COMP9319 Web Data Compression and Search

a1 results & considerations,

a2 tips

The most important slide

Raymond, please **“RECORD THIS LECTURE!”**

a1

Assumed knowledge

Official prerequisite of this course is COMP2521 / COMP1927 / COMP9024.

At the start of this course students should be able to:

- understand bit and byte operations in C/C++.
- write C/C++ code to read from/write to files or memory.
- produce **correct** programs in C/C++, i.e., compilation, running, testing, debugging, etc.
- produce readable code with clear documentation.
- appreciate use of abstraction in computing.

Produce a **correct** program

1. Understand the requirements
2. Coding
3. Testing & debugging

1. Understand the requirements (with *perf req*)
2. Coding
3. *Performance tuning*
4. Testing & debugging

In the past, students failed assigts because:

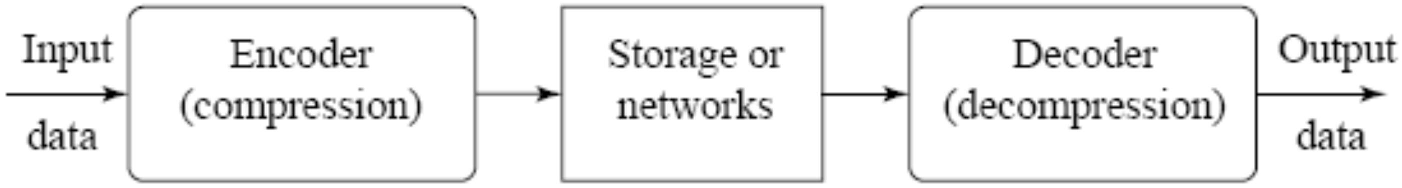
- *Plagiarism*
- Code failed to compile due to various reasons
- Program worked on windows but not CSE linux
- Late submission
- Program did not follow the spec
- Program failed auto-marking

For all assessments:

- marks granted based on correctness
- not based on efforts spent

Overview

- Compression refers to a process of coding that will effectively reduce the total number of bits needed to represent certain information.



- There are two main categories
  - Lossless (Input message = Output message)
  - Lossy (Input message ≠ Output message)

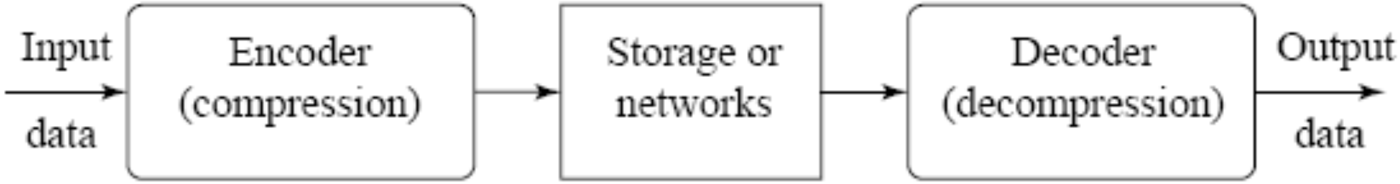
Your a1

- Can be easily converted into a real AC encoder & decoder

Any data incl binaries

Manipulated & output in binary instead of decimal

The original data (whatever they are)



Note: You'll need to work on the precision for files of any size.

10 autotest cases

1. The very naive case: BILL GATES
2. Same a1 but diff order: BILL STAGE
3. Decimal for decimal
4. A small alphabet set: A,C,G,T
5. Ordinary text 1kb: extracted fr the spec
6. Extracted fr a popular story (Gutenberg)
7. Decimal places for lo
8. Decimal places for hi
9. Calculation correctness (the prefix)
10. Modified input AC value to the decoder

10 autotest cases

Precision:

