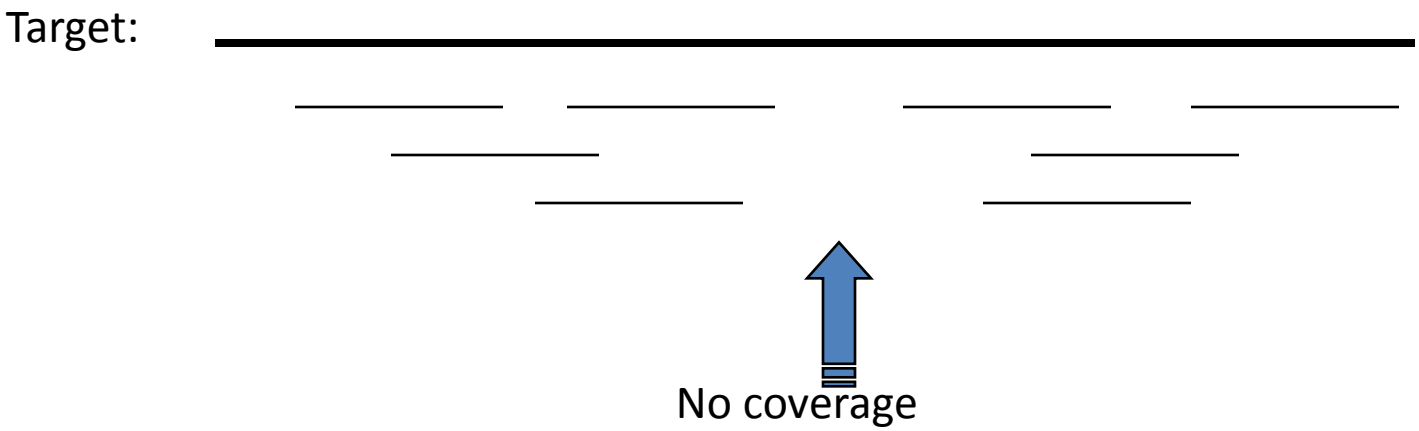
Genome sequencing and assembly

Biological background

- Problem as puzzle
- We do not know which letter from the set {A, C, G, T} is written on each card, but we do know that cards in the same position of opposite stands from a complementary pair.
- Our goal is obtain the letters using certain *hint*, which are (approximate) substrings of the rows.

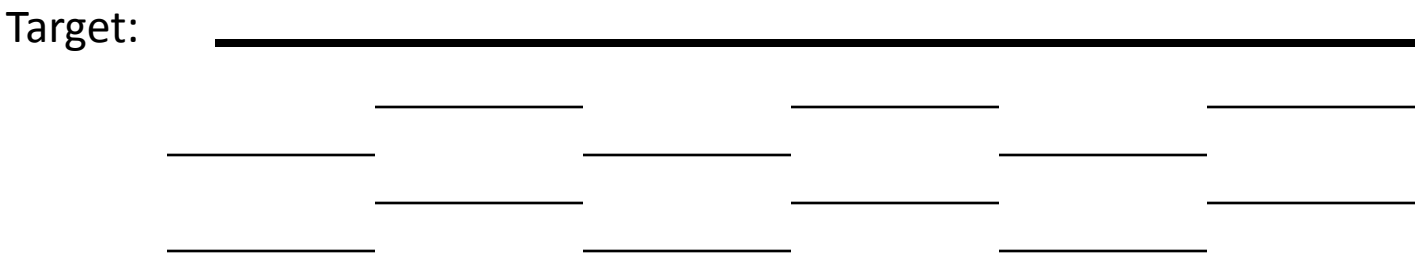
Quality Metrics

- The *coverage* at position *i* of the target or consensus sequence is the number of fragments that overlap that position



Quality Metrics

- Linkage – the degree of overlap between fragments



Perfect coverage, poor average linkage
poor minimum linkage

Biological background
--Complications

- The main factors that add to the complexity of the problem are:
  - Error
  - Unknown orientation
  - Repeated regions
  - Lack of coverage.

