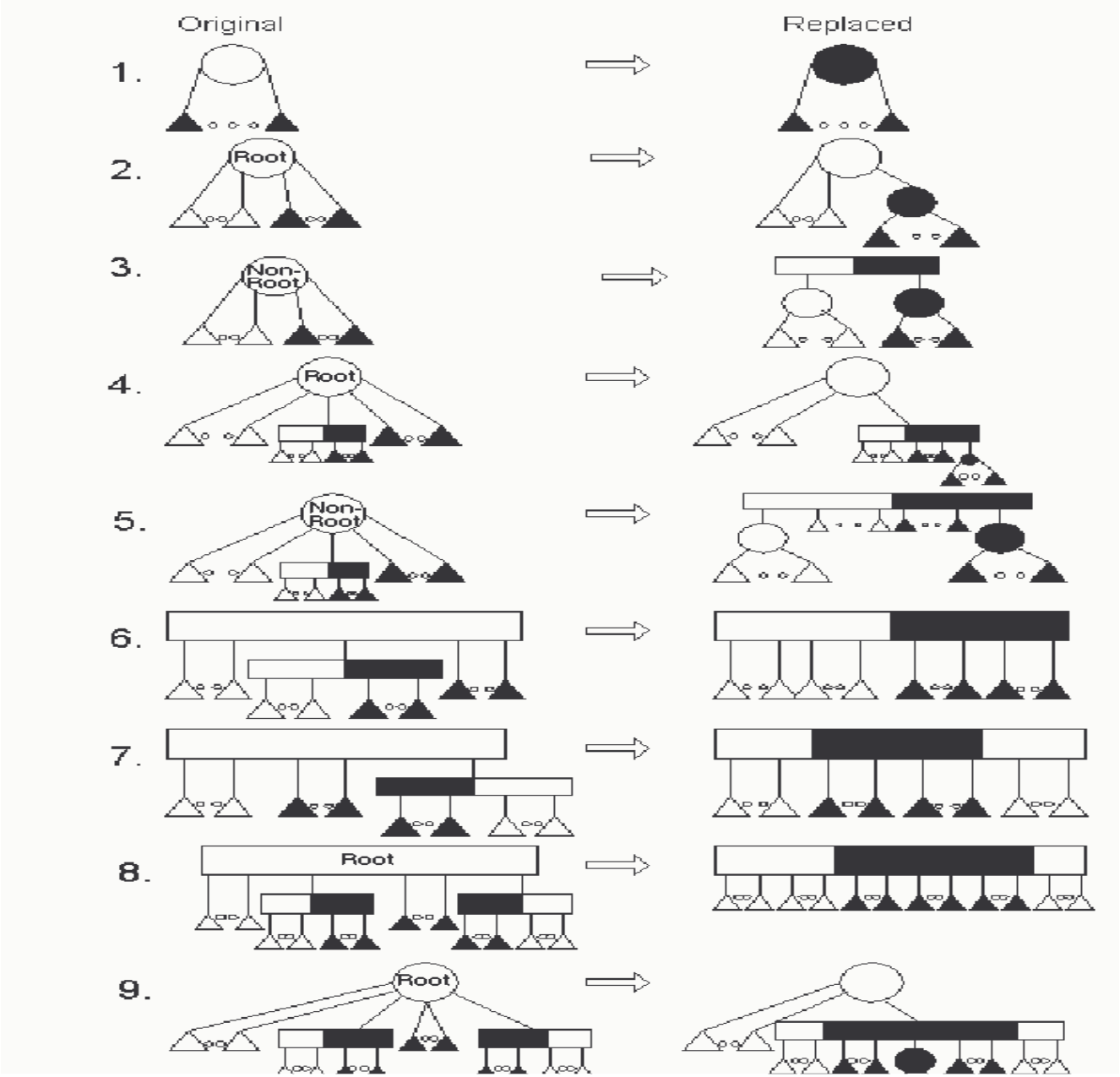
  to add new restriction to tree

P  all leaves in S

Q  segment in S

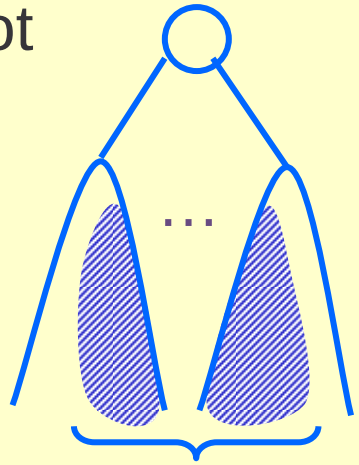
(partially) coloured nodes

replacement rules

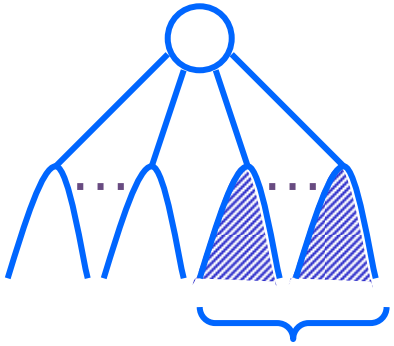


replacement rules (2,3)

root



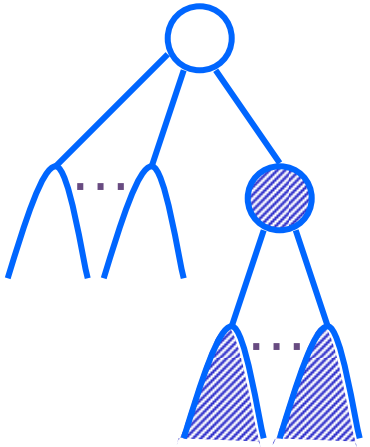
= lowest node having both coloured and non-coloured leaves



root

⇒

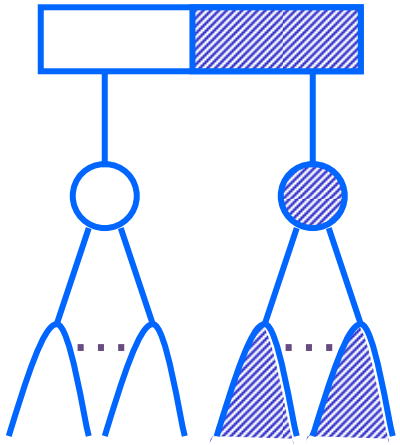
(2)



