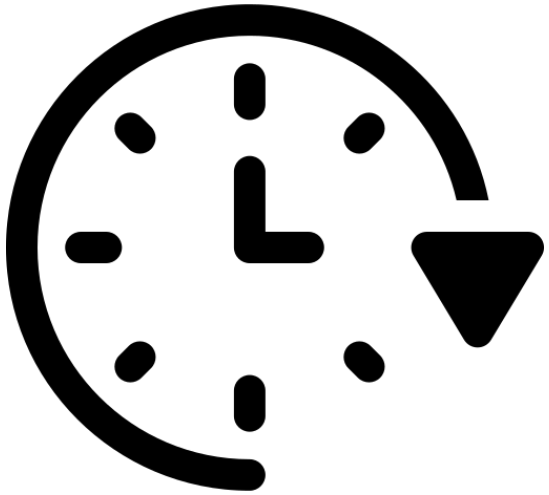
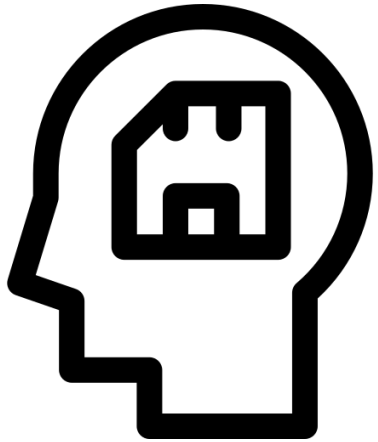
Complexity Analysis

Program A vs Program B

ถ้าโปรแกรม A กับ B ให้ Output เดียวกัน เราจะรู้ได้ไงว่าอันไหนดีกว่ากัน





Performance measurement

	Time	Memory
Program A	26 ms	1 MB
Program B	100 ms	256 KB

How to measure time/memory?

- To measure memory, we must know…
  - ???
- To measure time, we must know….
  - ???
  - ???
  - ???

It’s too hard!!!!

Complexity

- Big O notation
- Big Theta notation

Big Theta

If g(x) = Theta(f(x))

- There exist c, d and y that makes g(z) <= d * f(z) and g(z) >= c * f(z) for every z >= y

Example

- $f(x) = x^2 + 2x$ -> theta(???)
- $f(x) = x + \log(x)$ -> theta(???)
- $f(x, y) = x^2 + y$ -> theta(???)

Big theta for this function

```
int triangle_number(int n) {
    int sum = 0;
        for (int i = 1; i <= n; i++) {
                sum += i;
        }
        return sum;
}
```

Big theta for this function

```
\\สมติว่าเราใช้ triangle_number ข้างบน
int func(int n) {
    int total = 0;
    for (int i = 1; i <= n; i++) {
            total += trianble_number(i);
    }
}
```

Is Big Theta notation enough?

```
bool lucky (int n) {
    for (int i = 1; i <= n; i++) {
                if (rand() == 99) {
                        return true;
                }
        }
        return false;
}
```

Big O notation: The worst case

If g(x) = Theta(f(x))

- There exist d and y that makes g(z) <= d * f(z) for every z >= y

Big O for this function

```
bool lucky (int n) {
    for (int i = 1; i <= n; i++) {
                if (rand() == 99) {
                        return true;
                }
        }
        return false;
}
```

Time approximation (from my experience)

- Find O(f(…))
- Approximate max f(…)
- 10,000,000 / sec for average
- Can be faster/slower.

Big O is not everything

```
// n <= 10'000
int birthday_paradox(int n) {
    bool found[n];
        for (int i = 0; i < n; i++)
                found[i] = false;
        for (int i = 1; i <= n; i++) {
                int x = rand() % n;
                if (found[x]) return i;
                found[x] = true;
        }
        return n + 1;
}
```