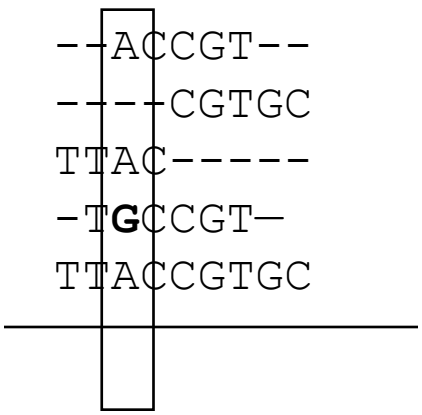
Biological background
--Complications

Errors

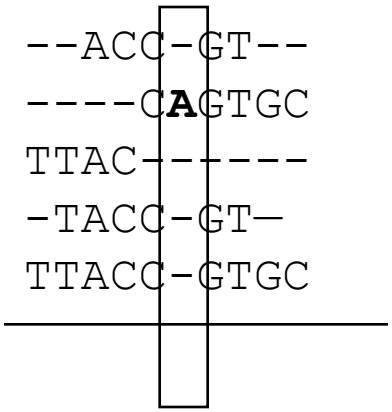
- It usually means algorithms that require more time and space when computer program deal with error.
- The simplest errors are called *base call* errors and comprise base substitutions, insertions and deletions in the fragments.
- Base call errors occurs in practice at rates varying from 1 to 5 errors every 100 characters.

Real World Complications

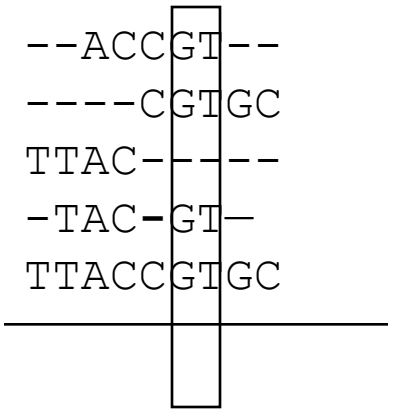
- Base call errors
- Chimeric fragments, contamination (e.g. from the vector)



Base Call Error



Insertion Error



Deletion Error

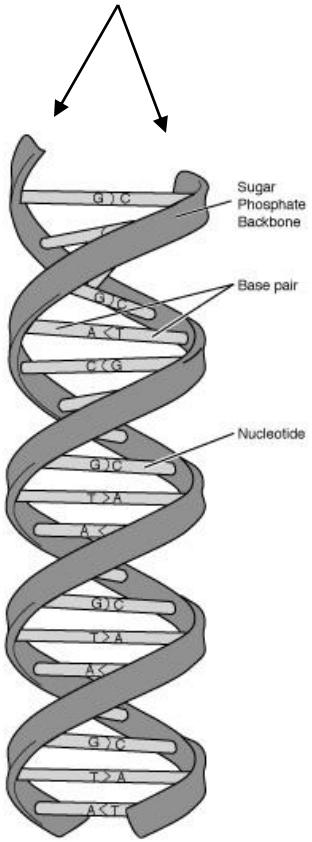
Biological background
--Complications

Unknown orientation

- We generally do not know to which strand a particular fragment belongs to.
- The input fragments as being all approximate substrings of the consensus sought either as given or in reverse complement.

Unknown Orientation

A fragment can come from either strand




```
CACGT     →   CACGT
ACGT      →   -ACGT
ACTACG    ←   --CGTAGT
GTACT     ←   -----AGTAC
ACTGA     →   ---------ACTGA
CTGA      →   ----------CTGA
```

