NTUEE DCLAB

Problem-Based Learning

Oct. 18, 2022

Bonus Lab:

Smith-Waterman (SW) Algorithm for Short-read Mapping

Presenter: Chung-Hsuan Yang

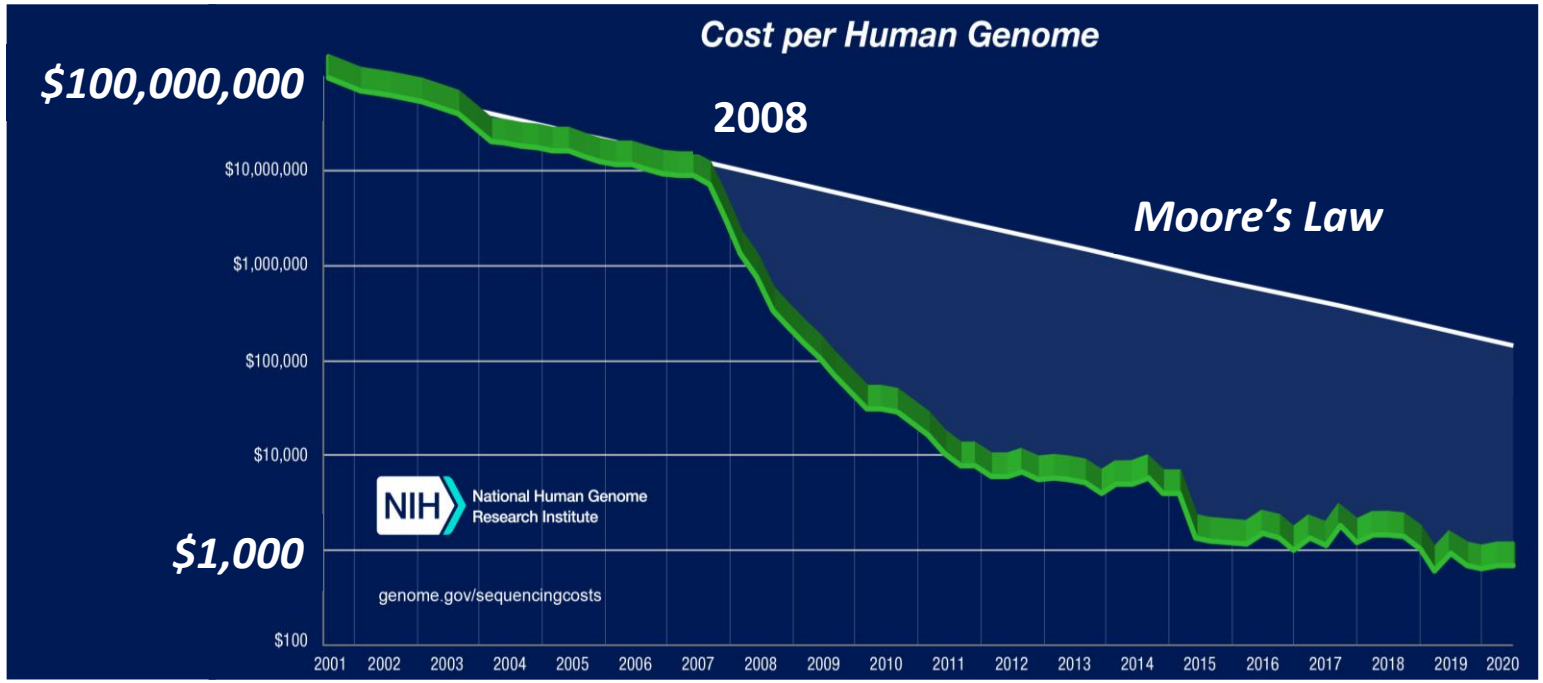
Advisor: Prof. Chia-Hsiang Yang

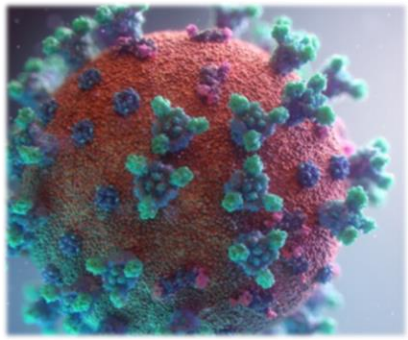
Graduate Institute of Electronics Engineering, National Taiwan University

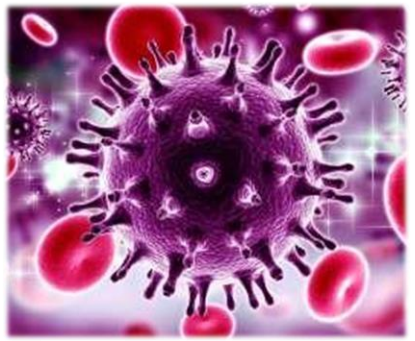
Outline

- Next-generation sequencing
- Short-read mapping algorithms
  - Seed-and-Extend
- Smith-Waterman Algorithm
- Affine gap penalty
- Lab introduction
- Hardware simulation
- Lab requirements
- System setup and run testing program
- Report regulations

