# COL100 Lecture 18

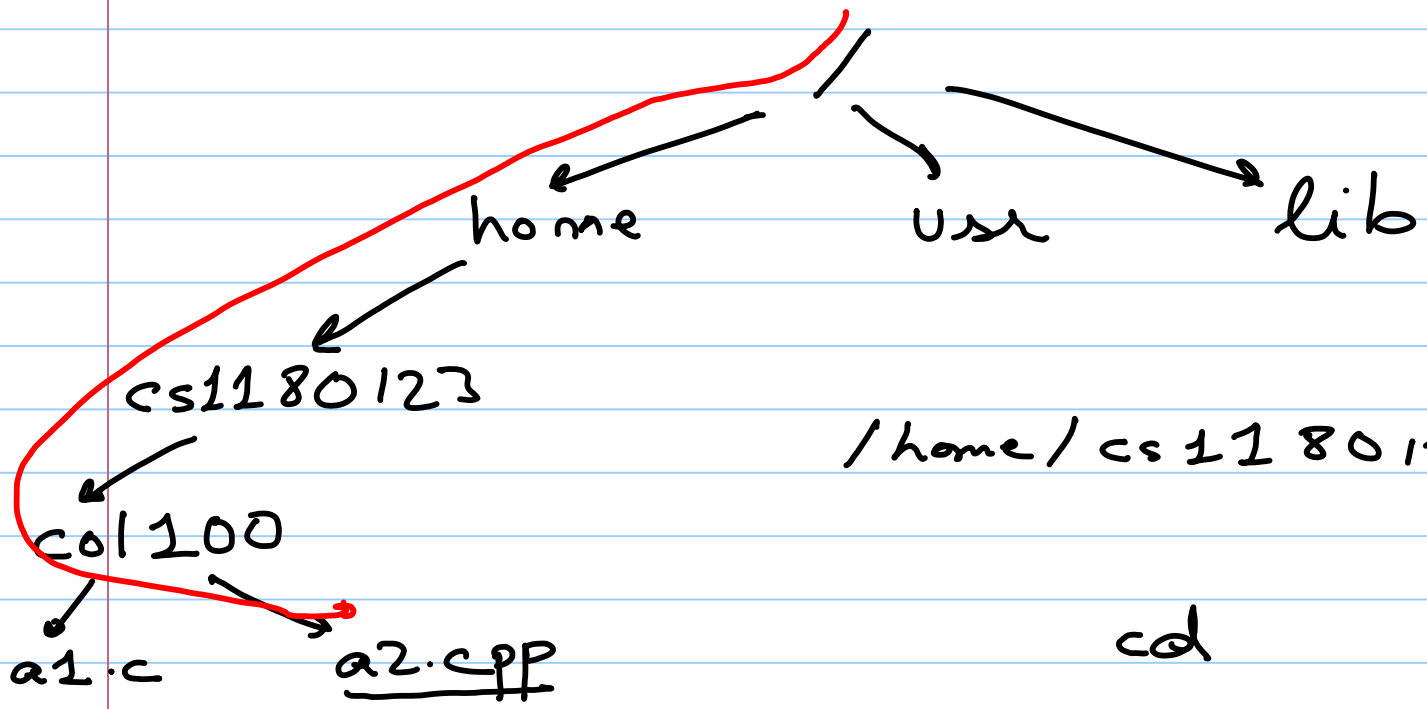
Note Title 9/27/2018

Review:

## File I/O

```
ifstream infile;
if(infile.open("filename.txt");
        infile.get(ch);
        infile >> word
        getline
infile.close();
```

# Filenames




/home/cs1180123/col100/a22.cpp

cd

open can fail.

what if it fails?

```
if ( infile.open("filename") )
{
    . . .
    . . .
    . . .
}
```

expr with side effects

# Output file streams

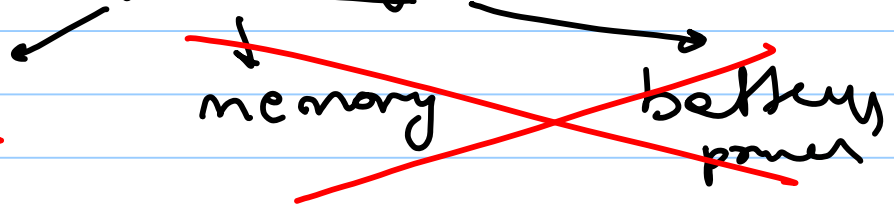
```
ofstream ofile;
ofile.open("filename.txt");
string word = "output";
int x = 3;
outfile << word << "=" << x << endl;
outfile.close();
```

filename.txt:

Output␣3\n

# Efficiency

- Lots of ways to solve a problem: some are more efficient
- How to measure efficiency




Programs

- Algorithms are better if they take less time

## Unit of time

→ We count the number of units of time and get the runtime

if input size = $N$, then program takes ↓ time

```
statement 1 ;   // eg. int x = 0      { 1 unit
statement 2 ;                           1 unit

for (int i = 0; i < N; i++)                          1 unit
{                  ↑ 1 unit   ↑ 1 unit
    statement 3;                                     1 unit
}
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < N; j++)
    {
        statement 4;                                 1 unit
    }
}
```

