Project 2: Haplotype Assembly

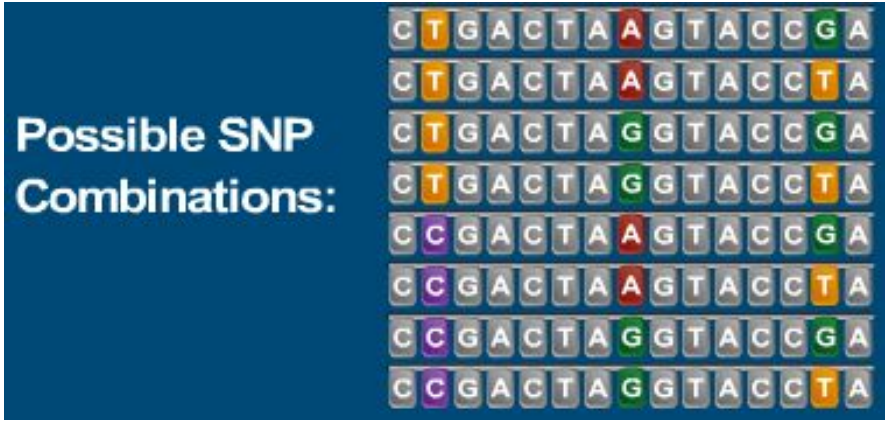
CSCM124: DIS 1B

Alfred Lucero

Biological Problem: Haplotype Assembly

Motivation/Background

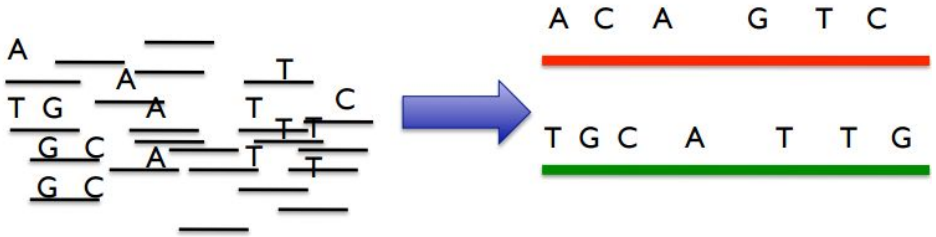
- **SNPs** can be **rearranged** to all possible SNP combinations and each combination is a **haplotype** along a particular region of DNA
- Everything comes in **pairs** as we get one haplotype from the mother and one from the father, making up our **“SNP Profile”**



http://learn.genetics.utah.edu/content/pharma/snips/

Assembling the Haplotypes

- **Problem:** Using a sequencer that generates reads from both haplotypes, assemble the proper pair of haplotypes (no sequencer error)
- Each read is coming from one haplotype, but we **don't know which**.
- The pair of haplotypes are represented as a pair of **complementary** binary strings.



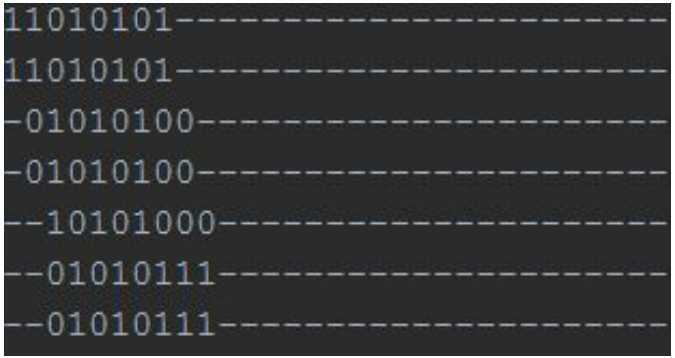
Computational Problem: Time/Spatial Complexity and Accuracy

Input/Output Format

Input: (for read and haplotype data)

Read Data: 1s, 0s, dashes(-) = no coverage at that base

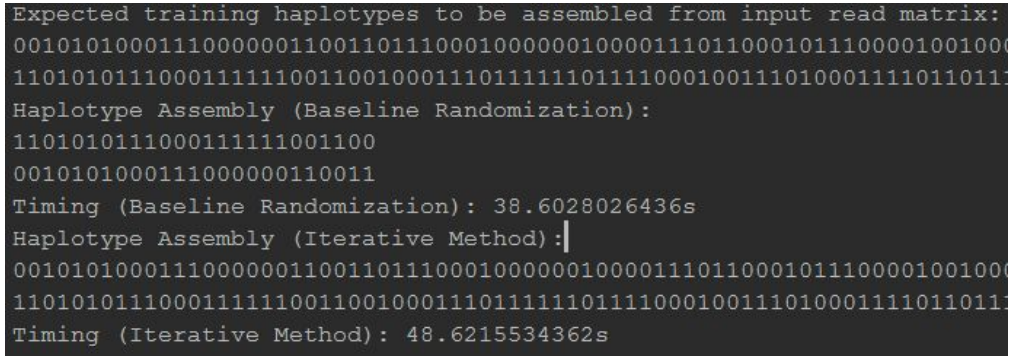
Haplotype Data: 1s, 0s, non-delimited, each row = 1 haplotype