Next-Generation Sequencing (NGS)



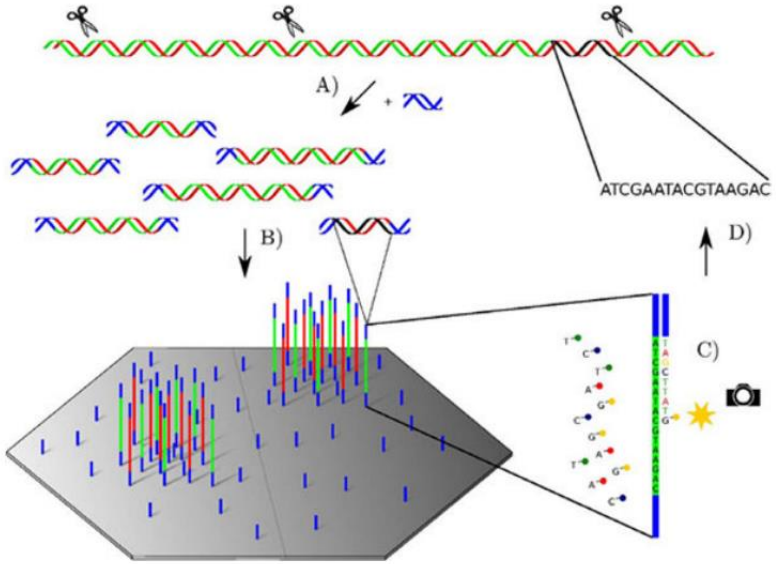




Next-Generation Sequencing (NGS)

- Shear the subject DNA sequence into fragments
- **Adapters** (blue) are attached to both sides of each fragment
- Read out the fragments when the complement bases are added
- The read out fragments are called **Short-read** (about 100 - 300 base pairs (bp))