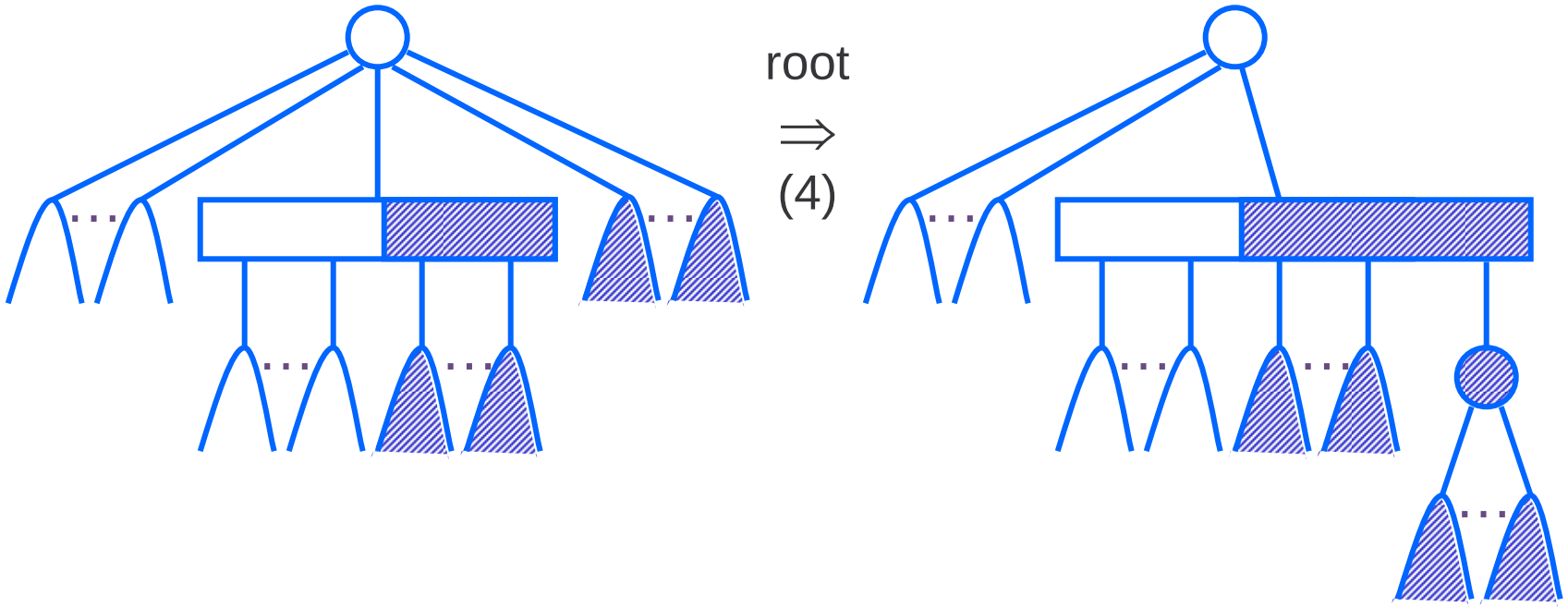
non-root

⇒

(3)



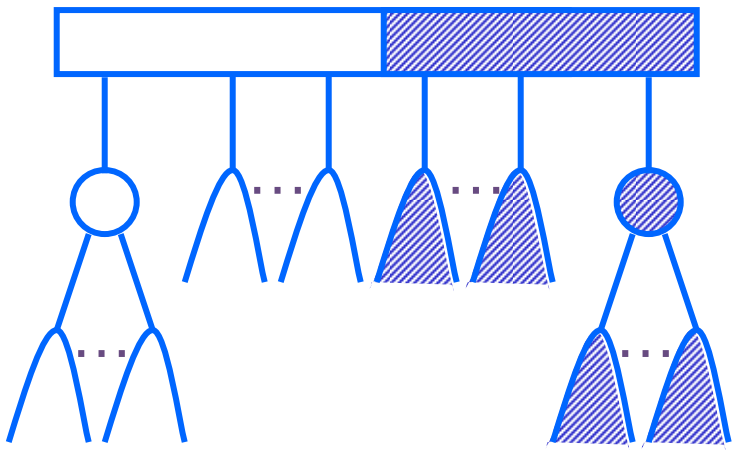
replacement rules (4,5)



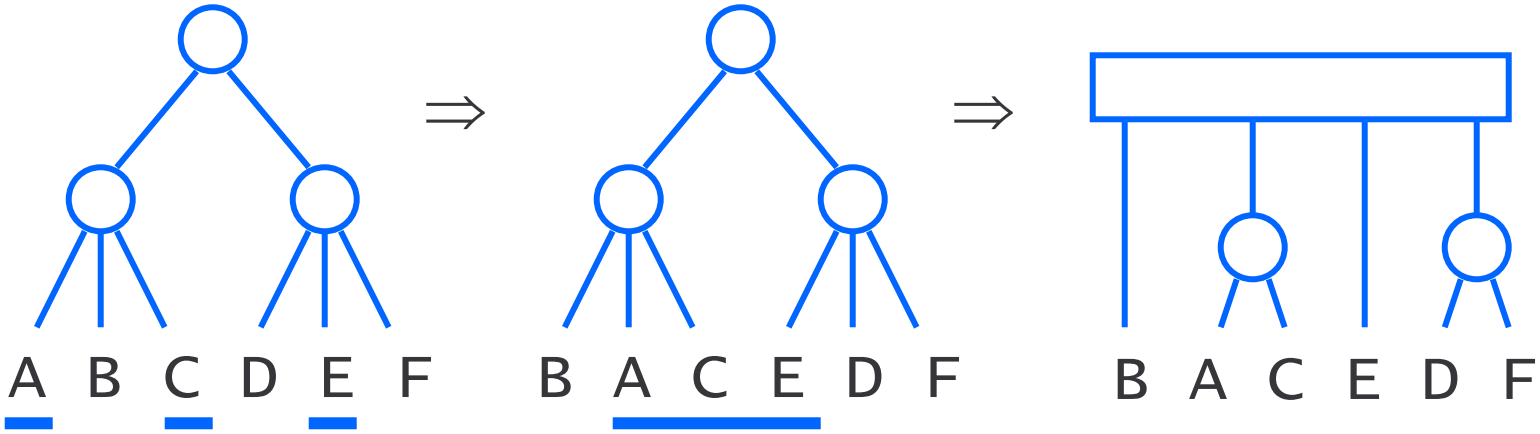
non-root

⇒

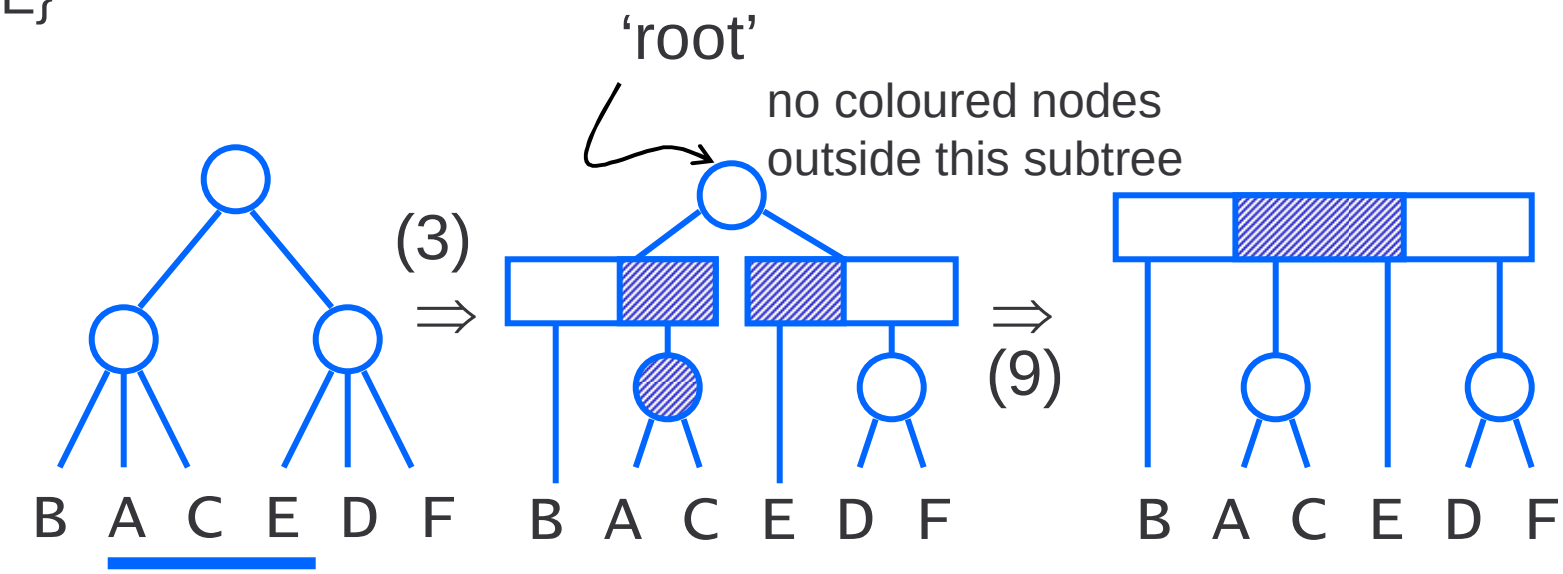
(5)