N units or more precisely 3N+1 units

$N^2$

N

Total runtime (roughly):

$$N^2 + (3N + 1) + 2$$

$$N^2 + N + 1$$ : ignore all constants

↓ only the biggest powers of N matter

$$\underbrace{N^2}$$

this is the "dominating" term

Another way to think about this:

$$N^2 + N + 1 < 2N^2$$

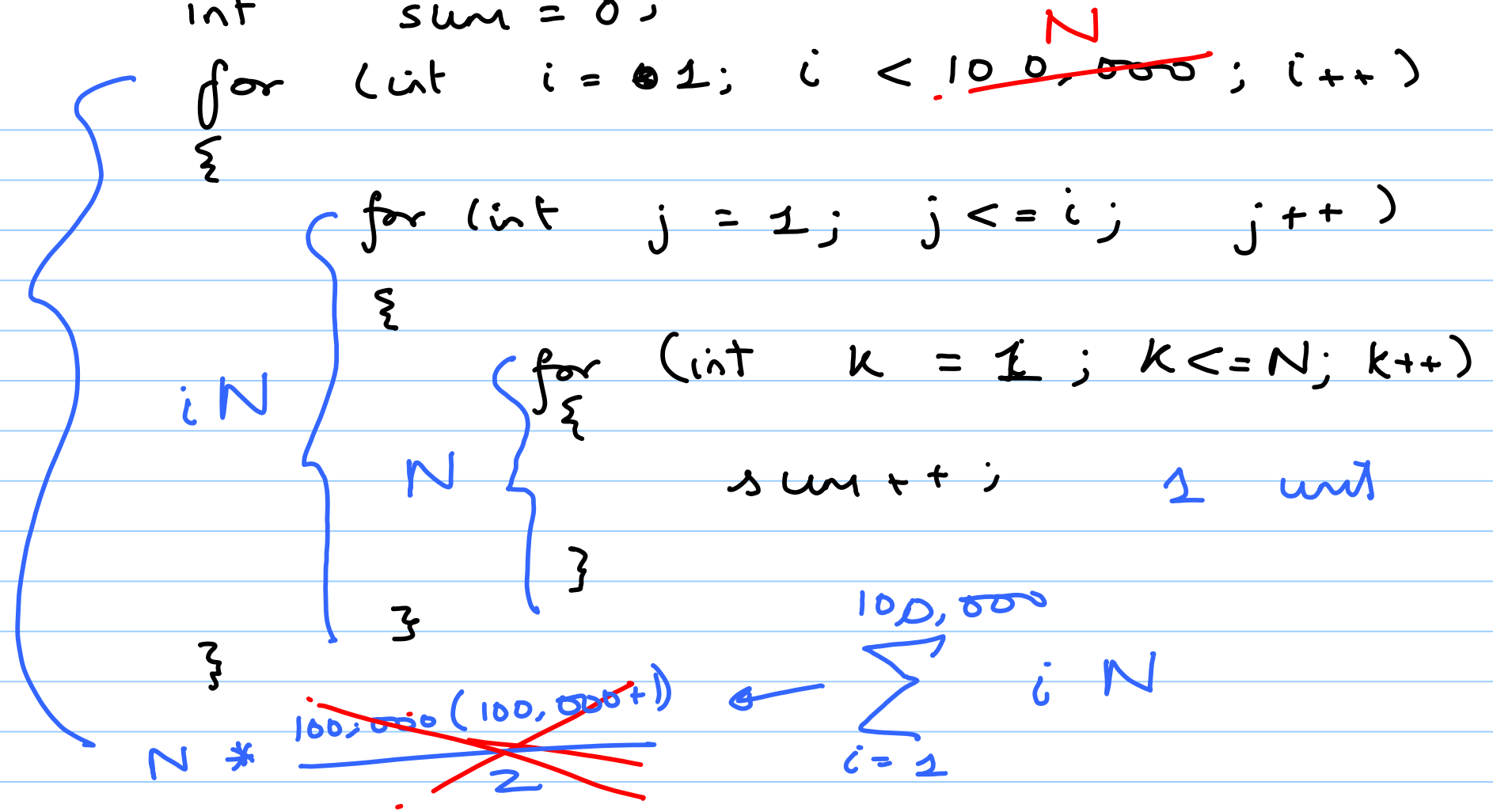
We will say that the code snippet has $O(N^2)$ runtime. → Big-O notation

order $- N^2$

Big O $- N^2$ $(O - N^2)$

# Finding Big-O ~~notation~~

- Work from the inner-most nesting level code
- Nested code multiplies
- Code at same nesting level adds


int sum = 0;
for (int i = 1; i < 100,000; i++)
{
for (int j = 1; j <= i; j++)
{
for (int k = 1; k <= N; k++)
{
sum++; 1 unit
}
}
}
i N
N
N * 100,000 (100,000+1) / 2
100,000
∑
i = 1
i N

$$\sum_{i=1}^{N} i \; N$$

$$\frac{N\,(N+1)}{2} \times N$$

$$O(N^3)$$

1 [ Vector<int> v;

$\sum_{i=1}^{N/2} i$ [ for (int x=1; x<=N; x+=2)
{
v.insert(0, x); v.length() mitad
}

N [ cout << v << endl;

$O(N^2)$

## Complexity class

a category of algorithmic efficiency based on the of algorithm's relationship to the input size $N$

~~O(1)~~

constant

O(1)