Sequencing Machine

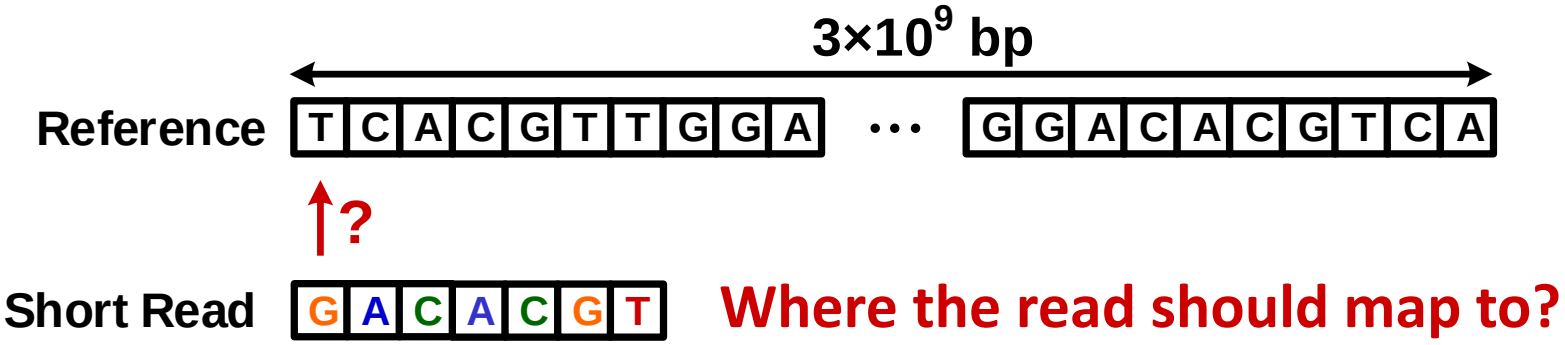


Output Result

< 1 day

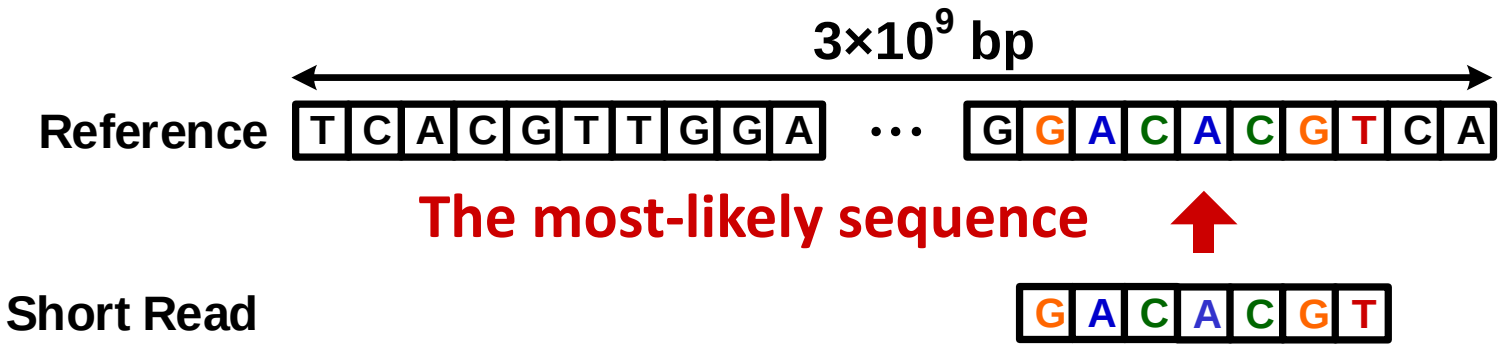
Short-read Mapping

- Find the most-likely sequence compared to the reference sequence



Short-read Mapping

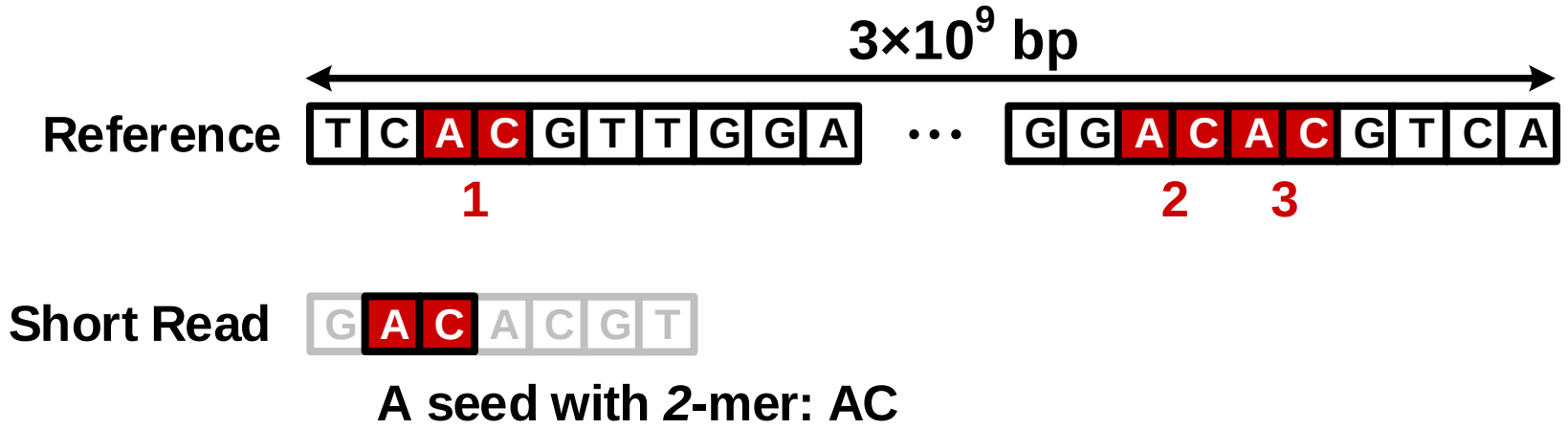
- Find the most-likely sequence compared to the reference sequence



- Heuristic method: sweep and compare base-by-base
  - Time-consuming (for hundreds of millions of short-reads)

