Haplotype Assembly

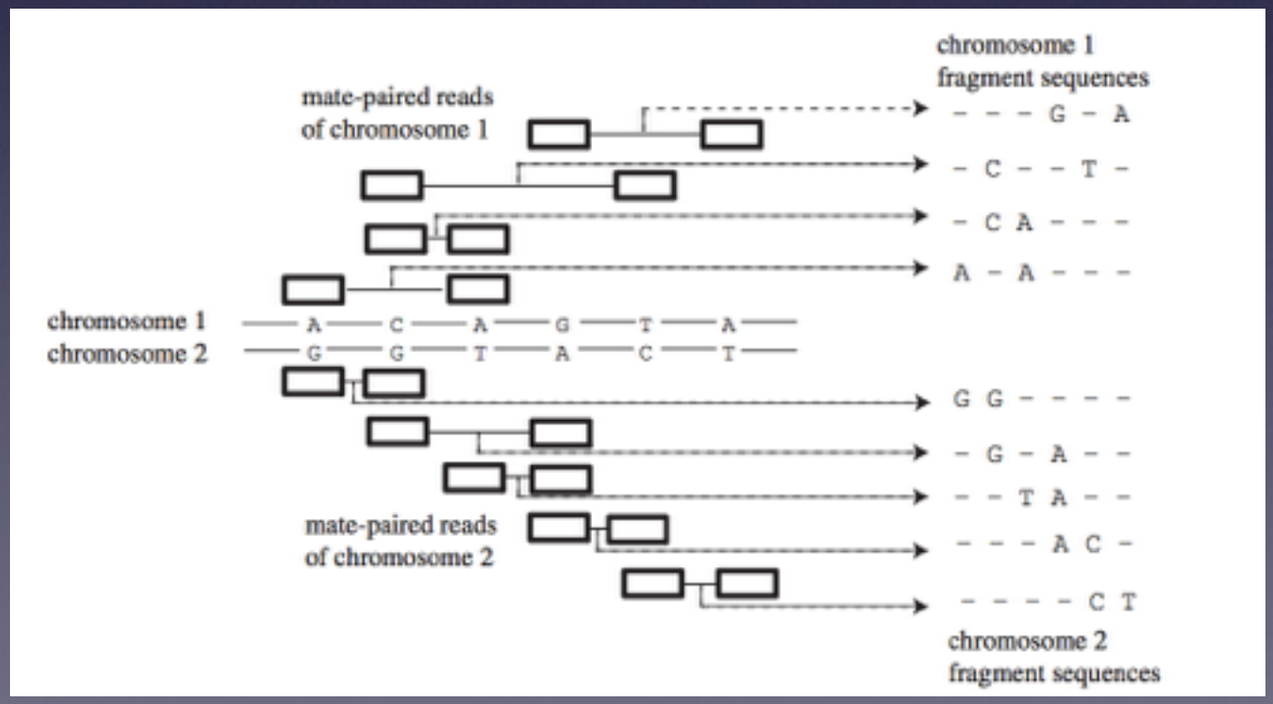
Presented by: Qinyi Yan (704406413)

Problem Motivation and Statement

- Haplotype inference is an important step for many types of analyses of genetic variation in the human genome

- The development of high-throughput sequencing technologies allows for a strategy to obtain haplotypes by combining sequence fragments.

- The problem of 'haplotype assembly' is the problem of assembling the two haplotypes for a chromosome given the collection of such fragments, or reads, and their locations in the haplotypes.



Computational problem

Original genotype:

ATACGGCTAGATTC

ATGCGGTTAGCTTT

In another form(our way):

01001101011110

0110111101010111

Distinguish heterozygous from homozygous

01001111011110

0110110101010111

Heterozygous genotype

0110

1001

Computational problem (Con't)

After Sequencing:

