CS 180 Discussion 1A/1E

Haoxin Zheng

10/06/2023

Announcements

- Homework 1 has been posted. Due on **11:59 PM, Oct. 11th**. *No late submission accepted.*

- The classes on **Oct. 12th** and **Dec. 5th** will be on Zoom. Location is the same as the professor's office hour Zoom link.

- Check updated syllabus on Bruinlearn.

What is Algorithm?

- How to think strategically about a problem and come up with a systemic approach to tackling it.

- An algorithm of a problem, should be a universal approach (for any X, return Y) , regardless of specific inputs.

- Key points to evaluate an algorithm:
  - 1) Does this approach work in all cases?
  - 2) Is this the most efficient approach?
    - What constraints do we have to consider? What conditions are we given?

Example

- Drawbacks?

```python
# my_first_calculator.py by AceLewis
# TODO: Make it work for all floating point numbers too

if 3/2 == 1:  # Because Python 2 does not know maths
    input = raw_input  # Python 2 compatibility

print('Welcome to this calculator!')
print('It can add, subtract, multiply and divide whole numbers from 0 to 50')
num1 = int(input('Please choose your first number: '))
sign = input('What do you want to do? +, -, /, or *: ')
num2 = int(input('Please choose your second number: '))

if num1 == 0 and sign == '+' and num2 == 0:
    print("0+0 = 0")
if num1 == 0 and sign == '+' and num2 == 1:
    print("0+1 = 1")
if num1 == 0 and sign == '+' and num2 == 2:
    print("0+2 = 2")
if num1 == 0 and sign == '+' and num2 == 3:
    print("0+3 = 3")
if num1 == 0 and sign == '+' and num2 == 4:
    print("0+4 = 4")
if num1 == 0 and sign == '+' and num2 == 5:
    print("0+5 = 5")
```

https://github.com/AceLewis/my_first_calculator.py/blob/master/my_first_calculator.py

Definitions (Famous person problem)

- **Given n people.**
- **“know”:** defined on two different people.
- **Famous person:** a person who satisfy
  - a) the person doesn’t know anyone
  - b) all other people know the person
- **Question**: Design an algorithm to find the famous person.
  - Operation: Ask A if you know B. Each operation cost time of 1

Algorithm 1

- 1) We ask everyone in the class if you know A – **(n-1)**
- 2) Then as A if A knows all the other people – **(n-1)**
- 3) We do this for everyone, meaning repeat 1)&2) **n** times
- 4) Total cost: **n*2(n-1)**

- Q: What operations are unnecessary?

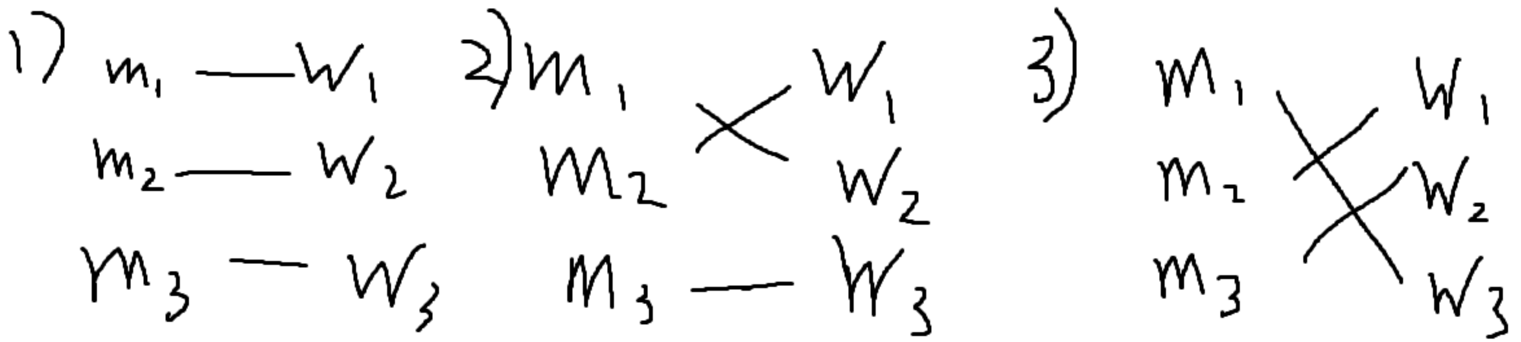
Algorithm 2 (better)