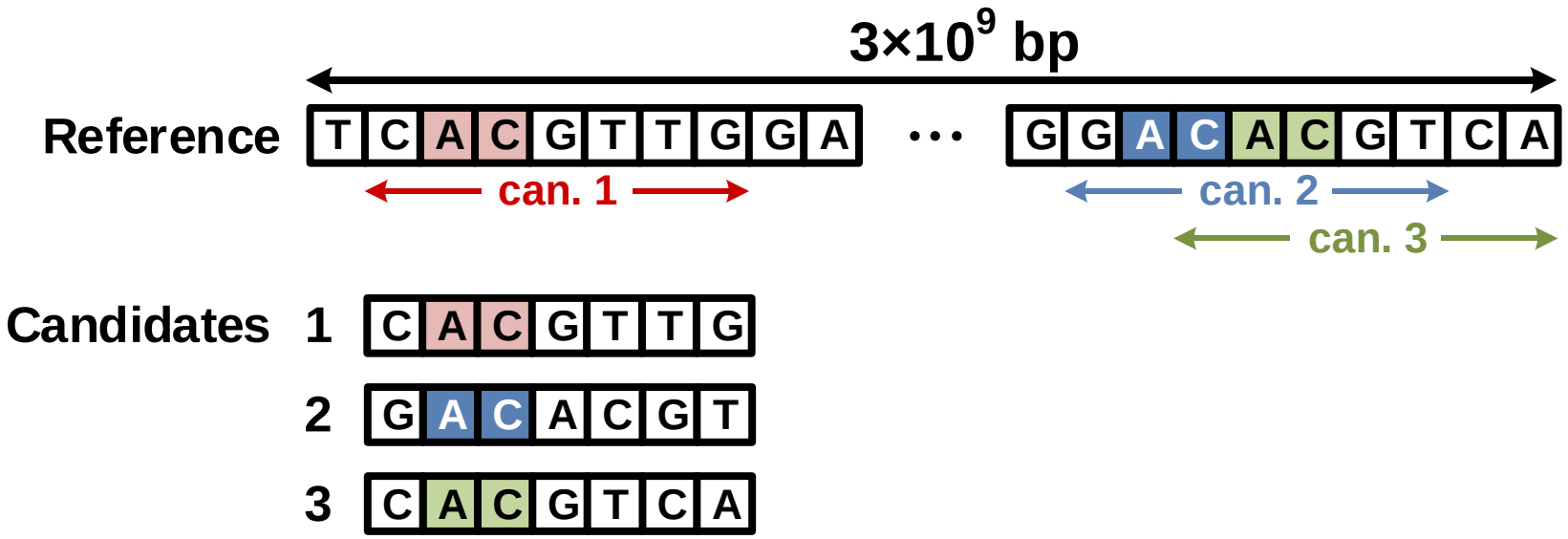
Seed-and-Extend

- Cut a short-read into shorter sequences (seeds)
  - k-mer: a sequence with length k
- Find all sequences (in the reference DNA) that exactly match the seed



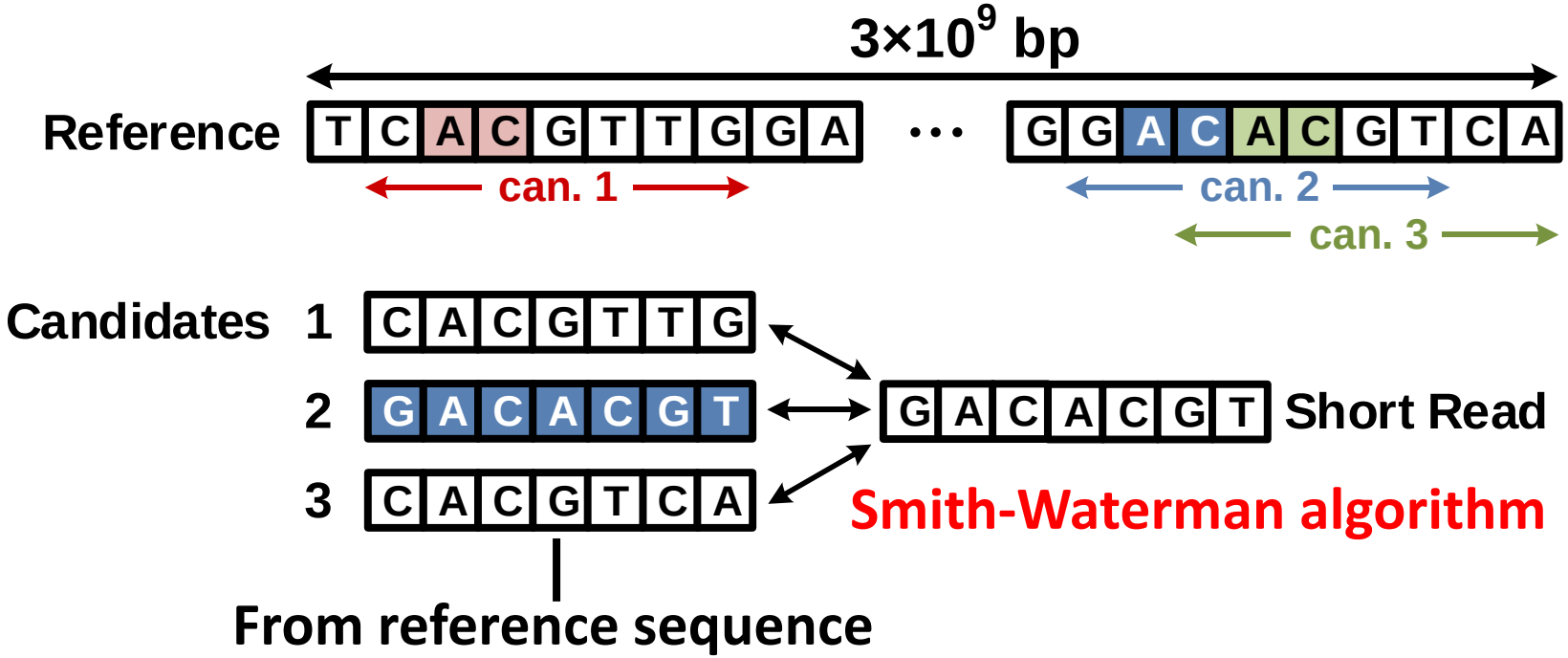
Seed-and-Extend

- Cut a short read into shorter sequences (seeds)
  - k-mer: a sequence with length k
- Find all sequences (in the reference DNA) that exactly match the seed
- Extend the sequences around all candidate locations