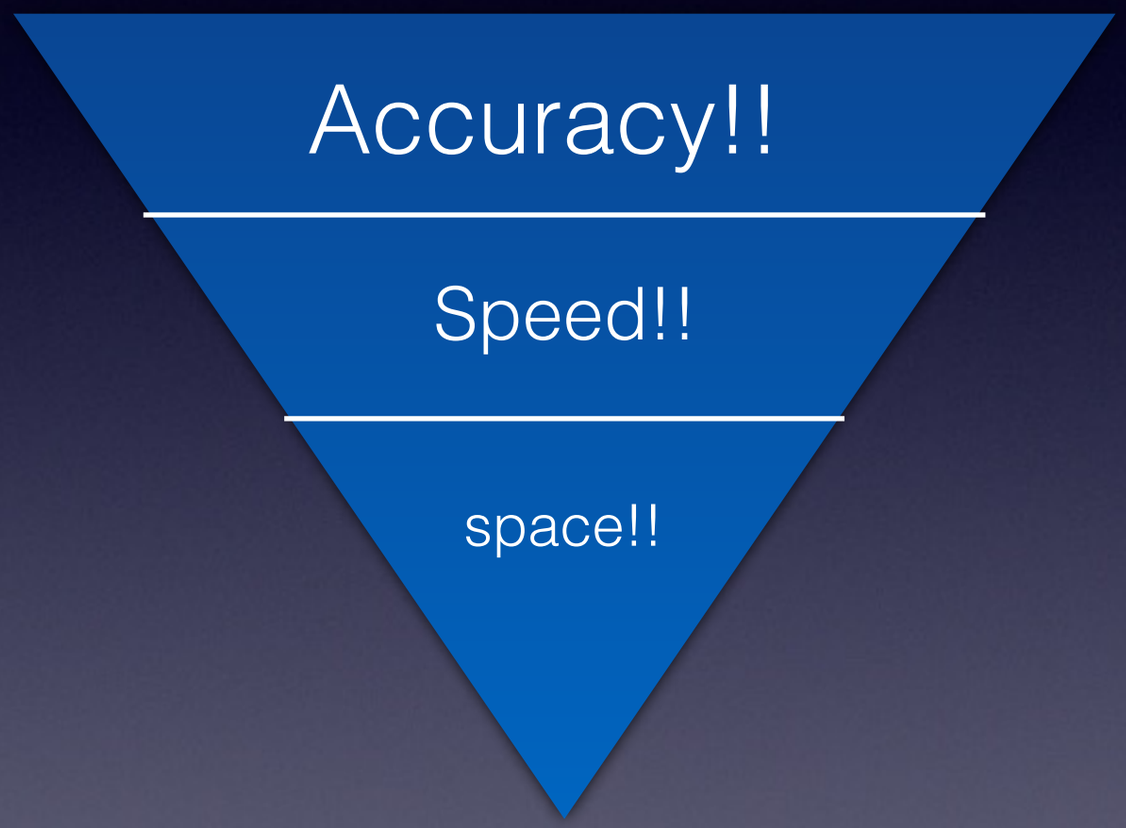
01
11
001
10

Added some errors:

01
11
101
10

Our Goal: reassemble the two haplotypes based on the relatedness of opposite

Benchmarks



Easy problem statement

Know condition:

- no errors:

  No need to counting 0s and 1s. No need to calculate frequency → short time small space

- Has homozygous SNPs:

  Need to extract homozygous SNPs → what's the threshold?

Other observation:

- Has non-overlapping after excluding homozygous SNPs! → 2^n -1 possibilities!

Easy problem algorithm

1st stage: Extract homozygous SNPs

0	0	1	1				
	1	0	0	1			
	0	1	1	1			
			0	1			
			1	1	0	0	
			1	1	1	0	
				1	0	0	…
…	…	…	…	…	…	…	…

2 approaches:

1) read column and calculate the frequency of 0s and 1s
-> threshold =100%
+: easy algorithm
-: takes time

2) found out the “error bits” of each read
+: saves time
-: inaccurate for large amount of homozygous SNPs