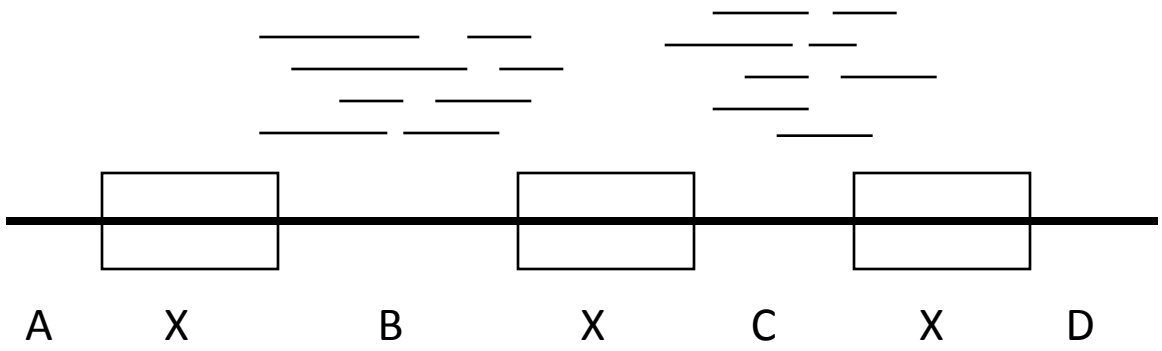
Biological background
--Complications

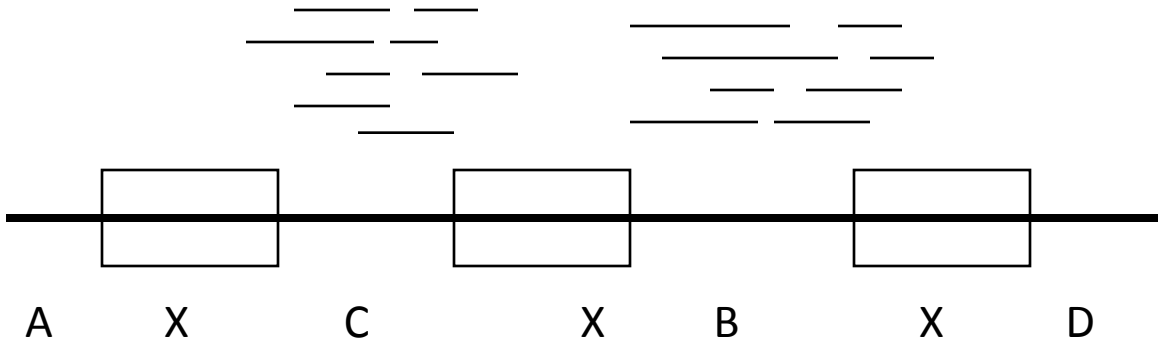
Repeated regions

- Problems:
  - If a fragment is totally contained in a repeat, we may have several places to put it in the final alignment. When the copies are not exactly equal, we may weaken the consensus by placing a fragment in the wrong way copy.
  - Repeats can be positioned in such a way as to render assembly inherently ambiguous.
- Direct repeats: repeated copies in the same strand.
- Inverted repeats: repeated regions in opposite strands