Seed-and-Extend

- Cut a short read into shorter sequences (seeds)
  - k-mer: a sequence with length k
- Find all sequences (in the reference DNA) that exactly match the seed
- Extend the sequences around all candidate locations
- Find the most-likely (inexact match) sequence with the read



Smith-Waterman Algorithm

- Calculate the similarity between two sequences
- Dynamic programming
- Input
  - Two sequences: a, b
  - Substitution score: $S(a_i, b_j)$
  - Gap penalty: G
- Two matrices
  - Score matrix
  - Direction matrix
- Output
  - Similarity score
  - Alignment path

Equation for scoring

$$H(i,j) = \max \begin{cases} H(i-1, j-1) + S(a_i, b_j) \\ H(i-1, j) - G \\ H(i, j-1) - G \\ 0 \end{cases}$$

$$S(a_i, b_j) = \begin{cases} 5, \text{ if } a_i = b_j \\ -3, \text{otherwise} \end{cases}$$

$$G = 2$$

Score matrix

		A	C	G	C	A	T
		0	0	0	0	0	0
A	0	5	3	1	0	0	0
C	0	3	10	8	6	5	3
G	0	1	8	15	13	11	9
T	0	0	6	13	12	10	16
C	0	0	4	11	18	16	14
A	0	0	5	9	16	**23**	21

Direction matrix

	A	C	G	C	A	T
A	↘	→	→			
C	↓	↘	→	→	→	→
G	↓	↓	↘	→	→	→
T		↓	↓	↘	↘	↘
C		↓	↓	↘	→	→
A		↓	↓	↓	↘	↓

Input sequences

Read (a) = ACGTCA

Reference (b) = ACGCAT

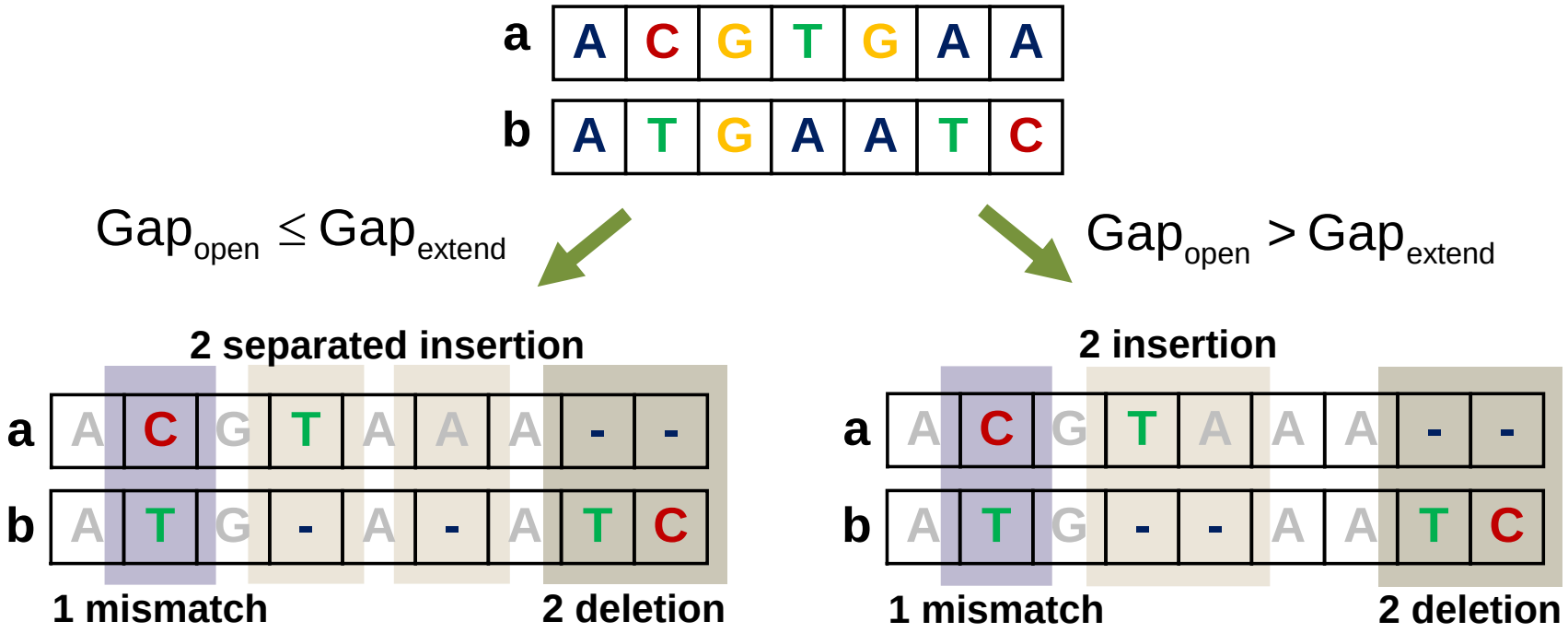
Alignment Result

Read (a) : ACGTCA

Reference (b) : ACG - CA

Affine Gap Penalty

- Consider gap opening and gap extension separately
- $Gap_{open} + Gap_{extend} \times (M-1)$, where M is the length of gap