Easy problem algorithm

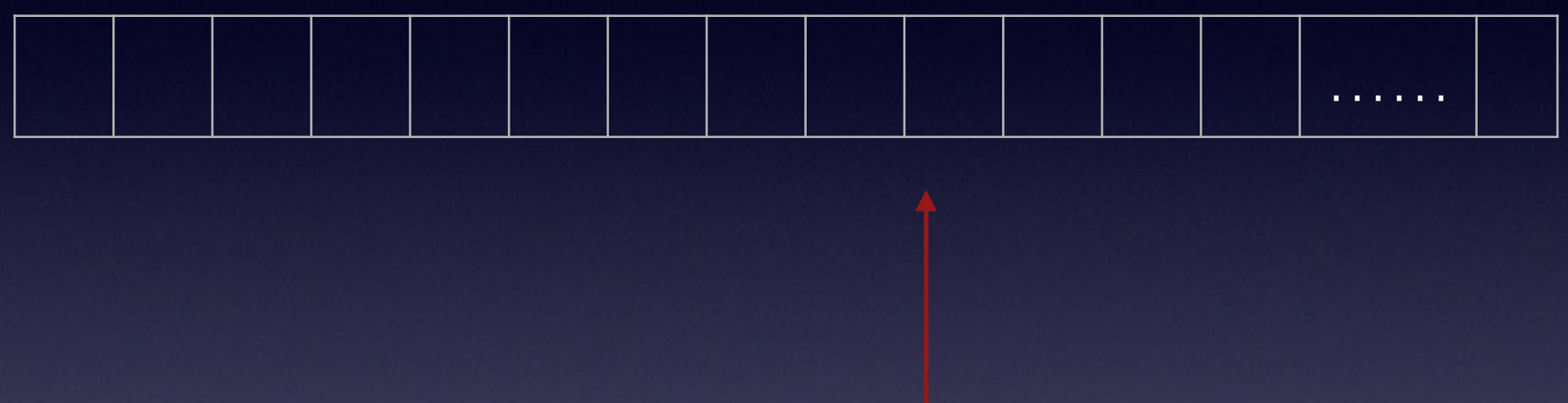
2nd stage:

an n size array (empty)

1st read: 0011——.....

Easy problem algorithm

an n size array (empty)



1st read: 0011——.....

Easy problem algorithm

0	0	1	1										.....	

Scenario 1:

2nd read: - 0110- - - -.....

The overlap positions are the same: plug in the non-overlapped bits

Easy problem algorithm

0	0	1	1	0									.....	

Scenario 1:

2nd read: - 0110- - - -.....

The overlap positions are the same: plug in the non-overlapped bits

Easy problem algorithm

0	0	1	1										.....	

Scenario 2:

2nd read: - 1001- - - -.....

The overlap positions are the same:
flip the non-overlapped bis then plug in

Easy problem algorithm

0	0	1	1	0									.....	

Scenario 2: 0

2nd read: - 1001- - - - -.....

The overlap positions are the same:
flip the non-overlapped bis then plug in

Easy problem algorithm

0	0	1	1										.....	

Scenario 3:

2nd read: - - - -10- - - -.....

Non-overlapped read!

Easy problem algorithm

bit 0	bit 1	bit 2	bit 3
0	0	1	1

Global container for previous haplotype

a new empty array

Scenario 3:

2nd read: - - - -10- - - -.....

Non-overlapped!

Easy problem algorithm

bit 0	bit 1	bit 2	bit 3
0	0	1	1

Global container for previous haplotype

Non-overlapped!

				1	0								……	

Scenario 3:

2nd read: - - - - -10- - - - -…..

save the assemble haplotype to a temporary container.

Use a new array and plug in the new read

Easy problem algorithm

What we have eventually:

bit 0	bit 1	bit 2	bit 3
0	0	1	1

bit 4	bit 5	bit 6	bit 7	bit 8	bit 9	bit 10	bit 11
0	1	1	1	0	0	1	1

.....

bit 97	bit 98	bit 99	bit 100
0	0	1	1

Global container for previous haplotype

2^n -1 possibilities

Easy problem accuracy

- Optimistic accuracy:

The one possibility with the highest accuracy
In our case: 100% accurate

- Worst accuracy:

assume at every non-overlap point , we had a wrong guess
Switch distance: n

Medium problem statement

The same thing with Easy problem:

- Has homozygous SNPs:

  Need to extract heterozygous SNPs → what's the threshold?

- Has non-overlapping after excluding homozygous SNPs! → 2^n -1 possibilities!