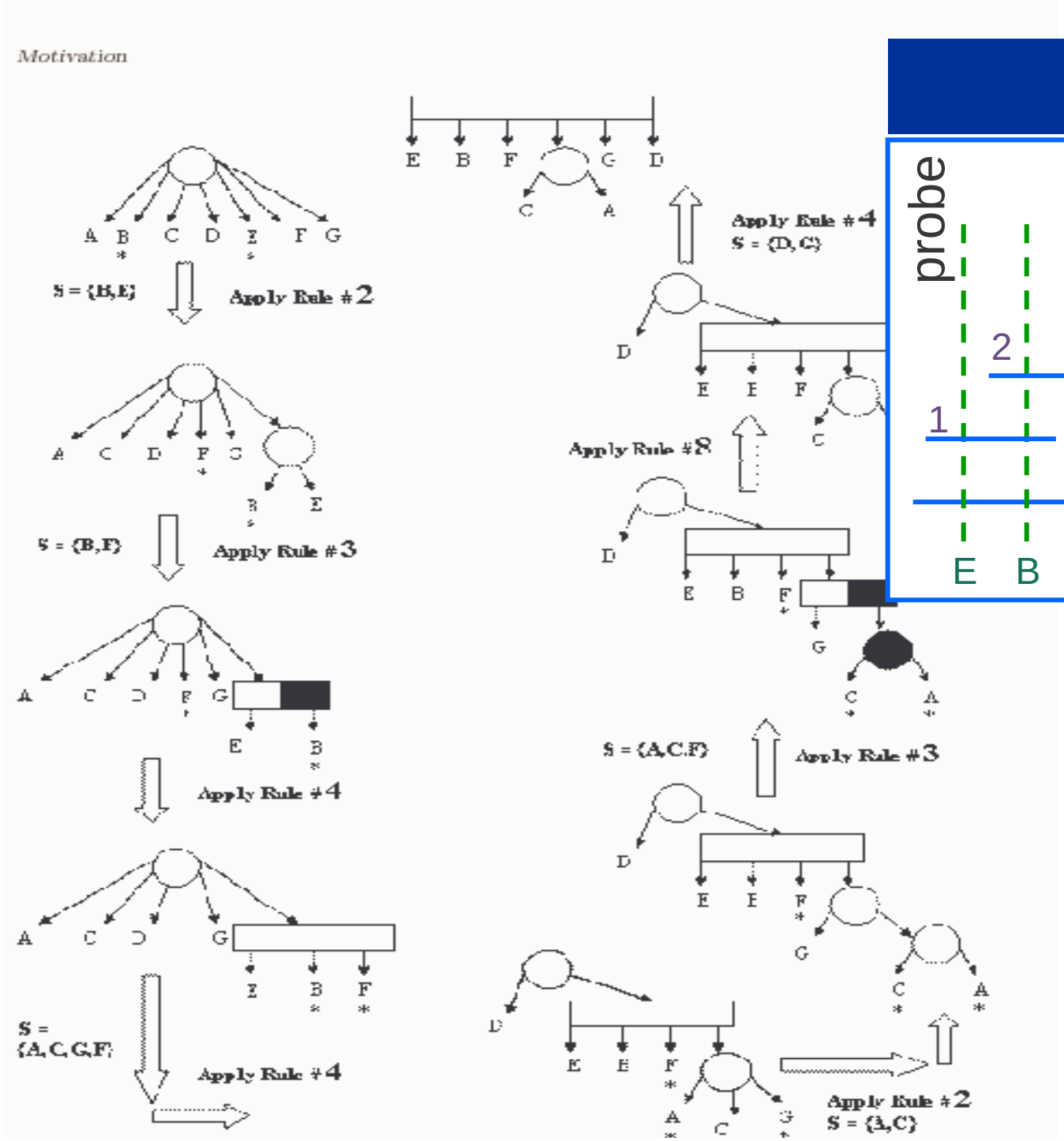


example

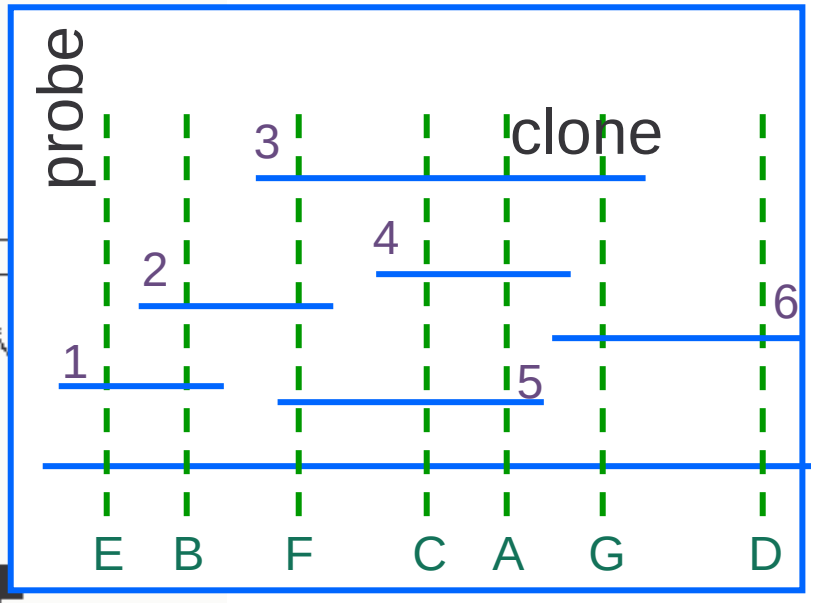


S = {A,C,E}





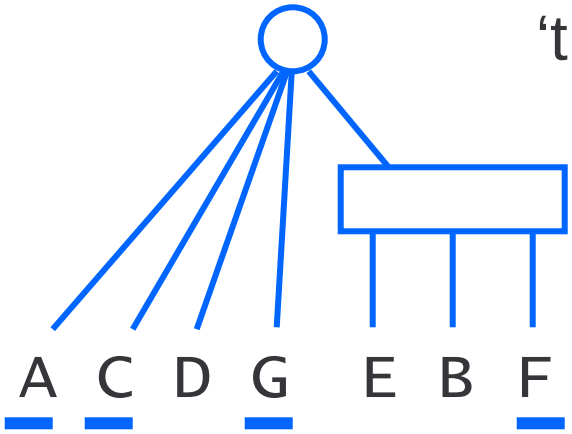
example



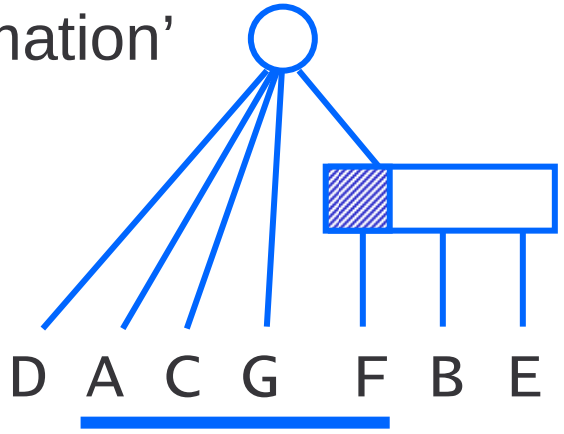
1 : { B, E }
2 : { B, F }
3 : { A, C, F, G }
4 : { A, C }
5 : { A, C, F }
6 : { D, G }