Repeats

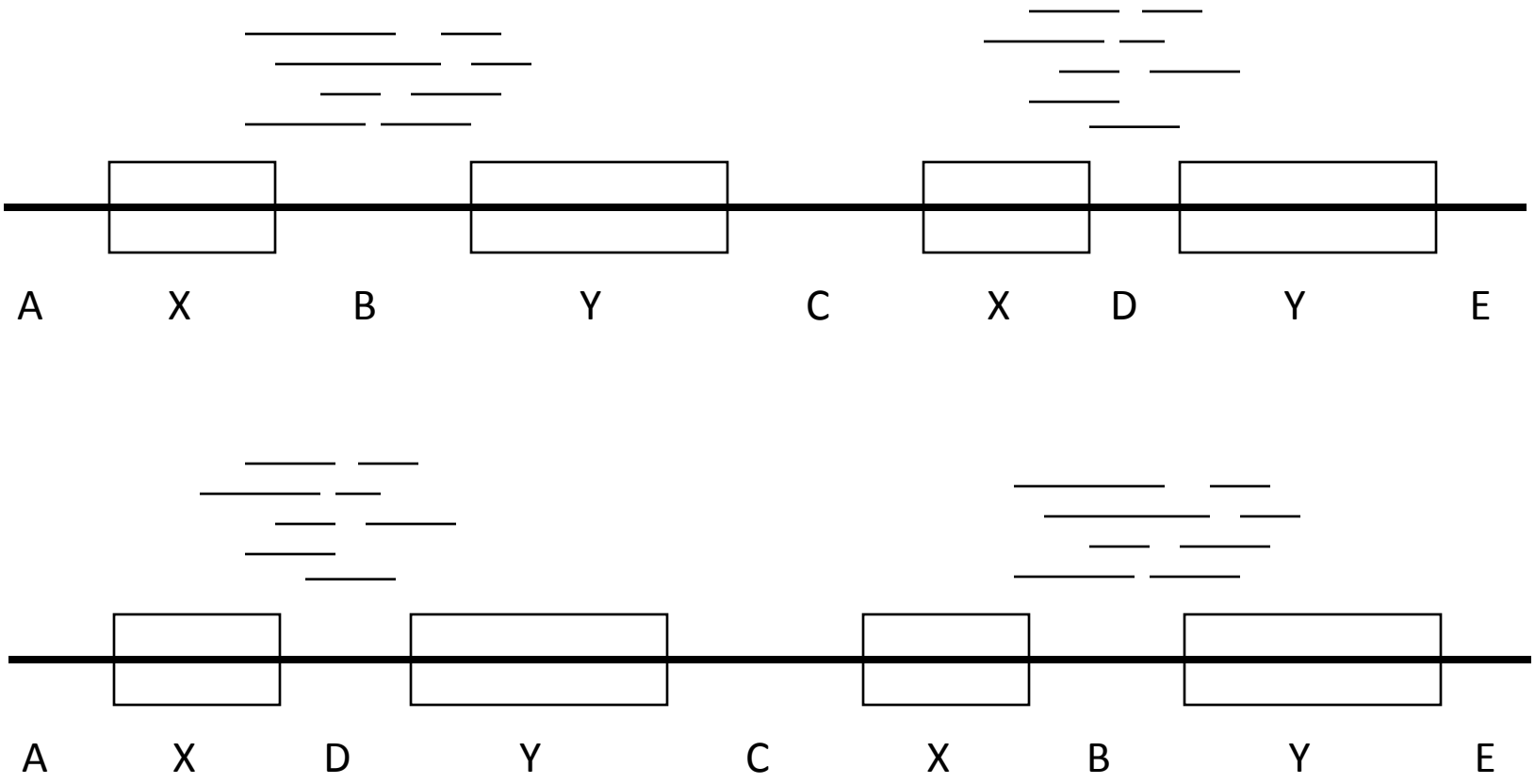
- Direct repeats





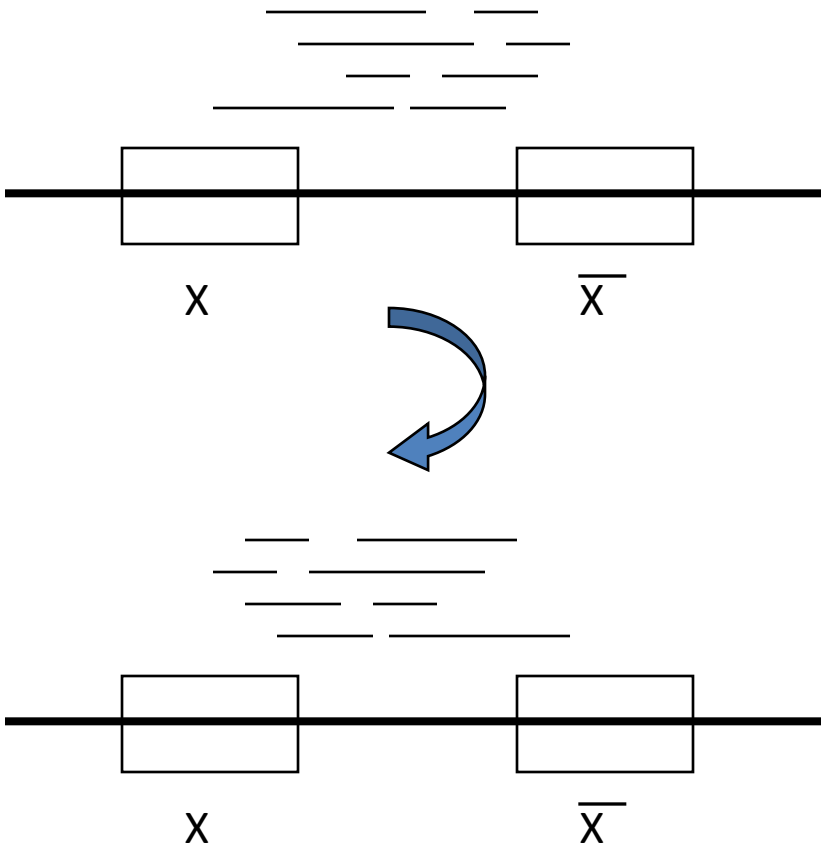
Repeats

- Direct repeats



Repeats

- Inverted repeats



Biological background
--Complications

Lack of coverage

- Coverage: position *i* of the target as the number of fragments that cover this position.
- Contigs: The contiguously covered regions
- Solution:
  - Sampling more fragments

