COMP10002 Workshop Week 3

Outlook:

1	Asymptotic Complexity
2	arguments of main()
3	char and strings, discuss 7.12, 7.14, 7.15, 7.16
4	Implement at least two from 7.12 7.14 7.15, 7.16 OR groupwork

Big-O (informal)

Equivalent writings:

1. $f(n) \in O(g(n))$
2. $f(n)$ grows no faster than $g(n)$
3. there is a constant $c > 0$ such that $c.g(n)$ is an upper bound of $f(n)$ when n is big enough

Strictly speaking:

$$2n + 3 \in O(n) \in O(n \log n) \in O(n^2) \in \ldots$$

But ***in computer science***, when using big-O we often mean ***the least upper bound***. So we only say:

$$2n + 3 \in O(n)$$

(Informal) Big-O Rules

Multiplicative constants can be reduced to 1:

$1000n^2$ or $0.0000001n^2$ is just n^2.

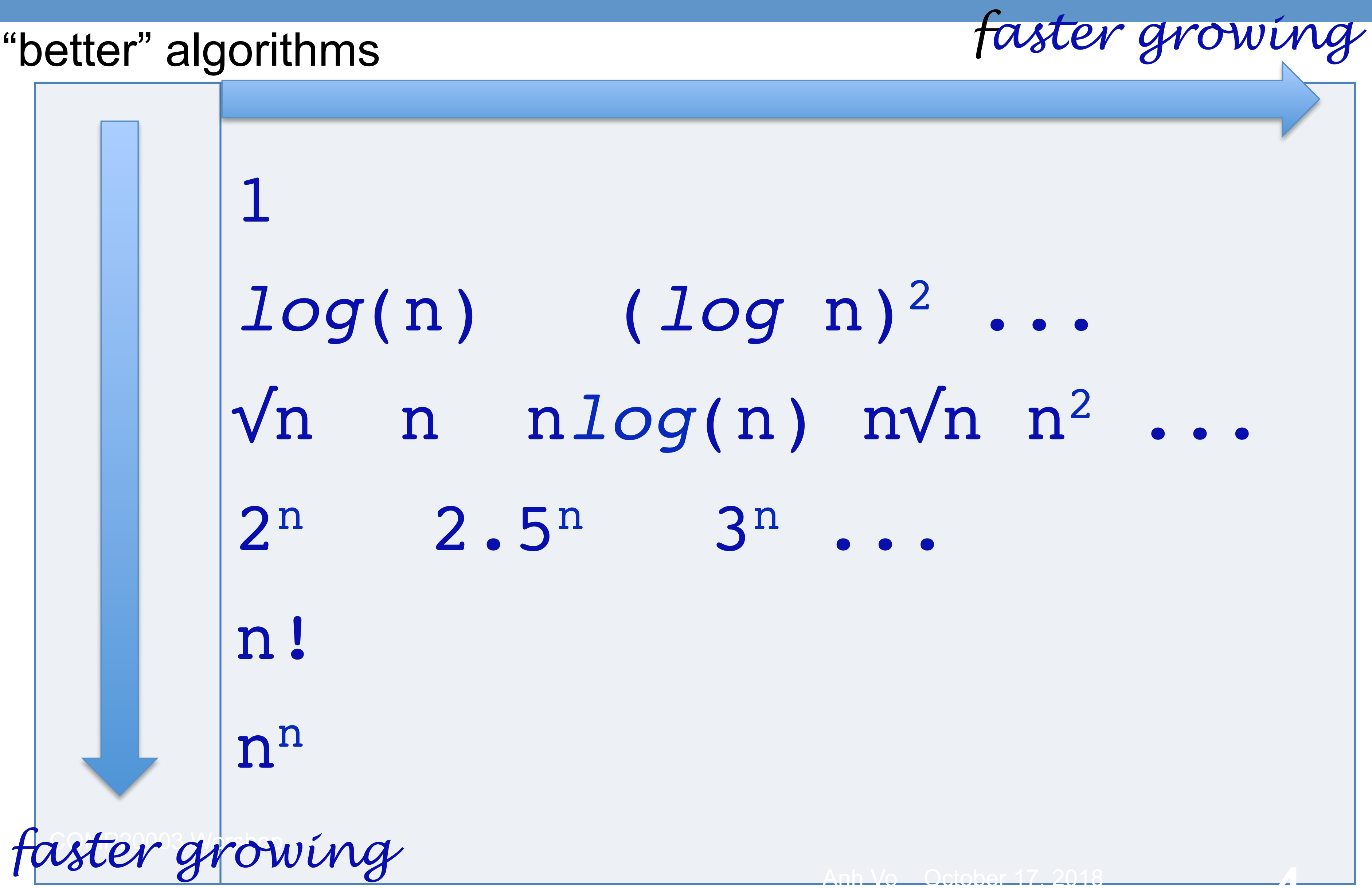
Base of logarithm doesn't matter:

$log_{10}n$ or log_2n is just $log\ n$

Lower-level additive parts can be omitted:

$2n^3 + 100000n^2 + 6n + 10^{12}$ is just n^3.