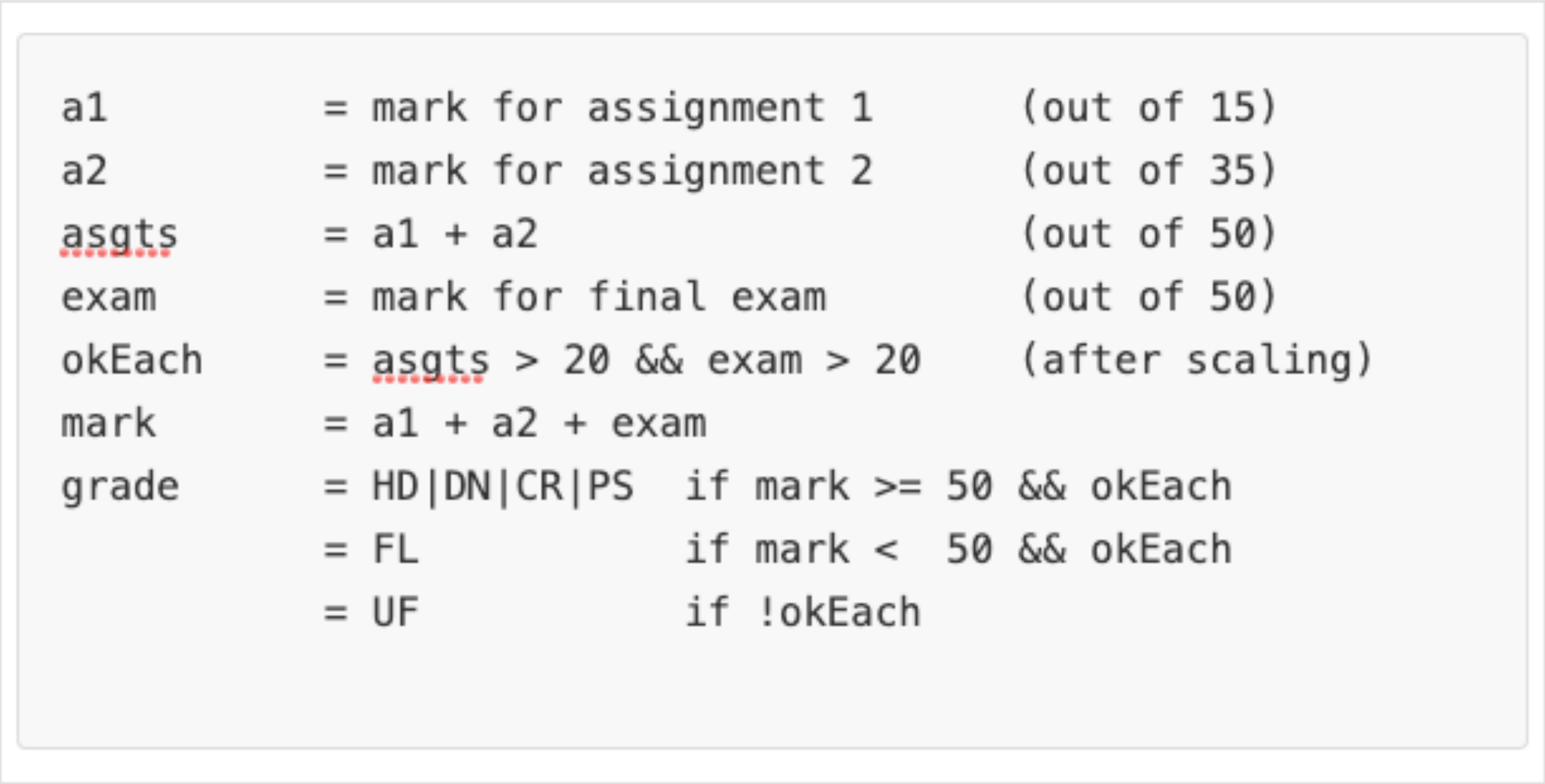
lo = 0.2572167752

hi = 0.2572167756

lo = 0.2572167751999999999999999968

hi = 0.2572167756000000000000000113

Assessment



```
a1        = mark for assignment 1      (out of 15)
a2        = mark for assignment 2      (out of 35)
asgts     = a1 + a2                    (out of 50)
exam      = mark for final exam        (out of 50)
okEach    = asgts > 20 && exam > 20    (after scaling)
mark      = a1 + a2 + exam
grade     = HD|DN|CR|PS  if mark >= 50 && okEach
          = FL           if mark <  50 && okEach
          = UF           if !okEach
```