Gene Sequencing

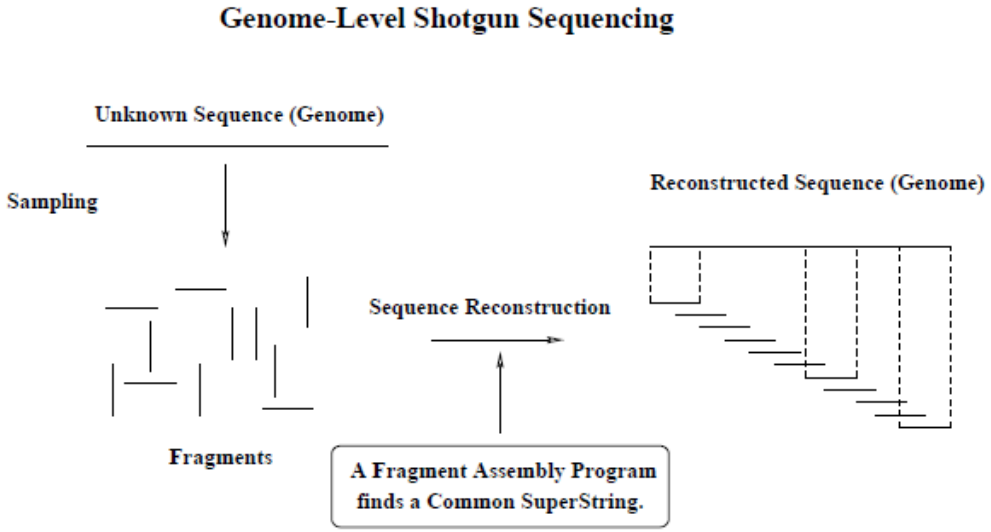
- Shotgun Sequencing (Fred Sanger 1982)

  1. Physically break the DNA
  2. DNA sequencer reads the DNA.
  3. Assembler reconstructs the original sequence.

- Assembly is *challenging*
  - Data contains errors
  - DNA has repetitive sections called repeats.
  - Gaps

Fragment Assembly

In *shotgun sequencing*, whole genomes are sequenced by making clones, breaking them into small pieces, and putting the pieces together again based on overlaps.



Note that the fragments are *randomly* sampled, and thus no positional information is available.

Gaps

Since we rely on fragment overlaps to identify their position, we must sample sufficient fragments to ensure enough overlaps.

Let T be the length of the target molecule being sequenced using n random fragments of length l, where we recognize all overlaps of length t or greater.

The *Lander-Waterman* equation gives the expected number of *gaps* g as

$$g = ne^{-n(l-t)/T}$$

Where does the e come from?