example

1 : { B, E }
2 : { B, F }

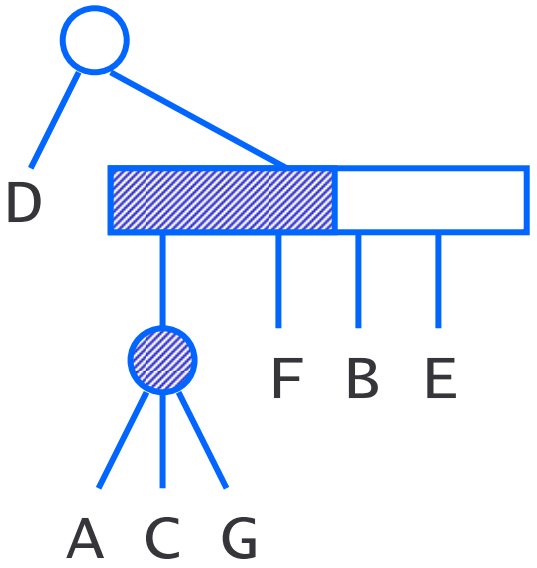


i) reorder
'transformation'

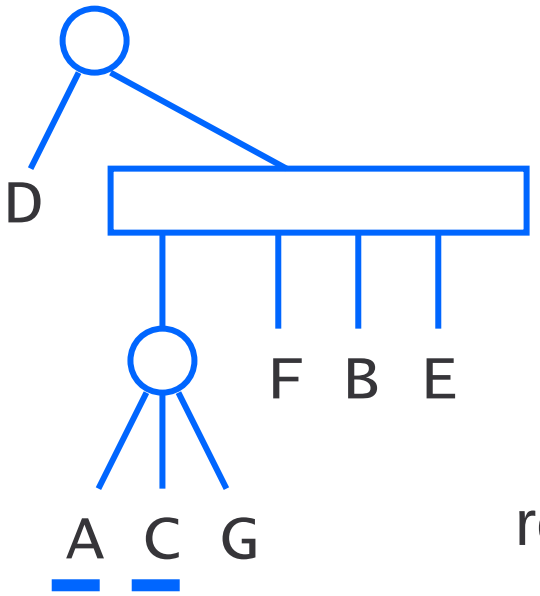


3 : { A, C, F, G }

ii) replacement rule
(4)

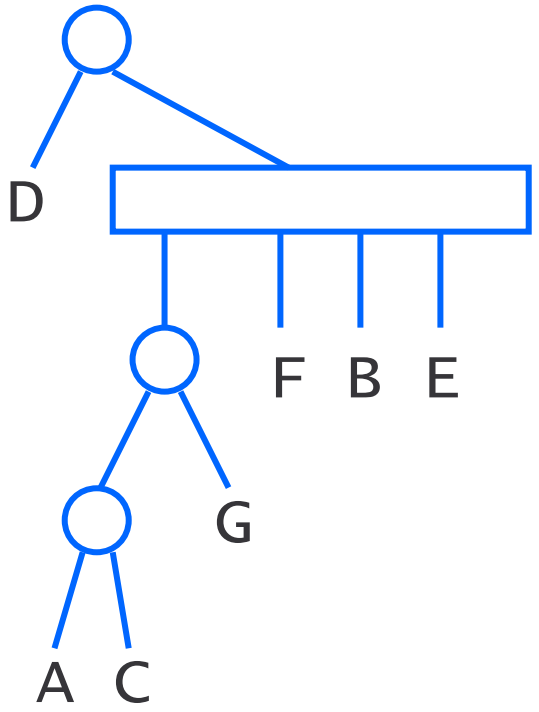


example

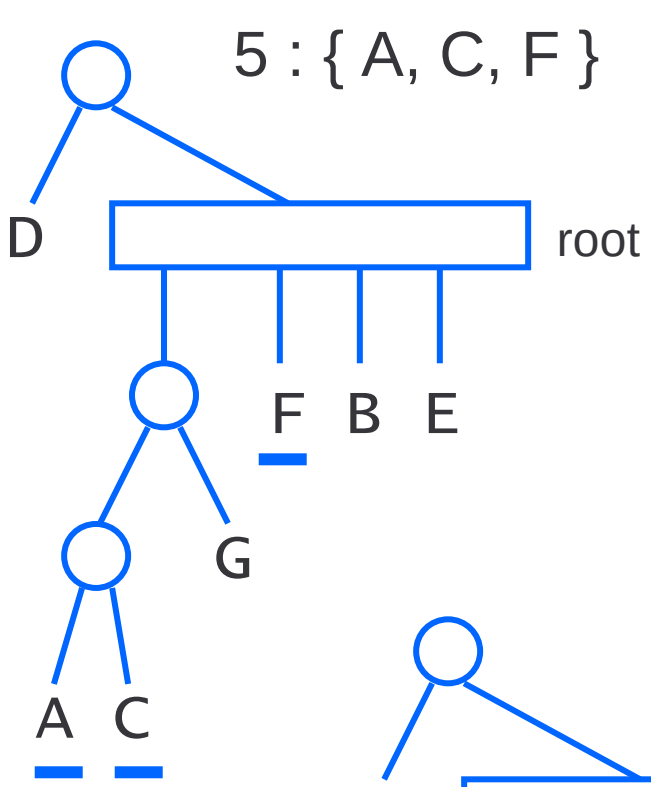


4 : { A, C }

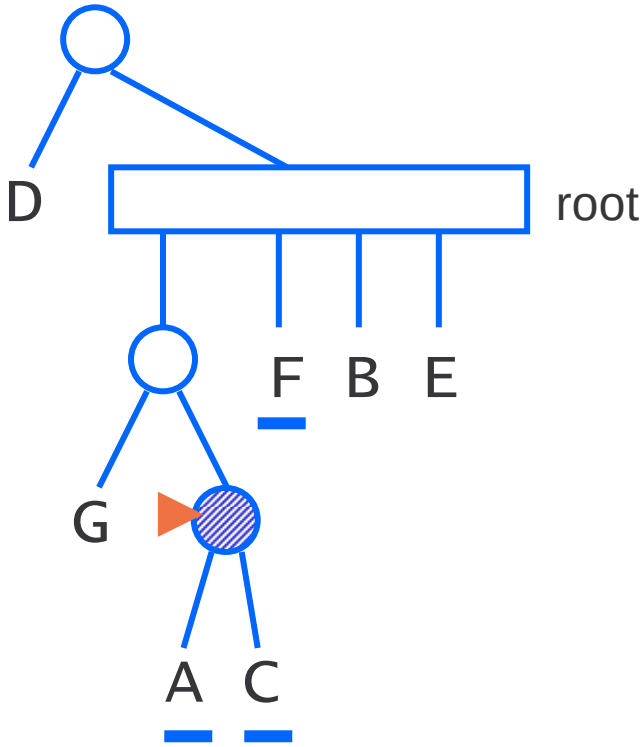
replacement rule
(2)