Difference from Easy problem:

- With errors!

  Need to count the 1s and 0s, calculate the frequency → relatively longer time and larger space

Medium problem algorithm

1st stage: Extract homozygous SNPs

0	0	1	1				
	1	0	0	1			
	0	1	1	1			
			0	1			
			1	0	0	0	
			1	1	1	0	
				1	0	0	…
…	…	…	…	…	…	…	…

Threshold matters!

In order to maximize accuracy, considers the size of overlapped column

Medium problem algorithm

1st stage: Extract homozygous SNPs

0	0	1	1				
	1	0	0	1			
	0	1	1	1			
			0	1			
			1	0	0	0	
			1	1	1	0	
				1	0	0	…
…	…	…	…	…	…	…	…

Threshold matters!

In order to maximize accuracy, considers the size of overlapped column

For long column: higher threshold
eg. 1/8 error rate

Medium problem algorithm

1st stage: Extract homozygous SNPs

0	0	1	1				
	1	0	0	1			
	0	1	1	1			
			0	1			
			1	0	0	0	
			1	1	1	0	
				1	0	0	…
…	…	…	…	…	…	…	…

Threshold matters!

In order to maximize accuracy, considers the size of overlapped column

For short column: lower threshold
eg. 1/4 error rate

Medium problem algorithm

1st stage: Extract homozygous SNPs

0	0	1	1				
	1	0	0	1			
	0	1	1	1			
			0	1			
			1	0	0	0	
			1	1	1	0	
				1	0	0	…
…	…	…	…	…	…	…	…

Threshold matters!

In order to maximize accuracy, considers the size of overlapped column

Gives homozygous extraction accuracy >96%

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

an “empty” hash map

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
…..	0	0	0	?
bit 99	0	0	0	?

1st read: 0011——…..

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

bit #	# of 0s	# of 1s	total	expectation
bit 0	1	0	1	0
bit 1	1	0	1	0
bit 2	0	1	1	1
…..	0	0	0	?
bit 99	0	0	0	?

1st read: 0011——…..

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

bit #	# of 0s	# of 1s	total	expectation
bit 0	2	0	2	0
bit 1	2	0	2	0
bit 2	0	2	2	1
…..	0	0	0	?
bit 99	0	0	0	?

Scenario 1:

2nd read: -0110——…..

over 75% the same with the hash map expectation

plug in the read, update expectation

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

bit #	# of 0s	# of 1s	total	expectation
bit 0	2	0	2	0
bit 1	2	0	2	0
bit 2	0	2	2	1
.....	0	0	0	?
bit 99	0	0	0	?

Scenario 2:

2nd read: -1001——.....

over 75% opposite with the hash map expectation

plug in the *flipped read, update expectation*

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

bit	# of	# of	tota	expecta
bit	2	0	2	0
bit	2	0	2	0
bit	0	2	2	1
.....	0	0	0	?
bit	0	0	0	?

undecided reads

-1111——.....

Scenario 3:

3rd read: -1111——.....

25% < same rate <75%

save the read to the temporary container

Medium problem algorithm

2nd stage: analyze heterozygous SNPs

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	1	1	1
bit 5	0	1	1	1
bit 6	0	0	0	?
...				

undecided reads

-1111——.....

bit 0	bit 1	bit 2	bit 3
0	0	1	1

Scenario 4:

4th read: - - - - -11——.....

non overlapped with the hash map expectation

1.try to plug the undecided reads

2. try to plug the read again

3. if failed, save the expected haplotype to temporary container, clear up the hash map, then plug the read.

medium problem algorithm

What we have eventually:

bit 0	bit 1	bit 2	bit 3
0	0	1	1

bit 4	bit 5	bit 6	bit 7	bit 8	bit 9	bit 10	bit 11
0	1	1	1	0	0	1	1

.....

bit 97	bit 98	bit 99	bit 100
0	0	1	1

Global container for previous haplotype

2^n -1 possibilities

Medium problem accuracy analysis

- Optimistic accuracy:
  The one possibility with the highest accuracy
  In our case: >88% accurate for low error rate data
  >75% for high error rate data
- Worst accuracy:
  assume at every non-overlap point , we had a wrong guess
  Switch distance: n