Affine Gap Penalty

Matrix *H*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	5	0	0	0	0	0	0
C	0	0	3	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Matrix *I*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0
C	0	2	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Matrix *D*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	2	1	0	0	0	0
C	0	0	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Equation for scoring

$$H(i,j) = \max \begin{cases} H(i-1, j-1) + S(a_i, b_j) \\ I(i,j) \\ D(i,j) \\ 0 \end{cases}$$

$$I(i,j) = \max \begin{cases} H(i-1, j) + G_{open} \\ I(i-1, j) + G_{extend} \\ 0 \end{cases}$$

$$D(i,j) = \max \begin{cases} H(i, j-1) + G_{open} \\ D(i, j-1) + G_{extend} \\ 0 \end{cases}$$

$$S(a_i, b_j) = \begin{cases} 5, \text{ if } a_i = b_j \\ -2, \text{ otherwise} \end{cases}$$

$$G_{open} = -3$$

$$G_{extend} = -1$$

Affine Gap Penalty

Matrix *H*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	5	0	0	0	0	0	0
C	0	0	3	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Matrix *I*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0
C	0	2	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Matrix *D*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	2	1	0	0	0	0
C	0	0	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Direction *H*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	H	0	0	0	0	0	0
C	0	0	H	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Direction *I*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0
C	0	H	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Direction *D*

		A	T	G	A	A	T	C
	0	0	0	0	0	0	0	0
A	0	0	H	D	0	0	0	0
C	0	0	0	0	0	0	0	0
G	0							
T	0							
G	0							
A	0							
A	0							

Lab Introduction

- **實驗目的**
  - 實作 Dynamic Programming 運算，比較軟硬體運算速度差別
  - 了解 RS-232 輸入輸出界面，理解模組溝通的基礎模式與系統間通訊的匯流排 (bus) 觀念

- **FPGA 上實現序列比對演算法 Smith-Waterman (SW)**
  - PC 端透過 RS232 傳輸待測 reference 序列以及 short-read 序列給 FPGA
  - FPGA 接收資料並進行 SW algorithm 運算
  - 運算完成後 FPGA 透過 RS232 將答案傳回給 PC 端
  - PC 端檢查是否和軟體有一致答案

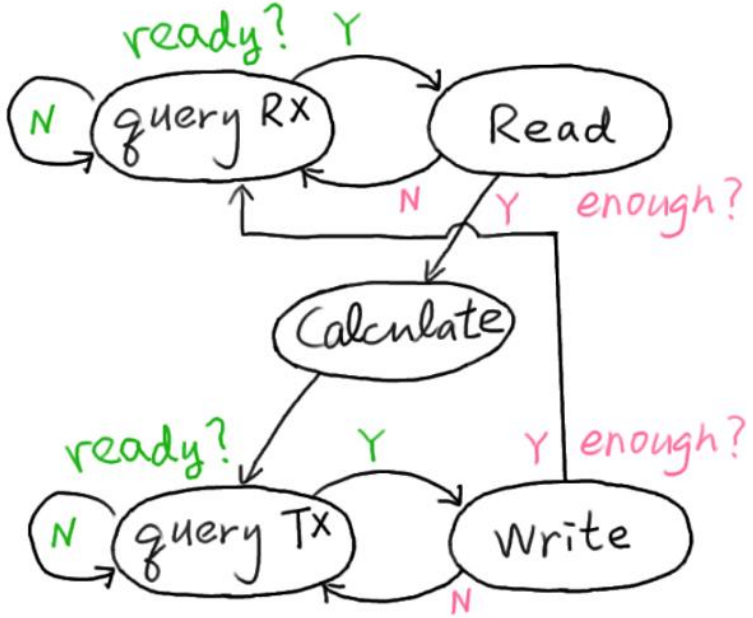
RS232

- Very old (1969) and very simple protocol
  - Only has two signal lines receiver/transmitter (RX/TX)
- But very slow (~10KB/s)
- Here, we use Qsys IP
  - Access different data by address BASE+0, 4, 8, …