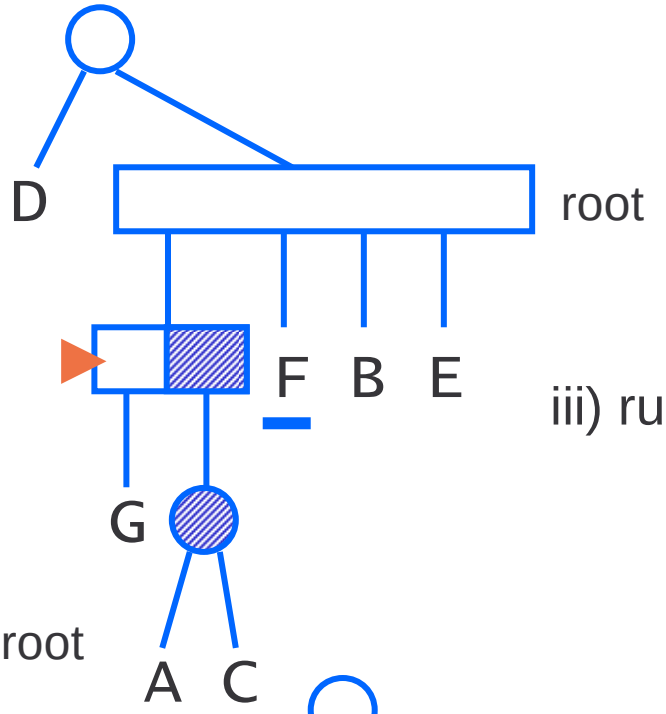
example



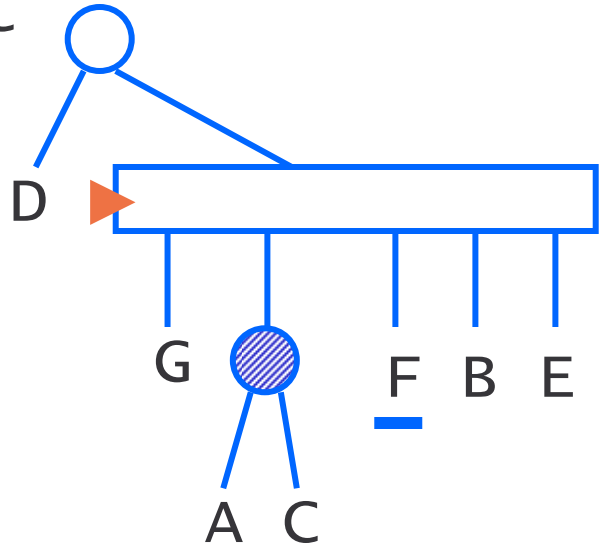
i) reorder & rule (1)



ii) rule (3)

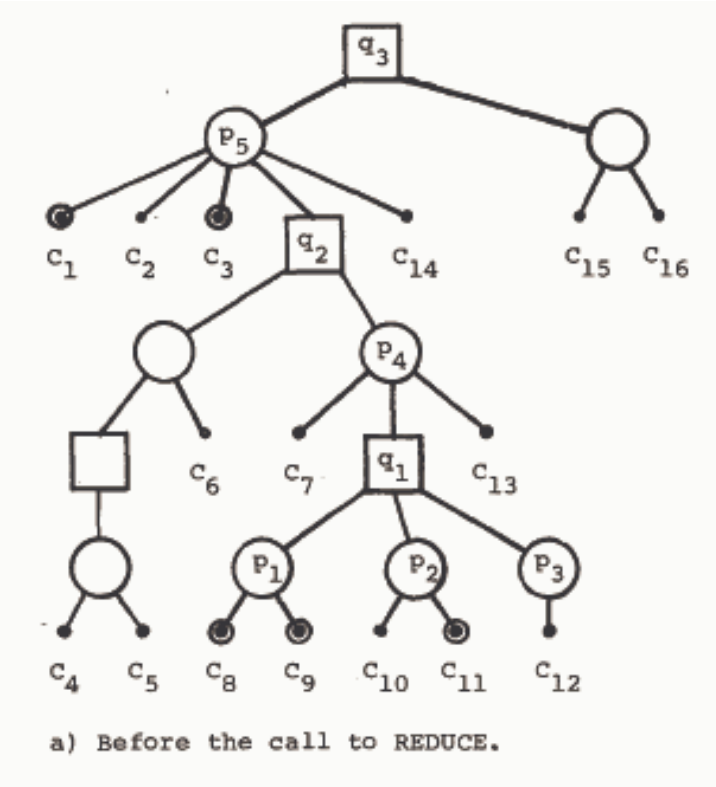


iii) rule (8)



original reference

K.S. Booth and G.S. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. JCSS 13:335-379, 1976.
also 7th STOC, 1975.



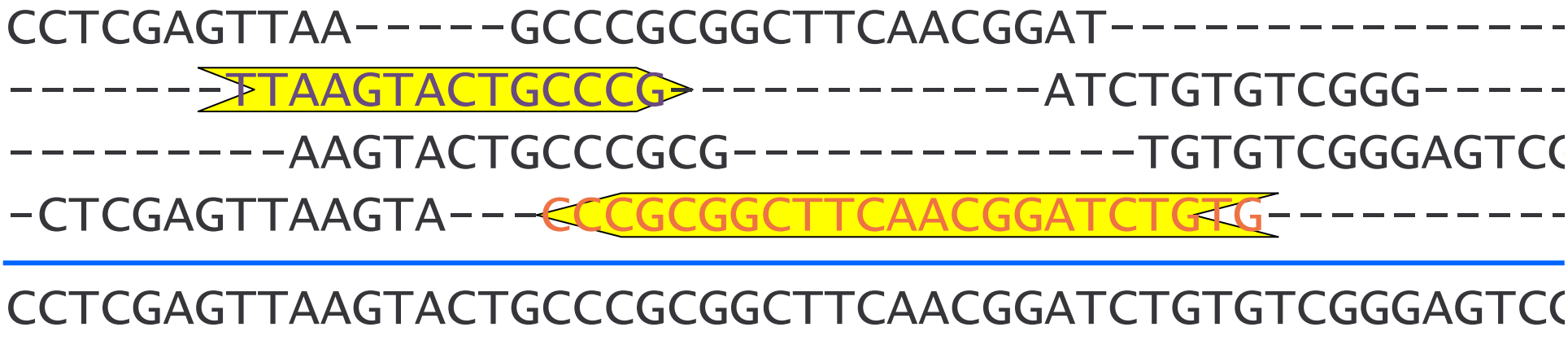
a) Before the call to REDUCE.