Remember:



"better" algorithms

faster growing

1

$\log(n)$ $(\log n)^2$ ...

$\sqrt{n}$ n $n\log(n)$ $n\sqrt{n}$ n^2 ...

2^n 2.5^n 3^n ...

$n!$

n^n

faster growing

Examples

Find big-O for:

a) $0.001n^3 + 10^9n^2 + 1$

b) $25\ (\log n)^5 + n + 100$

c) $n^{0.1} + (\log n)^{10}$

d) $(\log n)^3 + \sqrt{n}$

Program arguments

Write a program sum that accept two numbers and print out their sum. Example of execution:

```
$ ./sum 12 5
12.00 + 5.00 = 17.00
```

```
?: int main(int argc, char *argv[])
```

Program arguments: notes

Must check:

is the number of arguments as expected, and

when possible, is each argument valid.

Example: a program that accepts two positive numbers and print out their sum.

```
int main(int argc, char *argv[]) {
   double a, b;
   if (    argc != 3
        || (a=atof(argv[1]))<=0
        || (b=atof(argv[2]))<=0   ) {
      fprintf(stderr, “Usage: %s a b  where”
                “ a>0 and b>0\n”,argv[0]);
      exit(EXIT_FAILURE);
   }
   printf(“%.2f + %.2f = %.2f\n”, a, b, a+b);
   return 0;
}
```

char and <ctype.h>

```
int c;
c= getchar();
```

How to check if c is:

- a lower-case letter?

```
 if (         ?               )
     printf ("a lower-case letter!\n";
```

- an upper-case letter?

```
if (        ?          ) ...
```

- or just a letter?

```
if (            ?             ) ...
if (        ?          ) ...
```

char and <ctype.h>

```
int c;
c= getchar();
```

How to check if c is:

- a digit?

```
 if (          ?              )
     printf ("a digit!\n");
 if (          ?              ) ...
```

- a letter or digit

```
if (          ?              ) ...
```

Strings and <string.h>

```
char s1[10]= “Hello”;
char *s2=“1234”;
```

Which of the following statements are OK:

```
1. *s2 = “A”;
2. s1 = ‘A’;
3. s1 = “ABBA”;
4. printf(“%d\n”, strlen(s1+1));
5. printf(“%d\n”, strlen(s2));
6. strcpy(s2, s1);
7. strcpy(s1, s2);
8. s2[6]= ‘\0’;
```

Strings

```
char s1[10]= “Hello”;
char *s2=“1234”;
```

Which of the following statements are OK:

1. `s1++;`
2. `s2++;`
3. `s1= s2;`
4. `s2= s1 + 3;`

    (if Ok, then what happens to “`1234`” ?)

Strings

```
typedef char word_t [11];
char *s=“1234 abc9”, *p= s;
word w; int i= 0;
```

Which of the following fragments makes `w` be “`abc`”:

```
1. while (*p) w[i++]= *p++;
2. while (isalpha(*p)) w[i++]= *p++;
3. while (!isalpha(*p)) p++;
  while (isalpha(*p)) w[i++]= *p++;
4.  none of them
```

Strings

```
char *s=“123”;
int n;
```

Which of the following fragments give the same result as `n=atoi(s)`:

```
1. for (; isdigit(*s); s++)
        n= n*10 + (*s);
2. for (n=0; isdigit(*s); s++)
        n= n*10 + (*s)
3. for (n=0; *s && isdigit(*s); s++)
        n= n*10 + (*s);
4. none of the above
```

Case Study & Ex 7.16 – The Task

Design and implement a program that reads text from stdin, and writes a list of the distinct words that appear, together with their frequencies.

First step:

Make sure you understand the task, that you can imagine what's the input and output.

Case Study & Ex 7.16 – Understanding The Task

Design and implement a program that reads text from stdin, and writes a list of the distinct words that appear, together with their frequencies.

Sample texts:

```
A cat in a hat!
+-abc 10e12 e 1abc #e#abc.abcdefghijklm=xyz
```

Output= ?

Assumptions/limits:

- Word=
- ? what else?
- ?

Case Study & Ex 7.16 - Design

What’s input? What’s the best (or just feasible) way to get it?

Case Study & Ex 7.16 - Design

How to store output, which data structure?

And how to produce output?

Case Study & Ex 7.16 – Alistair's getword

```
int getword(char W[], int limit) {
  int c, len=0;
  /* first, skip over any non alphabetics */
  while ((c=getchar()) != EOF && !isalpha(c)) {
      /* 12+34 do nothing */
  }
  if (c==EOF)  return EOF;

  /* ok, first character of next word has been found */
  W[len++] = c;
  while (len<limit && (c=getchar())!=EOF && isalpha(c)) {
    /* 12+34 another character to be stored */
    W[len++] = c;
  }
  /* now close off the string */
  W[len] = '\0';
  return 0;
}
```

Ex 7.16 and others

Combine Alistair's `getword.c` and `words.c` into one `.c` file, then change it to meet the requirement of Ex 7.16.

Implement

- 7.12 (medium),
- 7.14 (easy),
- 7.15 (a bit hard).

Another choice: group work - doing 7.12, 7.14 or some other exercises on board/paper