Observation: lower error rate for larger data analysis: more overlap leads to higher accuracy

Hard problem statement

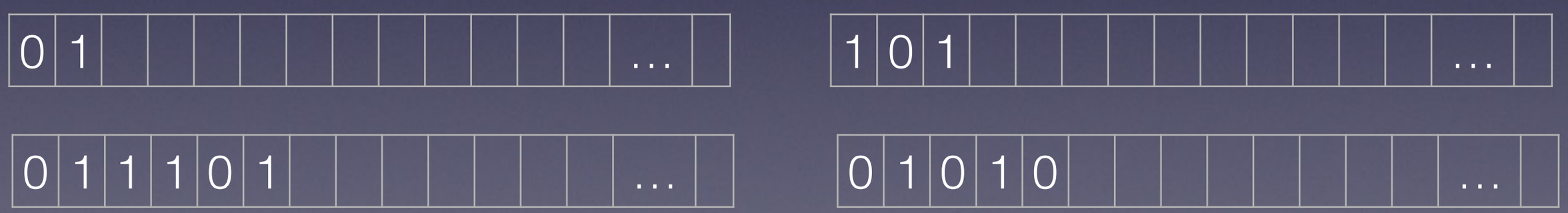
What's so special about multiple chromosome haplotypes?

- no homozygous or heterozygous SNPs!

  —All depends on frequency!

- might have reads that's non-overlapped with 1 or multiple haplotypes

  — Even harder to track all the possibilities



next read: - - - - 0011——.....

Hard problem algorithm

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

1st read: 1111——…..

plug in the first haplotype

Hard problem algorithm

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	1	1	1
bit 1	0	1	1	1
bit 2	0	1	1	1
bit3	0	1	1	1
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

2nd read: 1110——…..
find out which haplotype has the highest frequency of sameness

Scenario 1:
If frequency >threshold% bits are the same:
plug the read in that haplotype

Hard problem algorithm

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	1	1	1
bit 1	0	1	1	1
bit 2	0	1	1	1
bit3	0	1	1	1
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	0	0	?
…				

1110——…..

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

2nd read: 1110——…..

find out which haplotype has the highest frequency of sameness

Scenario 2:

If frequency <threshold% bits are the same:

put the read into “error read container”

Hard problem algorithm

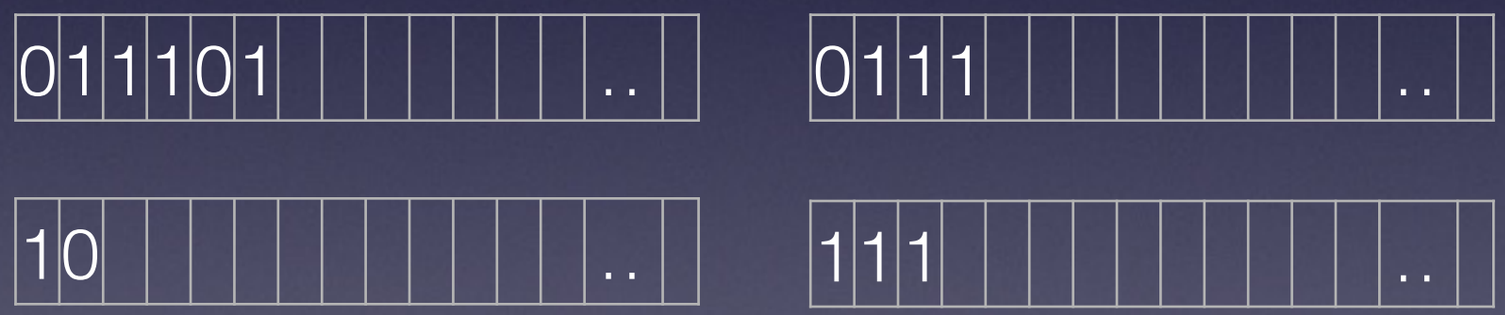
bit #	# of 0s	# of 1s	total	expectation
bit 0	0	1	1	1
bit 1	0	1	1	1
bit 2	0	1	1	1
bit3	0	1	1	1
bit 4	0	0	0	?
…				

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 4	0	0	0	?
…				

1110——…..

bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

3nd read: - - - -1110——…..



bit #	# of 0s	# of 1s	total	expectation
bit 0	0	0	0	?
bit 1	0	0	0	?
bit 2	0	0	0	?
bit3	0	0	0	?
bit 6	0	0	0	?
…				

Scenario 3: read is non-overlapped with some of the haplotypes

Hard problem algorithm

3nd read: - -1110——.....