Special considerations for a1

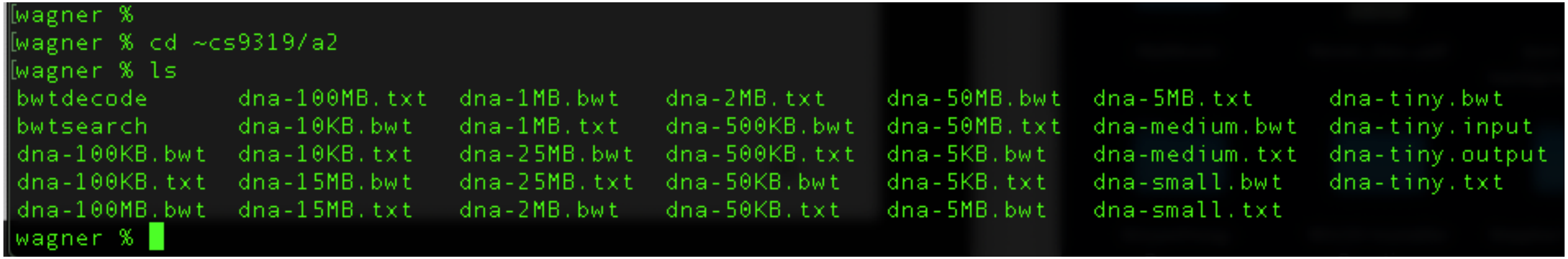
- *Assume your code is well documented or very readable*
- If final = (a1 + a2 + exam) < 50 then:
    - a1 will be re-marked by more tests and/or reading your code, max achievable final = 50

Special considerations for a1

- *Assume your code is well documented or very readable*
- Motivations to learn & try harder next time:
- If your (a2 + exam) is in the third quartile (Q3), i.e., the middle value between the median and the highest value of (a2+exam), then
  - the wrong tests of a1 will be re-marked by more tests and/or reading your code, max achievable = 80% of those tests

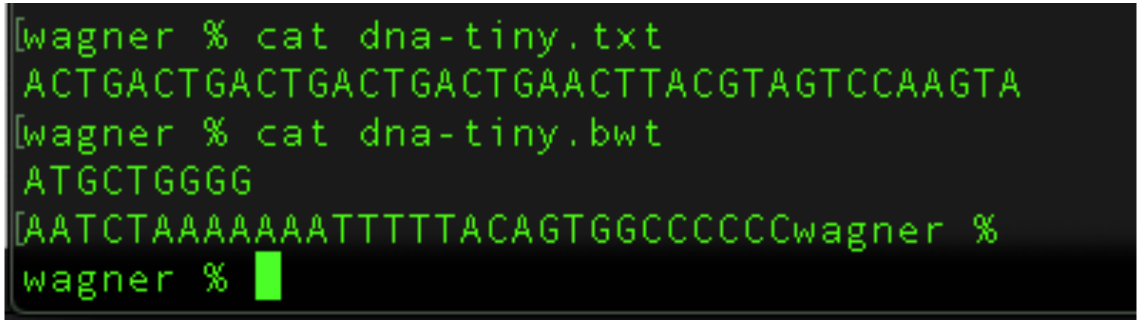
a2

a2



```
[wagner %
[wagner % cd ~cs9319/a2
[wagner % ls
bwtdecode      dna-100MB.txt  dna-1MB.bwt   dna-2MB.txt    dna-50MB.bwt  dna-5MB.txt     dna-tiny.bwt
bwtsearch      dna-10KB.bwt   dna-1MB.txt   dna-500KB.bwt  dna-50MB.txt  dna-medium.bwt  dna-tiny.input
dna-100KB.bwt  dna-10KB.txt   dna-25MB.bwt  dna-500KB.txt  dna-5KB.bwt   dna-medium.txt  dna-tiny.output
dna-100KB.txt  dna-15MB.bwt   dna-25MB.txt  dna-50KB.bwt   dna-5KB.txt   dna-small.bwt   dna-tiny.txt
dna-100MB.bwt  dna-15MB.txt   dna-2MB.bwt   dna-50KB.txt   dna-5MB.bwt   dna-small.txt
wagner % 
```

a2



```
[wagner % cat dna-tiny.txt
ACTGACTGACTGACTGACTGAACTTACGTAGTCCAAGTA
[wagner % cat dna-tiny.bwt
ATGCTGGGG
[AATCTAAAAAAATTTTTACAGTGGCCCCCCwagner %
wagner % █
```

