# Algorithms in Genome Analysis, Spring 2023

Veli Mäkinen

# Week 2

Haplotype Assembly

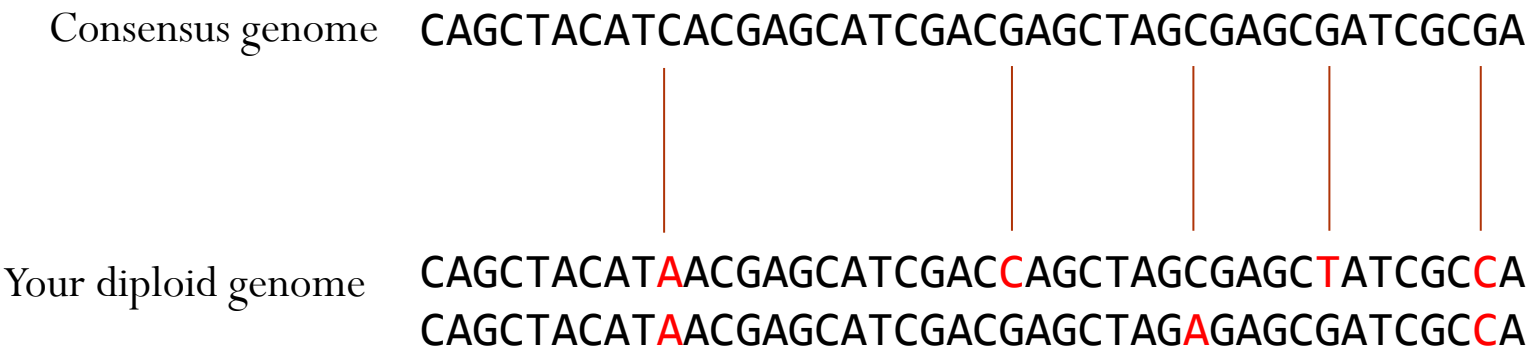
# High-throughput long read sequencing

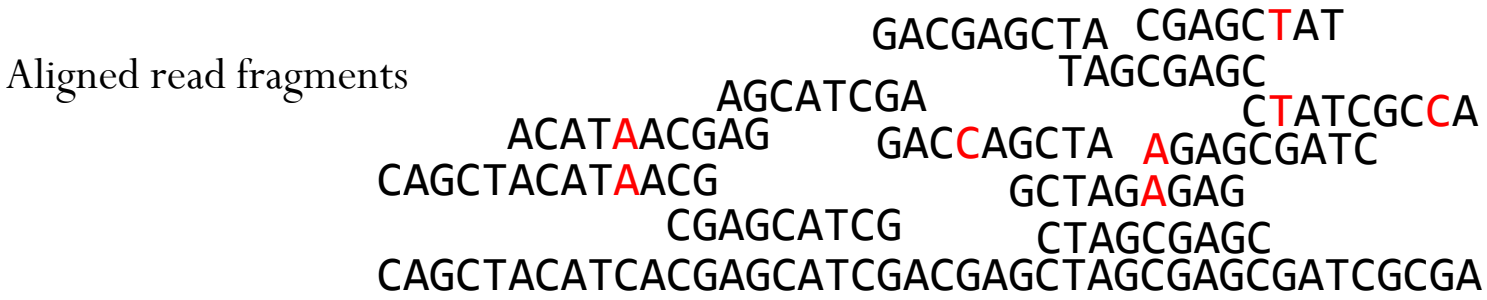
- In last 10 years, sequencing technologies have evolved so that reads of length tens of kilobases can be obtained.
- Reads can be obtained from single cells without the need of amplification.
- Sequencing errors are at pretty high level (>10%) compared to short, ~100 bp, reads (1-2%).
- Long reads enable phasing of diploid genomes:
  - *Haplotype assembly*