- 1) We can know there can only be 0/1 famous person.
  - Proof? What we can learn from this?
- 2) Result of question "If A knows B" can only be Yes / No
  - a) If yes, then A cannot be the famous person
  - b) If no, then B cannot be the famous person
- 3) Keep the one could be famous, continue until one person X left – **(n-1)**
- 4) Now the X is the only candidate who could be a famous person.
- 5) Ask if X knows everyone else – **(n-1)**
- 6) Ask if everyone else knows X – **(n-1)**
- 7) In total: **3(n-1)**

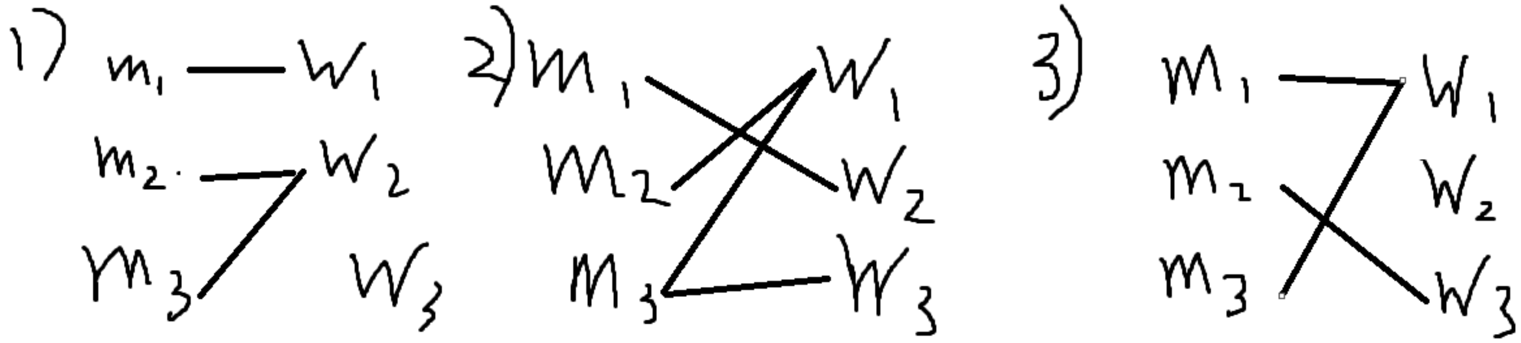
Definitions (Stable matching problem)

- 1) **Given n men and n women**: M={$m_1, m_2, \ldots, m_n$}, W={$w_{1,} w_2, \ldots, w_n$}
- 2) **Complete matching** : An 1-1 mapping between men and women.

Complete matching:



NOT complete matching:



Definitions (Stable matching problem)

- 1) **Given n men and n women**: M={$m_1, m_2, \ldots, m_n$}, W={$w_{1,} w_2, \ldots, w_n$}
- 2) **Complete matching** : An 1-1 mapping between men and women.
  - Notation: Each matching is notated as:

$m_1 - w_1$

$m_2 - w_2$

$\{(m_1, w_1), (m_2, w_2)\}$

Definitions (Stable matching problem)

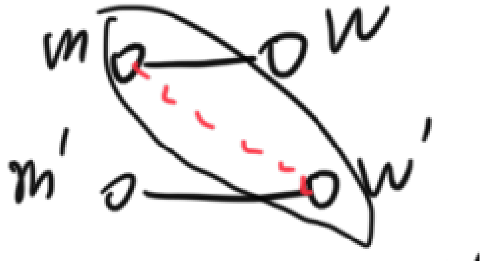
- 1) **Given n men and n women**: M={$m_1, m_2, \dots, m_n$}, W={$w_{1,} w_2, \dots, w_n$}
- 2) **Complete matching** : An 1-1 mapping between men and women.
- 3) **Preference list:** Each m_i(i=1,2,…n), w_j(j=1,2,…n) has its preference list.
  - E.g. n=3:

m1	$w_3 > w_2 > w_1$
m2	$w_2 > w_3 > w_1$
m3	$w_3 > w_2 > w_1$

w1	$m_1 > m_2 > m_3$
w2	$m_2 > m_1 > m_3$
w3	$m_3 > m_2 > m_1$

Definitions (Stable matching problem)