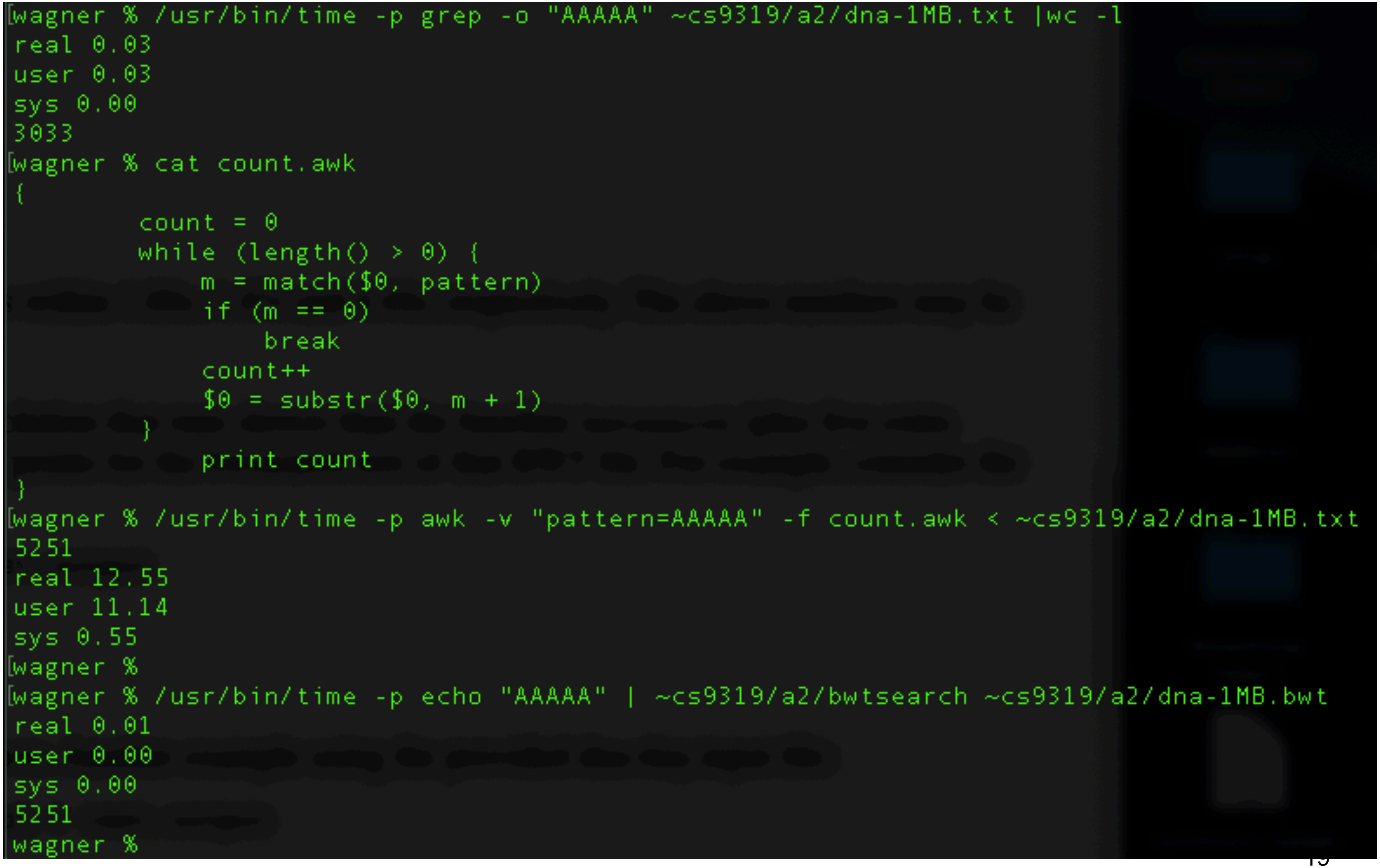
a2

T = banana
P = ana

Overlapping?