# Haplotype assembly

Phasing diploid genomes with long reads

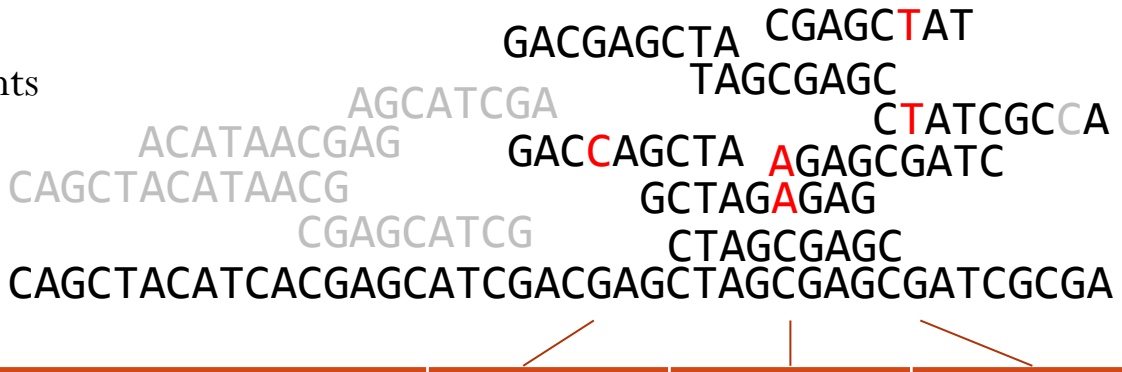
# Variation calling, diploid genomes



# From variants to binary matrix

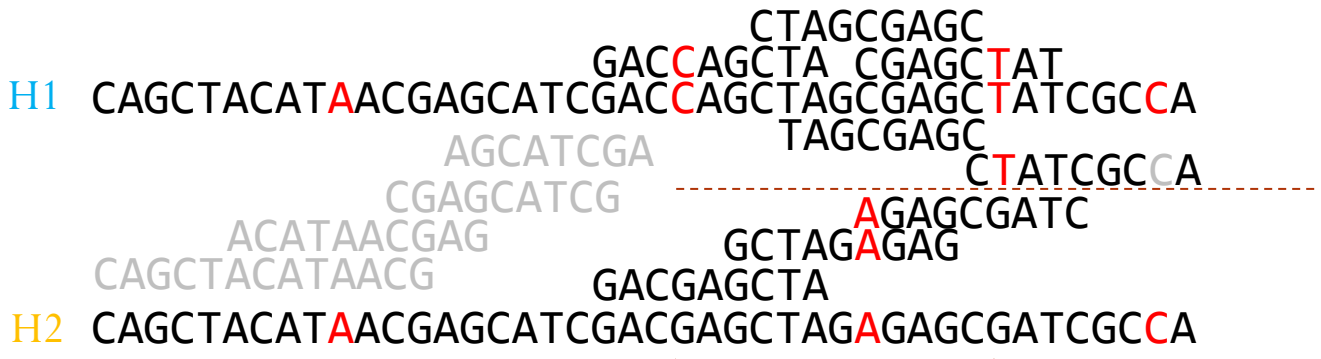
Aligned read fragments




| Reads \ SNPs | G->C | C->A | G->T |
|---|---|---|---|
| r1=GACGAGCTA | 0 | - | - |
| r2=GACCAGCTA | 1 | - | - |
| r3=GCTAGAGAG | - | 1 | - |
| r4=CTAGCGAGC | - | 0 | - |
| r5=TAGCGAGC | - | 0 | - |
| r6=CGAGCTAT | - | 0 | 1 |
| r7=AGAGCGATC | - | 1 | 0 |
| r8=CTATCGCCA | - | - | 1 |

# Perfect Haplotype assembly

Phased fragments

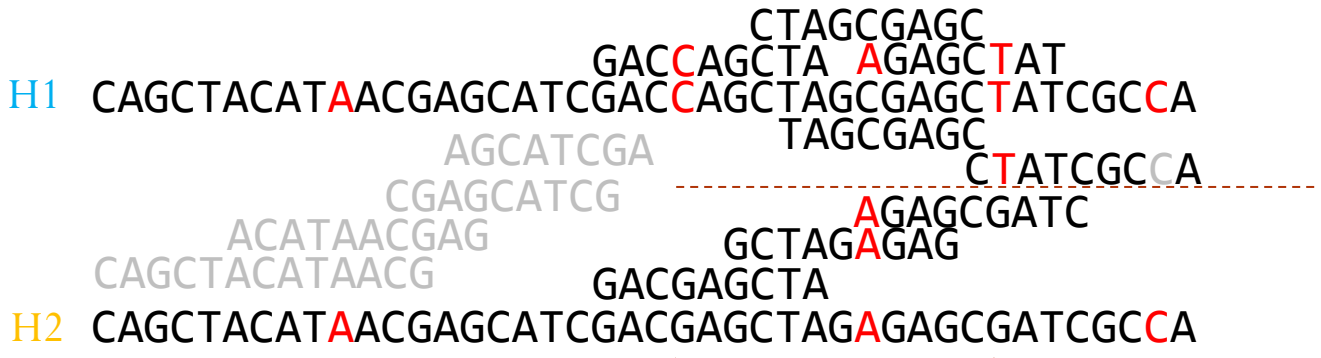



| Reads \ SNPs | G->C | C->A | G->T |
|---|---|---|---|
| r1=GACGAGCTA | 0 | - | - |
| r2=GACCAGCTA | 1 | - | - |
| r3=GCTAGAGAG | - | 1 | - |
| r4=CTAGCGAGC | - | 0 | - |
| r5=TAGCGAGC | - | 0 | - |
| r6=CGAGCTAT | - | 0 | 1 |
| r7=AGAGCGATC | - | 1 | 0 |
| r8=CTATCGCCA | - | - | 1 |