Output: (for haplotype data)

Haplotype Data: return it in the form it came in or construct the longest stretches you are confident in and set the possible error ones with dashes

Timing and output for each method displayed



Computational Problem

Given an MxN (1000x100) input read matrix of 1s, 0s, and dashes, accurately assemble the pair of complementary haplotypes given no sequencing error

- Do so with **best spatial** and **time** complexities taken into account and with **accurate** haplotype results
- M = Number of reads = 1000
- N = Number of bases = 100

h0:	?	?	?	?	?	?	?	?	?	?	?	?	?
h1:	?	?	?	?	?	?	?	?	?	?	?	?	?

reads	0	1	2	3	4	5	6	7	8	9	10	11	12
read1	0	0	–	–	–	–	–	–	–	–	–	–	–
read2	–	–	1	1	–	–	–	–	–	–	–	–	–
read3	0	0	0	0	–	–	–	–	–	–	–	–	–
read4	–	–	1	0	1	–	–	–	–	–	–	–	–
read5	–	–	0	–	–	0	–	–	–	–	–	–	–
read6	–	–	–	0	–	–	–	–	–	–	1	1	–
read7	–	–	–	–	0	0	0	–	–	–	–	–	–
read8	–	–	–	–	0	1	1	0	–	–	–	–	–
read9	–	–	–	–	–	–	–	–	1	1	–	–	–
read10	–	–	–	–	–	–	–	1	1	0	–	–	0

Baseline Method 1: Check All Combinations O(N*2^N)

Base Algorithm 1: Check all Combinations

Create all 2^N Combinations of Binary Strings (25 Bases in our Case)

While no combination matches with any of the haplotype results

Check if combination Xi matches with first 25 bases of either haplotype

If it matches, break out and compose the complement haplotype as well

Time Complexity: **O(N*2^N)**

Spatial Complexity: **O(N*2^N)**

Accuracy: **1/ (2^N)** chance of guessing one of the haplotypes correctly

Improved Iterative Method/Baseline Method 2: Scan Entire Matrix O(MN)

Base Algorithm 2: Iterative Approach

For each row, i, in input_matrix

For each column, j, in input_matrix

If the haplotype[last_index] is equal to the input_matrix[i][last_index] and j > last_index and

Input_matrix[i][j] is not a dash

Append input_matrix[i][j] to the haplotype and set last_index added to j

Time Complexity: **O(MN)**

Spatial Complexity: **O(MN)**

Accuracy: **100%** chance of assembling the haplotypes correctly

Greedy Method: Continue From Last Overlap O(N)