FRAGMENT ASSEMBLY

- shortest superstring
- sequencing by hybridization

example

f1 = ATCCGTTGAAGCCGCGGGC
f2 = TTAACTCGAGG
f3 = TTAAGTACTGCCCG
f4 = ATCTGTGTCGGG
f5 = CGACTCCCGACACA
f6 = CACAGATCCGTTGAAGCCGCGGG
f7 = CTCGAGTTAAGTA
f8 = CGCGGGCAGTACTT



model: shortest common superstring

shortest common superstring

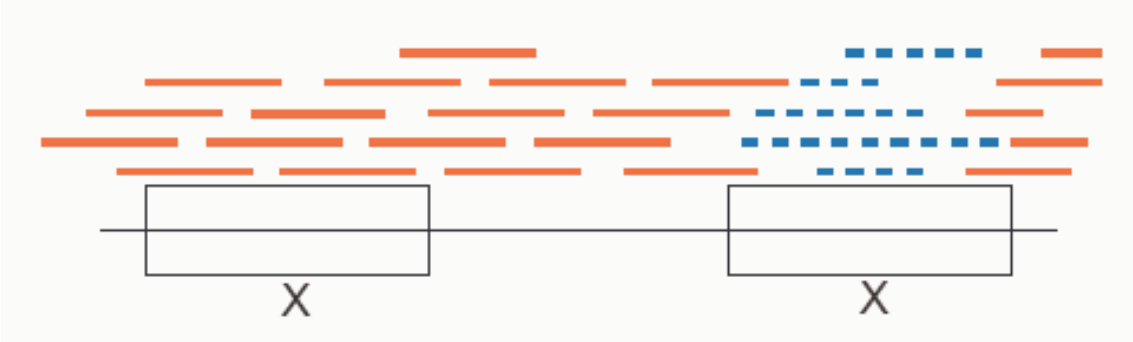
given a set of fragments F,

find the shortest string s that contains every f ∈ F as a substring

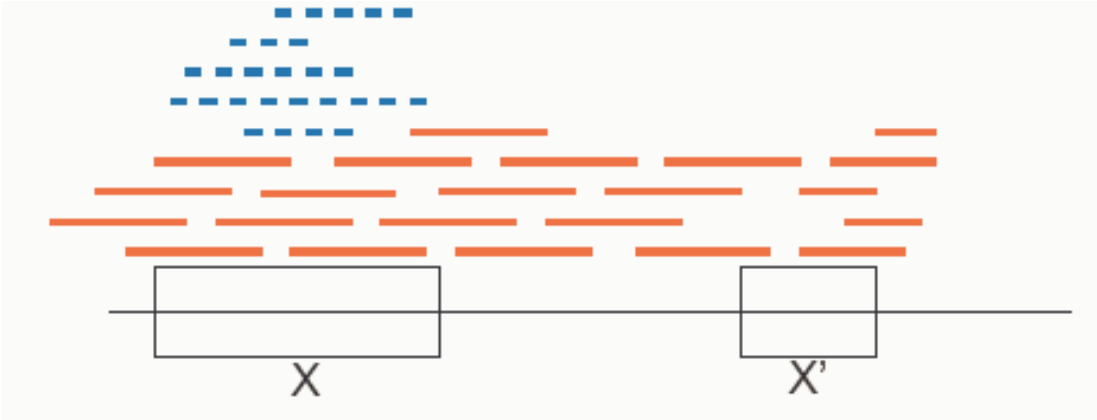
- is NP-hard
- is perhaps not what we want

“ An elegant theoretical abstraction, but fundamentally flawed ” – R. Karp

repeats ☹

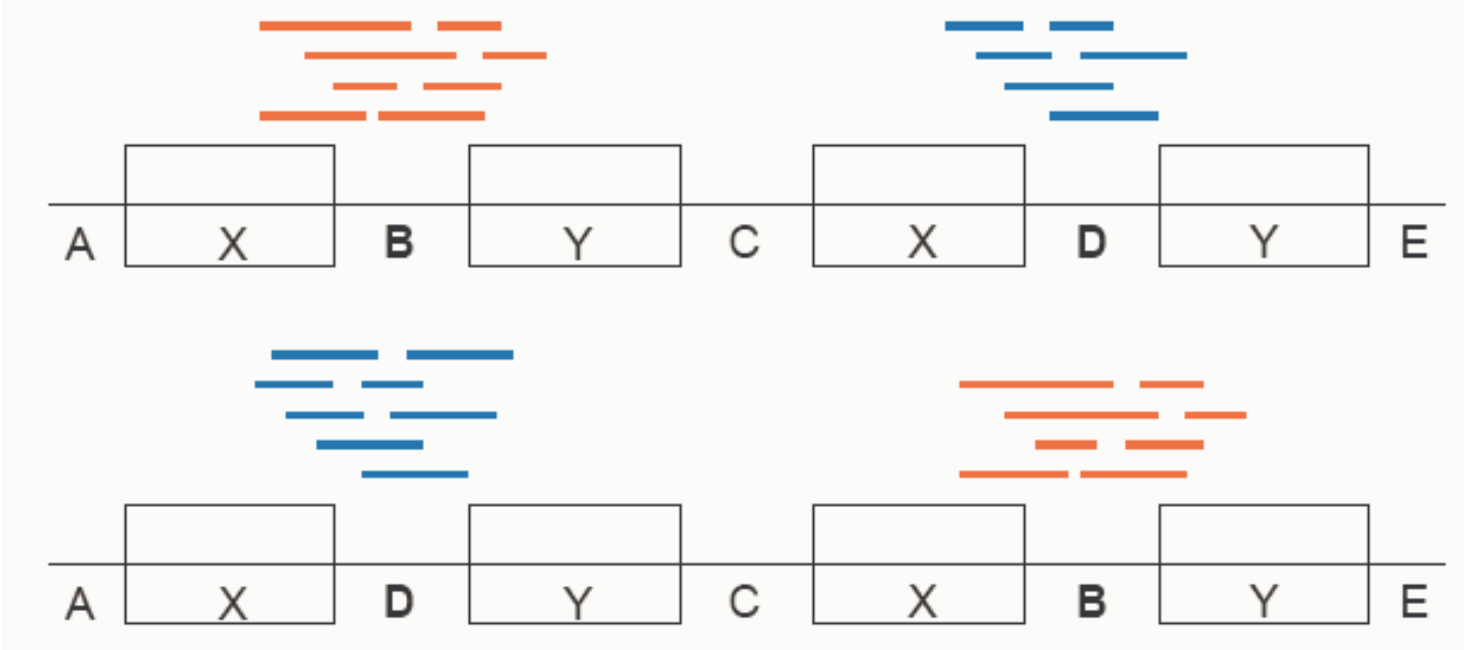


shortest common superstring



but: covering is not ‘uniform’