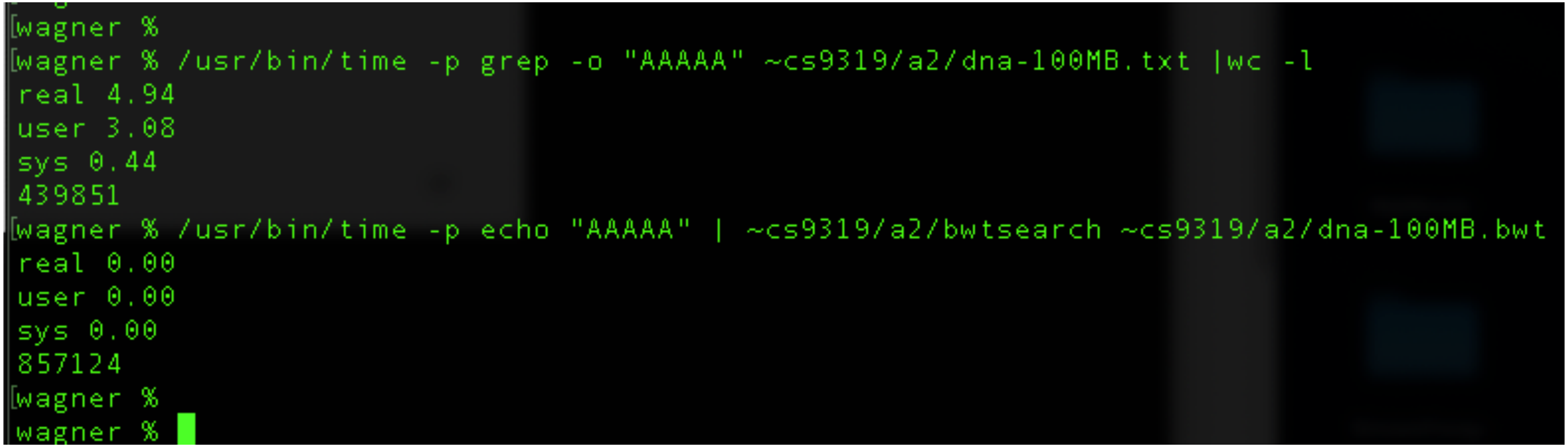
banana

grep is fast (BM), but no overlap matches



```
wagner % /usr/bin/time -p grep -o "AAAAA" ~cs9319/a2/dna-1MB.txt |wc -l
real 0.03
user 0.03
sys 0.00
3033
wagner % cat count.awk
{
        count = 0
        while (length() > 0) {
            m = match($0, pattern)
            if (m == 0)
                break
            count++
            $0 = substr($0, m + 1)
        }
            print count
}
wagner % /usr/bin/time -p awk -v "pattern=AAAAA" -f count.awk < ~cs9319/a2/dna-1MB.txt
5251
real 12.55
user 11.14
sys 0.55
wagner %
wagner % /usr/bin/time -p echo "AAAAA" | ~cs9319/a2/bwtsearch ~cs9319/a2/dna-1MB.bwt
real 0.01
user 0.00
sys 0.00
5251
wagner %
```

grep is fast (BM), but still far from backward search



```
wagner %
wagner % /usr/bin/time -p grep -o "AAAAA" ~cs9319/a2/dna-100MB.txt |wc -l
real 4.94
user 3.08
sys 0.44
439851
wagner % /usr/bin/time -p echo "AAAAA" | ~cs9319/a2/bwtsearch ~cs9319/a2/dna-100MB.bwt
real 0.00
user 0.00
sys 0.00
857124
wagner %
wagner %
```

Memory

- Don’t call unnecessary functions / use unnecessary libraries
- Don’t allocate too much
- Release them when they’re not needed

Speed

- File I/Os dominate the time
- For reversing, ideally, max 1 read per decoding char
- For search, ideally, max 2 reads per search term char
- Also, minimize the size per read

Other questions?