Greedy Algorithm: Last Overlap Method

While i and j is less than the number of rows and columns

If haplotype's last index matches with matrix's

While j < number of columns

If there isn't a dash

Append that base to haplotype

Set last_index = j

Else proceed to the next row starting from

Last_index and break out

Else proceed to next row starting from last_index

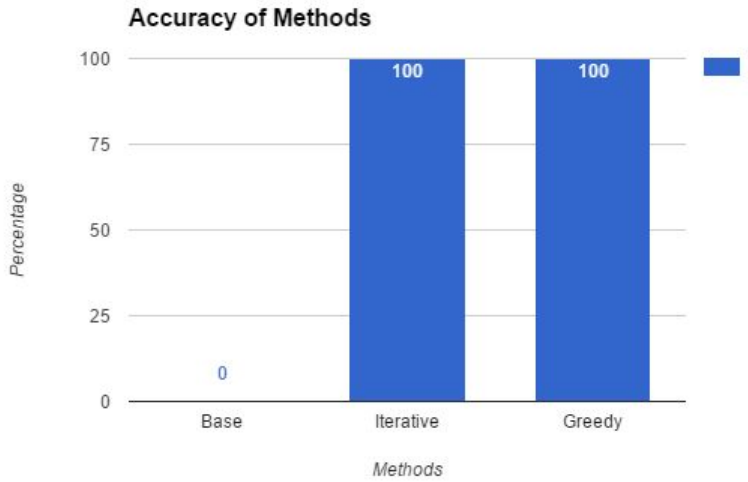
Time Complexity: **O(N)**

Spatial Complexity: **O(MN)**

Accuracy: **100%** chance of assembling the haplotypes correctly

Comparing Performance of Three Methods

Performance Comparisons: Accuracy

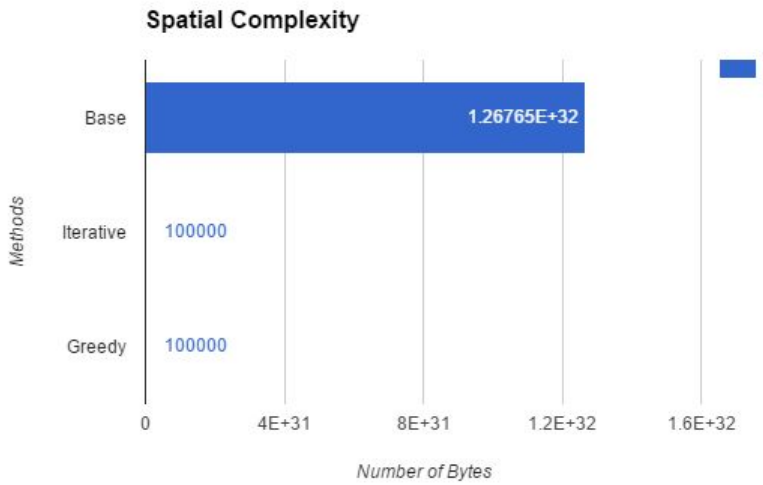


Base Method: Guess and Check from all Linear Combinations

Do 25 SNPs at a time

1/(2^N) = 1/(2^100) = 7.88*10^-31
~ **0%**

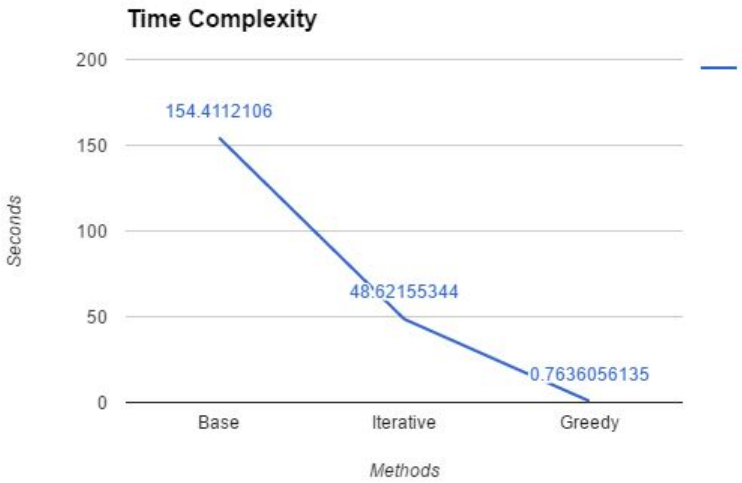
Performance Comparisons: Spatial



Clearly the baseline method requires a lot of space to hold all **2^100 combinations of 100 byte strings**

The other two methods hold the input_matrix **(100*1000 bytes)**

Performance Comparisons: Time



Important Notes: **using python's timeit module**

Base method data is from the method being **run once** and it would take 38.6028s to compare 25 bases at a time -> multiply by 4 we get 154.4s

Iterative and greedy methods are run **1000 times** each and the time is summed up respectively

The greedy method improves dramatically as it is in **linear time**
~64 times as fast as iterative run 1000 times
~202 times as fast as base run one time

Implications and Observations

Implications and Observations

- Perhaps **we can modify the current linear greedy method to account for errors** and keep track of the positions that may be susceptible to error by seeing multiple overlaps that may pertain to one haplotype or another -> algorithm will probably be polynomial to handle that case
- Haplotype inference is important for many types of **analyses of genetic variation in the human genome,** coming from high-throughput sequencing technologies
- In real-life **errors will be high** in comparison to no sequencing errors as in this case so there exists greedy and stochastic algorithms to solve this but it is known to be **NP-hard**

References

http://learn.genetics.utah.edu/content/pharma/snips/

http://bioinformatics.oxfordjournals.org/content/26/12/i183.abstract

Professor Eskin's Slides for Haplotype Assembly :D