repeats ☹



aXbYcXdYe ⇒ aXdYcXbYe

also: aXbXcXd ⇒ aXcXbXd

base errors ☹

experimental

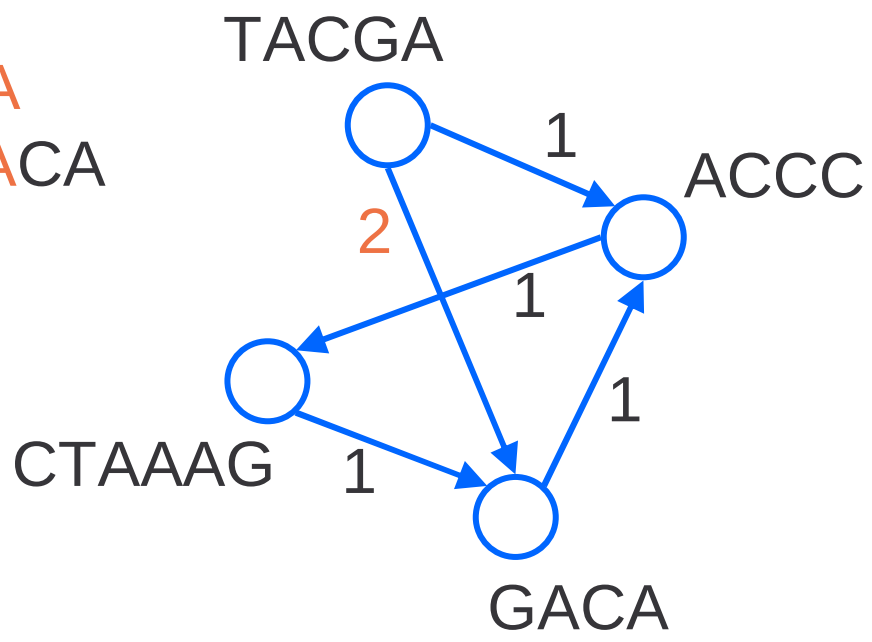
substitutions / insertions / deletions

chimeras

direction of strings …

tool: overlap graph

TACGA
GACA



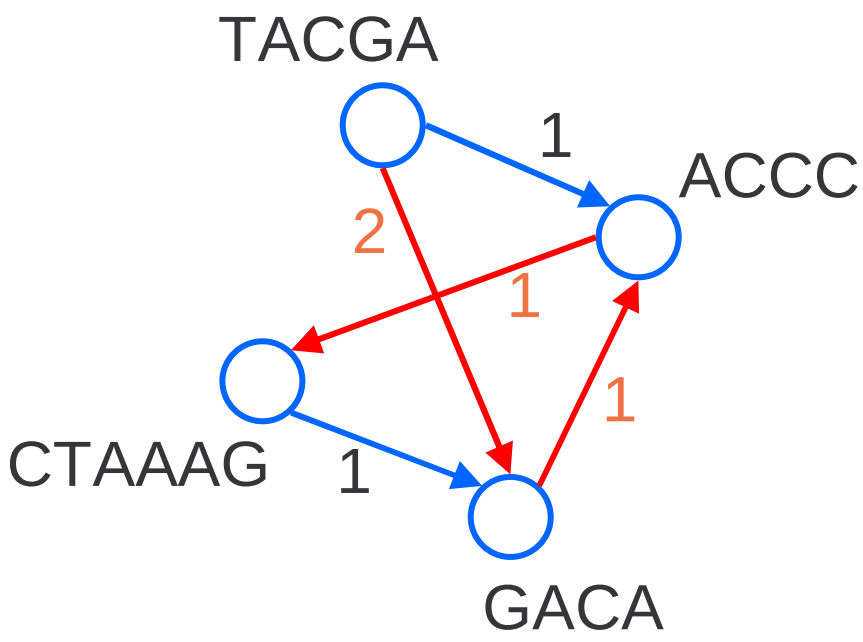
{ ACCC,
CTAAAG,
GACA,
TACGA }

assumption:
no substrings (inclusion)

omit zero weight edges

compute overlaps : suffix tree [exact & fast]
or alignment ! [error proof]

overlap graph: Hamilton path



Hamilton ~ visit every node (exactly) once

TACGA
GACA
ACCC
CTAAAG

TACGACACCCTAAAG

length ‘superstring’ =
total length strings – length path

look for *longest* Hamilton path
☹ NP complete ⇒ heuristics

overlap graph: greedy algorithm

```
ATGC
  TGCAT
        GCC
ATGCATGCC
```

greedy

simple heuristic:
join strings with maximal overlap

approximation within factor ??
conjecture: factor 2 of optimal
(proofs for 4, 2.75 …)

```
   ATGC
TGCAT
      GCC
TGCATGCC
```

optimal

+additional heuristics

general ‘bad’ example:

	$C(AT)^k$ $(TA)^k$ $(AT)^kG$	
greedy	$C(AT)^kG(TA)^k$	4k+2
best	$C(AT)^{k+1}G$	2k+4

overlap graph: problems

AGTATTGGCAATC AATCGATG

ATGCAAACCT

TTGGCAATCACT CCTTTTGG

AGTATTGGCAATCACTAATCGATGCAAACCTTTTGG

length 36
weak link 0
'greedy'

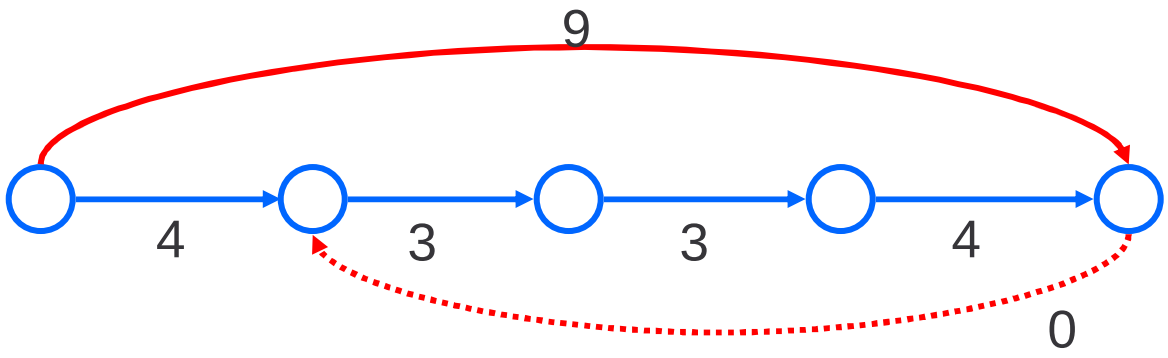
AGTATTGGCAATC CCTTTTGG

AATCGATG TTGGCAATCACT

ATGCAAACCT

AGTATTGGCAATCGATGCAAACCTTTTGGCAATCACT

length 37
weak link 3
'topological sorting'



ideal world

consensus

probabilistic models

- how much of the genome is covered?

$E(\text{not covered}) = e^{-R}$

$R = N \cdot L / G$ *redundancy*

L clone length

N number of clones

G genome length

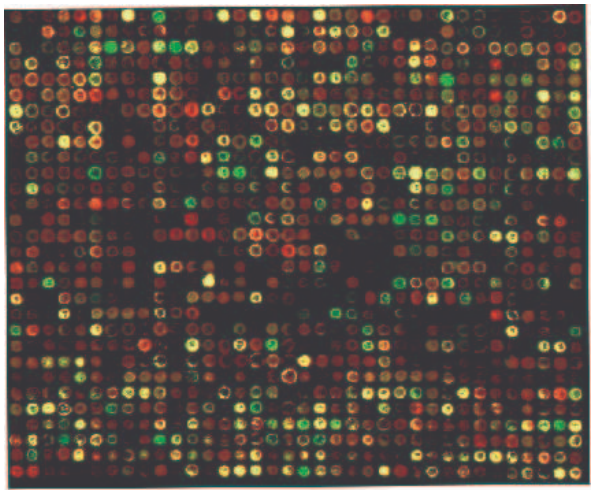
- probability of islands (contig's)