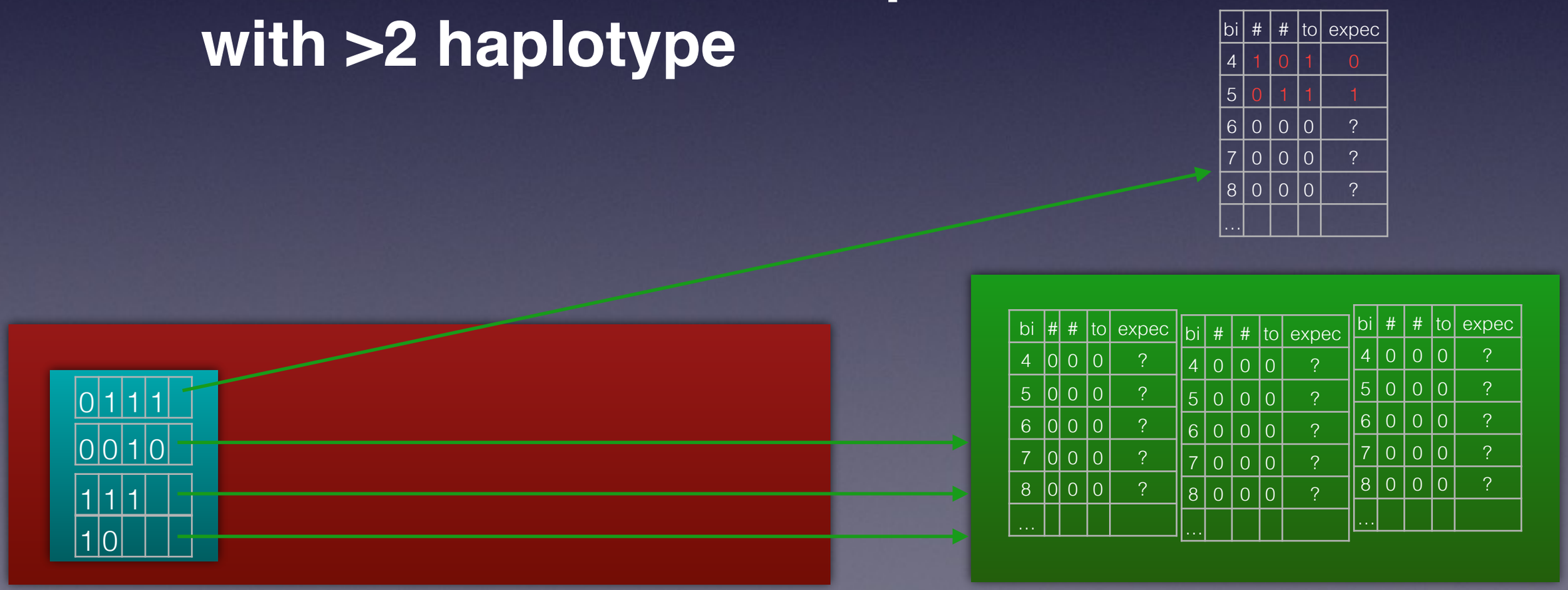
sub-scenario 3-1: non-overlap with 1 haplotype

plug the read in that haplotype if none of the other haplotypes meets the frequency threshold