Offset	Register Name	R/W	Description/Register Bits													
			15:13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO	Reserved					1	1	Receive Data						
1	txdata	WO	Reserved					1	1	Transmit Data						
2	status 2	RW	Reserved	eop	cts	dcts	1	e	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW	Reserved	ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor 3	RW	Baud Rate Divisor													
5	endof-packet 3	RW	Reserved					1	1	End-of-Packet Value						

SW_Wrapper

- 操作 Qsys 生成的 RS232 IP
  - 先讀入資料 (reference sequence & read sequence)
  - 讀取完後交給 SW_core 進行運算
  - 將答案 (null, column, row, highest score)，31bytes寫出



- 在讀寫前要先確定IP準備好了
  - 讀取BASE+8的[7]和[6] (前頁螢光筆標示處)
  - Ex: 當addr給BASE+8，readdata[7]代表RX準備情況
  - 同時要確認 avm_waitrequest 為 0 並在讀到時將 addr更改成 BASE+0

- 讀寫時每次只有8 bits
  - 所以每一筆 256b 資料要分 32 次讀
  - Ex:當addr給BASE+0，readdata[7:0]是RX送來的8b資料

Code Template

- DE2_115/
  - Design setup files
- cpp/
  - Software Code (c++) for SW algorithm and generating testing data
- pc_python/
  - Python executable test program for PC
- tb_verilog/
  - Verilog testbench for core and wrapper
- SW_Core.sv
  - Implement Smith-Waterman algorithm here
- SW_Wrapper.sv
  - Implement controller for RS232 protocol
  - Including reading check bits and read/write data

Generating Testing Data

- Move to cpp/
  > *cd ./lab_bonus_SW/src/cpp*
- Compile ./cpp/src/gen_data.cpp
  > *source compile_gen.sh*
- Generate testing data
  > *source generate_testdata.sh*
- There are two files being generated
  - ./test_data/random_pattern.txt
  - ./test_data/random_pattern.bin

in gen_data.cpp

```
56
57      srand (2);      //random seed
58      int random_num; // random number
59      std::string reference_seq, read_seq, replace_seq;
60
61      char* ref_buffer = new char[32];
62      char* read_buffer = new char[32];
63      int temp_byte = 0;
64
65      for (int i=0; i<data_num; i++){
```

You can change the random seed to generate different testing data

in generate_testdata.sh

```
$ generate_testdata.sh ×
C: > Users > dclab > Desktop > lab_bouns_SW > src > cpp > $ generate_testdata.sh
1   time ./gen.out -read_len 128 -ref_len 128 -data_num 100
2
```

Perform SW Algorithm (c code)

- Move to cpp/

  > *cd ./lab_bonus_SW/src/cpp*

- Compile ./cpp/src/main.cpp

  > *source compile_SW.sh*

- Perform SW algorithm

  > *source run_SW.sh*

- The results will be recorded in
  - ./exe_SW.log

SW_Core.sv

```
`define REF_MAX_LENGTH              128
`define READ_MAX_LENGTH             128

`define REF_LENGTH                  128
`define READ_LENGTH                 128

//* Score parameters
`define DP_SW_SCORE_BITWIDTH        10

`define CONST_MATCH_SCORE           1
`define CONST_MISMATCH_SCORE        -4
`define CONST_GAP_OPEN              -6
`define CONST_GAP_EXTEND            -1

// SW Core ----------------------------------------
module SW_core(
    input                                   clk,
    input                                   rst,

    output reg                              o_ready,
    input                                   i_valid,
    input [2*`REF_MAX_LENGTH-1:0]           i_sequence_ref,     // reference seq
    input [2*`READ_MAX_LENGTH-1:0]          i_sequence_read,    // read seq
    input [$clog2(`REF_MAX_LENGTH):0]       i_seq_ref_length,   // (1-based)
    input [$clog2(`READ_MAX_LENGTH):0]      i_seq_read_length,  // (1-based)

    input                                   i_ready,
    output reg                              o_valid,
    output signed [`DP_SW_SCORE_BITWIDTH-1:0]   o_alignment_score,
    output reg [$clog2(`REF_MAX_LENGTH)-1:0]    o_column,
    output reg [$clog2(`READ_MAX_LENGTH)-1:0]   o_row
);
```

SW_Wrapper.sv

```
`define REF_MAX_LENGTH              128
`define READ_MAX_LENGTH             128

`define REF_LENGTH                  128
`define READ_LENGTH                 128

//* Score parameters
`define DP_SW_SCORE_BITWIDTH        10

`define CONST_MATCH_SCORE           1
`define CONST_MISMATCH_SCORE        -4
`define CONST_GAP_OPEN              -6
`define CONST_GAP_EXTEND            -1

module SW_Wrapper (
    input         avm_rst,
    input         avm_clk,
    output  [4:0] avm_address,
    output        avm_read,
    input  [31:0] avm_readdata,
    output        avm_write,
    output [31:0] avm_writedata,
    input         avm_waitrequest
);
```

Hardware Simulation

- Testbench for core and wrapper are provided in tb_verilog/
- To run simulation for SW core
  > source 01_run_core.sh

- To run simulation for SW wrapper
  > source 02_run_wrapper.sh

- Use nWave to check the waveform and happy debugging
  – It is advised to test individual modules first

- Feel free to design your own testbench!