expected number of islands $Ne^{-R(1-\theta)}$

θ *overlap factor*

sequencing by hybridization

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



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

$\ell = 3$

ATTGAC

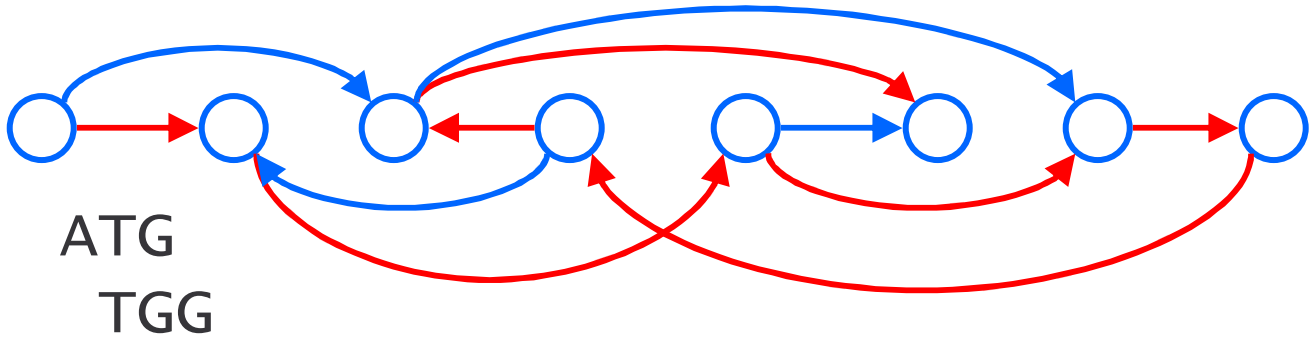
SBH example

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

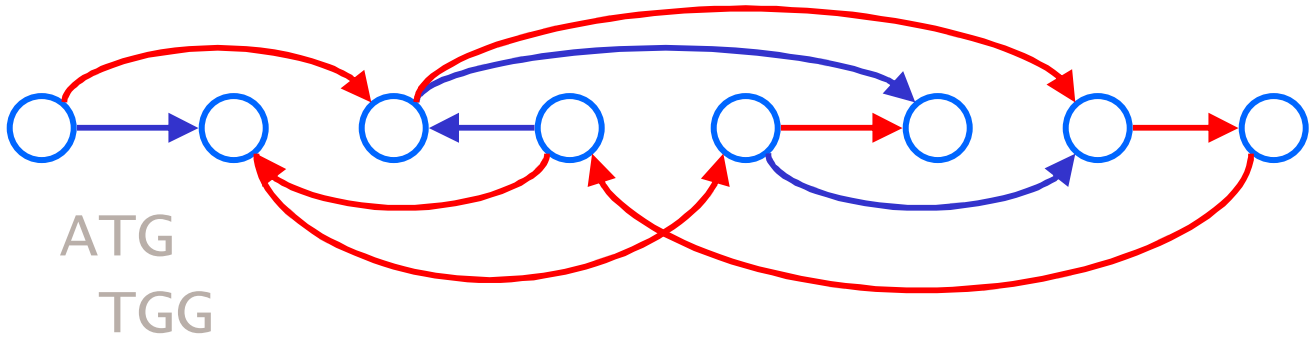
SBH example

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)

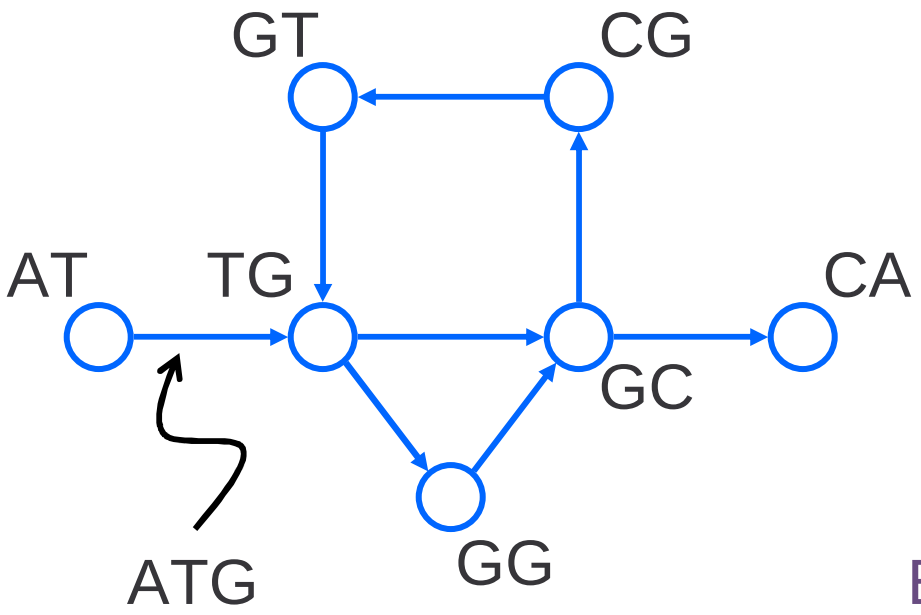
triplet=node

SBH example

we can do better with same problem:

$\ell = 3$

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



ATGGCGTGCA

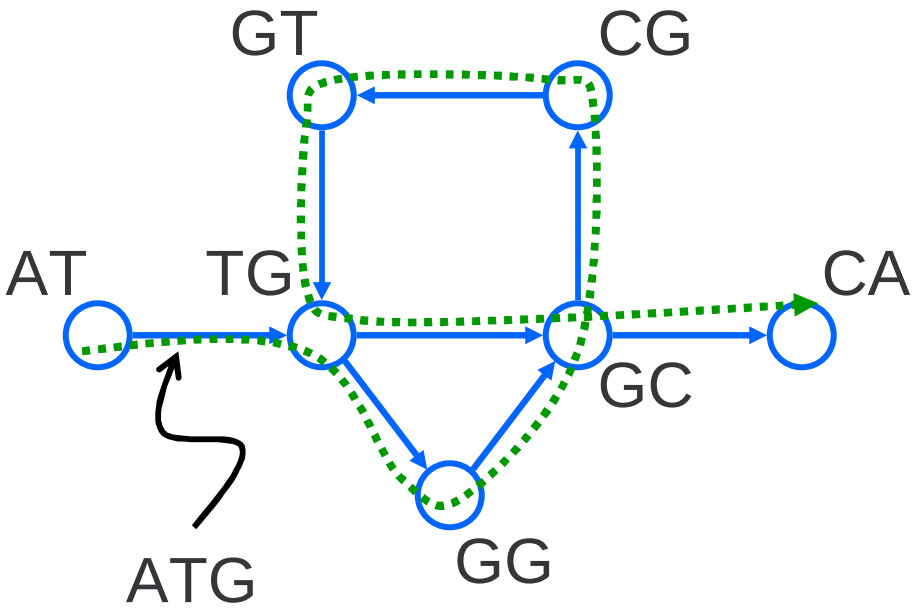
Euler approach: edges
(overlap ℓ-1 = node)
linear ☺

triplet=edge

SBH example

$\ell = 3$

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



ATGGCGTGCA
ATGCGTGGCA

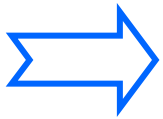
Euler approach: edges

even degree nodes
(except start+finish)

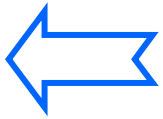
scheme

real world

model : ‘abstraction’



is this what we want?
(can we handle errors?)



algorithm
- NP complete : heuristics

characterization
- how solutions relate

this slide intentionally left blank