H1

H2

# Haplotype assembly with minimum error correction




Phased fragments

| Reads \ SNPs | G->C | C->A | G->T |
|---|---|---|---|
| r1=GACGAGCTA | 0 | - | - |
| r2=GACCAGCTA | 1 | - | - |
| r3=GCTAGAGAG | - | 1 | - |
| r4=CTAGCGAGC | - | 0 | - |
| r5=TAGCGAGC | - | 0 | - |
| r6=AGAGCTAT | - | 1->0 | 1 |
| r7=AGAGCGATC | - | 1 | 0 |
| r8=CTATCGCCA | - | - | 1 |

H1

H2

# Naive algorithm

- Go through all $2^r$ partitionings, where r is the #reads
- For each column in a fixed partitioning, choose consensus bit for each part, flip other bits
- Choose partitioning with minimum amount of flips

H1

H2

| Reads \ SNPs | G->C | C->A | G->T |
|---|---|---|---|
| r1=GACGAGCTA | 0 | - | - |
| r2=GACCAGCTA | 1 | - | - |
| r3=GCTAGAGAG | - | 1 | - |
| r4=CTAGCGAGC | - | 0 | - |
| r5=TAGCGAGC | - | 0 | - |
| r6=AGAGCTAT | - | 1->0 | 1 |
| r7=AGAGCGATC | - | 1 | 0 |
| r8=CTATCGCCA | - | - | 1 |

# Faster algorithm

- Consider column i. Only $2^c$ partitionings having different number of required bit flips at column i, where c is the *maximum coverage*.
- For each such *active* partitioning (C1,C2) at column i, let c(C1,C2,i) be the minimum number of bit flips at submatrix M[1..r, 1..i]. Let flips(C1,C2,i) give the number of flips induced by (C1,C2) at column i.
- Then we obtain a dynamic programming recurrence

  c(C1,C2,i)=min {c(C'1,C'2,i-1)+flips(C1,C2,i)
  | (C'1,C'2) is an *active* partitioning at column i-1 *compatible with* (C1,C2) at column i}.
- Total running time $O(2^c\, 2^c\, nc)=\tilde{O}(4^c)$.

# Example

| Reads \ SNPs | G->C | C->A | G->T |
|---|---|---|---|
| r1=GACGAGCTA | 0 | - | - |
| r2=GACCAGCTA | 1 | - | - |
| r3=GCTAGAGAG | - | 1 | - |
| r4=CTAGCGAGC | - | 0 | - |
| r5=TAGCGAGC | - | 0 | - |
| r6=AGAGCTAT | - | 1 | 1 |
| r7=AGAGCGATC | - | 1 | 0 |
| r8=CTATCGCCA | - | - | 1 |

c({{1,2},{},1})=flips({1,2},{},1})=1
c({{1},{2},1})=flips({1},{2},1})=0

c({{3,4,5,6,7},{},2})=c({{1},{2},1})+2=2
…
c({{4,5},{3,6,7},2})=c({{1},{2},1})+0=0
…
c({{4,5,6},{3,7,},2})=c({{1},{2},1})+1=1
….

c({{6,7,8},{},3})=c({{4,5},{3,6,7},2})+1=1
c({{6,7},{8},3})=c({{4,5},{3,6,7},2})+1=1
c({{6,8},{7},3})=c({{4,5,6},{3,7},2})+0=1

# Haplotype assembly - summary

- Perfect case is easy to solve
- With minimum error correction, the problem becomes NP-hard (proof omitted here, see course book)
- Dynamic programming gives $\tilde{O}(4^c)$ time algorithm, with $c$ the maximum coverage
  - This can be improved to $\tilde{O}(2^c)$ with some clever organization of the computation using gray-codes
- Many optimal solutions / different phasings possible
- Distant mutations / short reads result to arbitrary phasing
- Not just SNPs, the model can be extended to indels