Lab Requirements (1/2)

- Input reference length: 128 bp (256 bits)
- Input read length: 128 bp (256 bits)
  - A: 00, C: 01, G: 10, T: 11

- Provided Scoring system
  - Match score: 1
  - Mismatch score: -4
  - Gap open penalty: -6
  - Gap extension penalty: -1

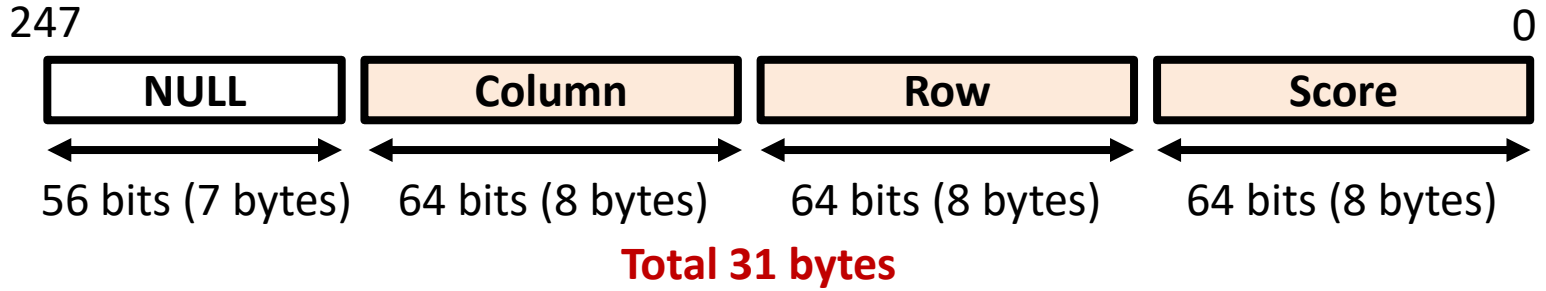
- Output Results
  - Highest score
  - Row
  - Column

```
`define REF_MAX_LENGTH          128
`define READ_MAX_LENGTH         128

`define REF_LENGTH              128
`define READ_LENGTH             128

//* Score parameters
`define DP_SW_SCORE_BITWIDTH    10

`define CONST_MATCH_SCORE       1
`define CONST_MISMATCH_SCORE    -4
`define CONST_GAP_OPEN          -6
`define CONST_GAP_EXTEND        -1
```



Lab Requirements (2/2)

- Key0 可以 reset
- Pass all patterns
  - 100 public patterns
  - 5 hidden patterns (Demo 當天會公布)
- Demo 時間: **12/01 (四) 12:20 – 13:10**
  - 記得提前一個禮拜和助教預約是否要 demo
- Bonus (請在 demo 時和 report 中詳細說明)
  - 能輸出 backtracking alignment results
  - 不需要按 reset 即可連續運算多份 patterns
  - 其他想得到的變化
- Materials
  - https://github.com/sandy30538/NTUEE-DCLAB-Materials.git

System Setup

Testing Program

- Environment setup
  - Install Python2
  - ez_setup.py (https://pypi.org/project/setuptools/) or
    sudo apt-get install python-pip
  - (sudo) pip install pySerial
- Usage
  - Copy **random_pattern.bin** and **pattern_ans.txt** next to the executable
  - ./test_rs232.py [COM? | /dev/ttyS0 | /dev/ttyUSB0]
- Several testing data are provided
- Flow
  - Send 32-Byte reference sequence
  - Send 32-Byte read sequence
  - FPGA perform SW algorithm
  - Receive 31-Byte result (col, row, score)

Submission

- 繳交項目
  - Report(.pdf)
  - source code(.v)
- 繳交檔案架構

```
team01_lab1
|-team01_lab1_report.pdf
|-src
  |-<all of your verilog code>.v
```

- **先創一個teamXX_lab_bonus的資料夾再壓縮，不要直接壓縮繳交項目**

Report Regulation

- 內容應包含
  - 層級架構
  - Block Diagram (必須包含 Data Path，control signal可有可無)
  - FSM or Scheduling
  - Fitter Summary 截圖
  - Timing Analyzer 截圖
  - 問與答
    - 1. Short-read mapping 的流程為何？
    - 2. 以軟體實現 SW 演算法的挑戰是什麼？
    - 3. 以硬體實現的優勢是甚麼？

- 一組交一份，以pdf檔繳交
- 命名方式：teamXX_lab_bonus_report.pdf
  - Ex: team01_lab_bonus_report.pdf
- 繳交期限：demo隔天午夜