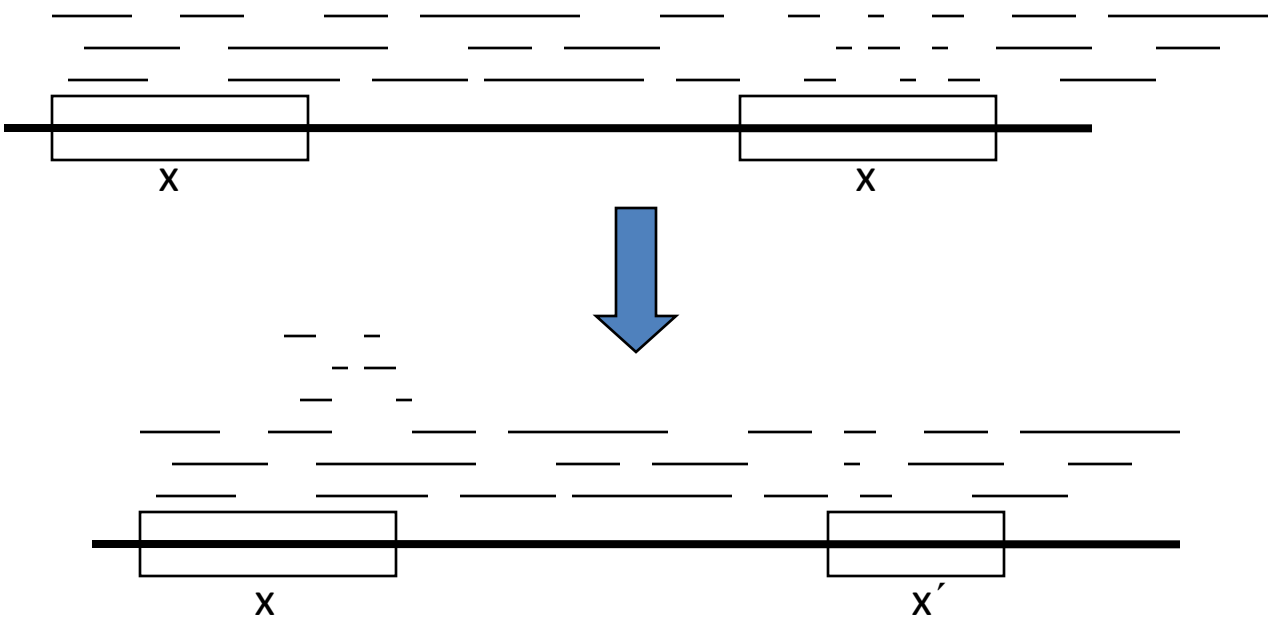
Suppose we have as many fragments as bases, i.e. $T = n$ and each fragment is length 1. The probability p that base i is *not* sampled is

$$p = (\frac{n-1}{n})^n \rightarrow \frac{1}{e}$$

Gene Sequencing

- Assembly Algorithms
  - **Shotgun sequencing assembly problem**
    - Find the shortest common superstring of a set of sequences.
    - Given strings {s1, s2, …} find the shortest string T such that every S_i is a substring of T.
    - This is NP-hard.
    - Approximation algorithm for this is efficient, the greedy algorithm.
    - Greedy algorithms were the first successful assembly algorithm implemented.
    - Used for organisms such as bacteria, single-celled eukaryotes.
    - Because of the greedy algorithm's limitations, two other algorithms were derived.
    - The first genome sequenced by shotgun sequencing was that of cauliflower mosaic virus, published in 1981.

Problems with the SCS model



- Directionality of fragments must be known
- No consideration of *coverage*
- Some simple consideration of *linkage*
- No consideration of base call errors

Gene Sequencing

- Assembly Algorithms
  - **Shotgun sequencing assembly problem**
- **Shortest common superstring**
  - Input: A collection, F, of strings (fragments)
  - Output: A shortest possible string S such that for every $f \in F$, S is a superstring of f.
- Example:
  - $F = \{\text{ACT, CTA, AGT}\}$
  - $S = \text{ACTAGT}$

Gene Sequencing

- Assembly Algorithms
  - **Overlap-layout-consensus**
    - Algorithm based on graph theory
    - A graph is constructed
      - » **nodes are reads**
      - » **edges represent overlapping reads**
    - A contig is a simple path in the graph
    - Simple path – contains each node at most once
    - An assembler builds the graph
    - Output is a set of nonintersecting simple paths, each path being a contig.

Gene Sequencing

- Assembly Algorithms
  - **Eularian path**
    - graph theory
    - Eularian path – a path that visits all edges of a graph
    - Breaks reads into overlapping n-mers.
    - k-mers refer to all the possible subsequences (of length k) from a read obtained through DNA Sequencing.
    - The amount of k-mers possible given a string of length, L, is L-K+1
    - The number of possible k-mers given n possibilities (4 in the case of DNA e.g. ACTG) is n^k
    - Basic problem is to find a path that uses all the edges.
    - Eularian path is more efficient. In practice both are equally fast.

Example - ACTTA and CTTAG
represents ACTTAG