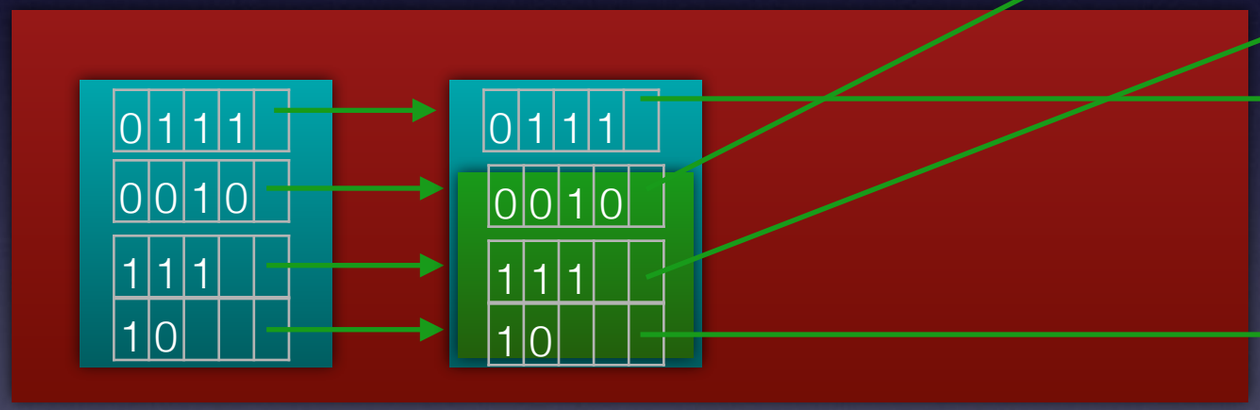
Hard problem algorithm

3nd read: - - - -1110——.....

sub-scenario 3-2: non-overlap with >2 haplotype



Hard problem algorithm

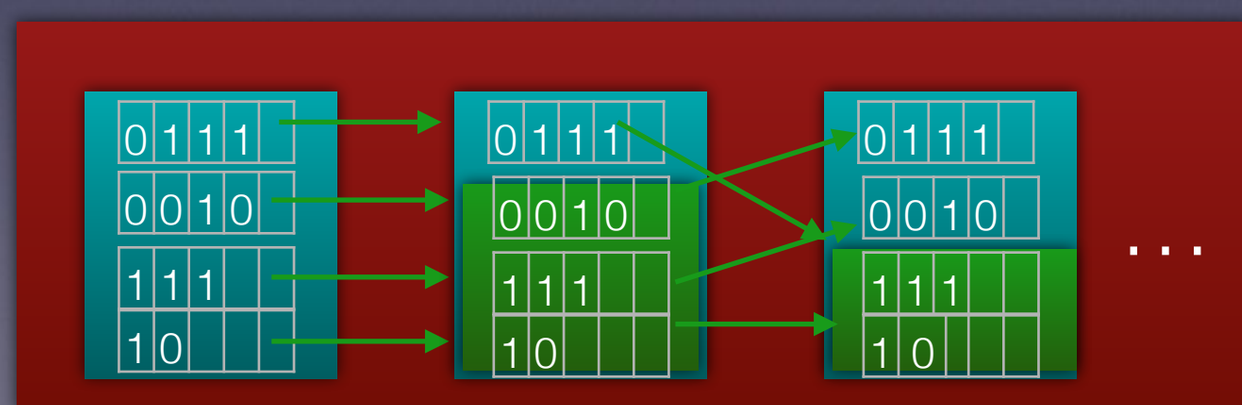


bi	#	#	to	expec
4	0	0	0	?
5	0	0	0	?
6	0	0	0	?
7	0	0	0	?
8	0	0	0	?
…				

bi	#	#	to	expec
4	1	0	1	0
5	0	1	1	1
6	0	0	0	?
7	0	0	0	?
8	0	0	0	?
…				

bi	#	#	to	expec
4	0	0	0	?
5	0	0	0	?
6	0	0	0	?
7	0	0	0	?
8	0	0	0	?
…				

bi	#	#	to	expec
4	0	0	0	?
5	0	0	0	?
6	0	0	0	?
7	0	0	0	?
8	0	0	0	?
…				



Hard problem analysis

Accuracy

All depends on the threshold!

The algorithm is working, still adjusting the threshold to find a lowest error rate

Time:

Always refers to the hash map, giving us O(nlogn) time complex

Finding out all the possibilities from the data structure costs exponential time