- 4) A complete matching S is **unstable** if $\exists\ \{(m,w),(m',w')\} \in S$ such that
  - m prefers w' to w,
  - w' prefers m to m',
  - $\boldsymbol{(m, w')}$ **called unstable edge**

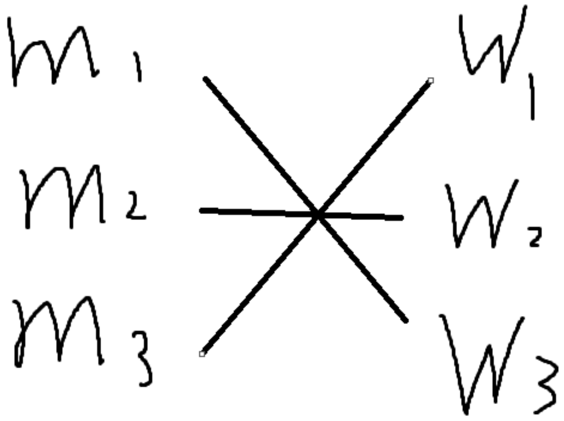


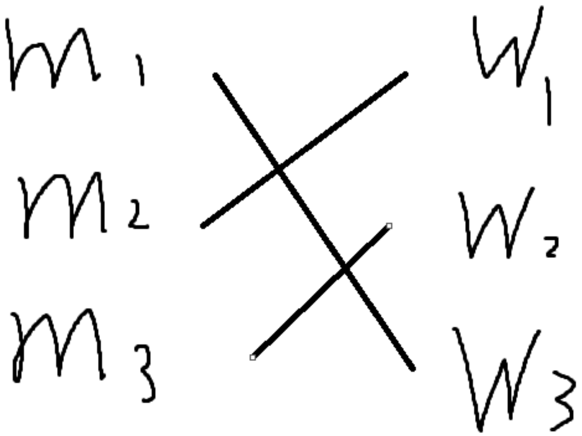
- 5) A complete matching S is **stable (Stable Matching)** if there isn't any **unstable edge**

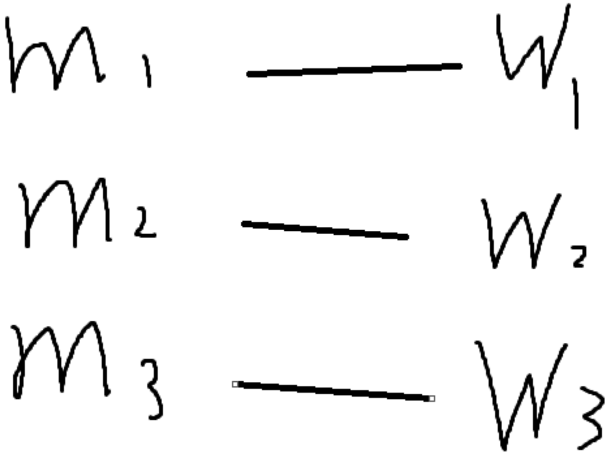
Examples

m1	$w_3 > w_2 > w_1$
m2	$w_2 > w_3 > w_1$
m3	$w_3 > w_2 > w_1$

w1	$m_1 > m_2 > m_3$
w2	$m_2 > m_1 > m_3$
w3	$m_3 > m_2 > m_1$







Gale-Shapley Algorithm (GS Algorithm)

- S = ∅
- While ∃ unmatched m:
  - Find this m (arbitrarily)
  - Find the highest-rank w for m to whom m has not proposed to.
  - If w is unmatched:
    - Add (m, w) to S
  - Else:
    - If w prefers m to the current partner (m'):
      - replace (m', w) by (m, w) in S.
    - Else:
      - w rejects m

Example (GS Algorithm)

m1	$w_1 > w_2 > w_3$
m2	$w_1 > w_3 > w_2$
m3	$w_2 > w_1 > w_3$

w1	$m_3 > m_2 > m_1$
w2	$m_2 > m_1 > m_3$
w3	$m_1 > m_3 > m_2$

m_1 W_1

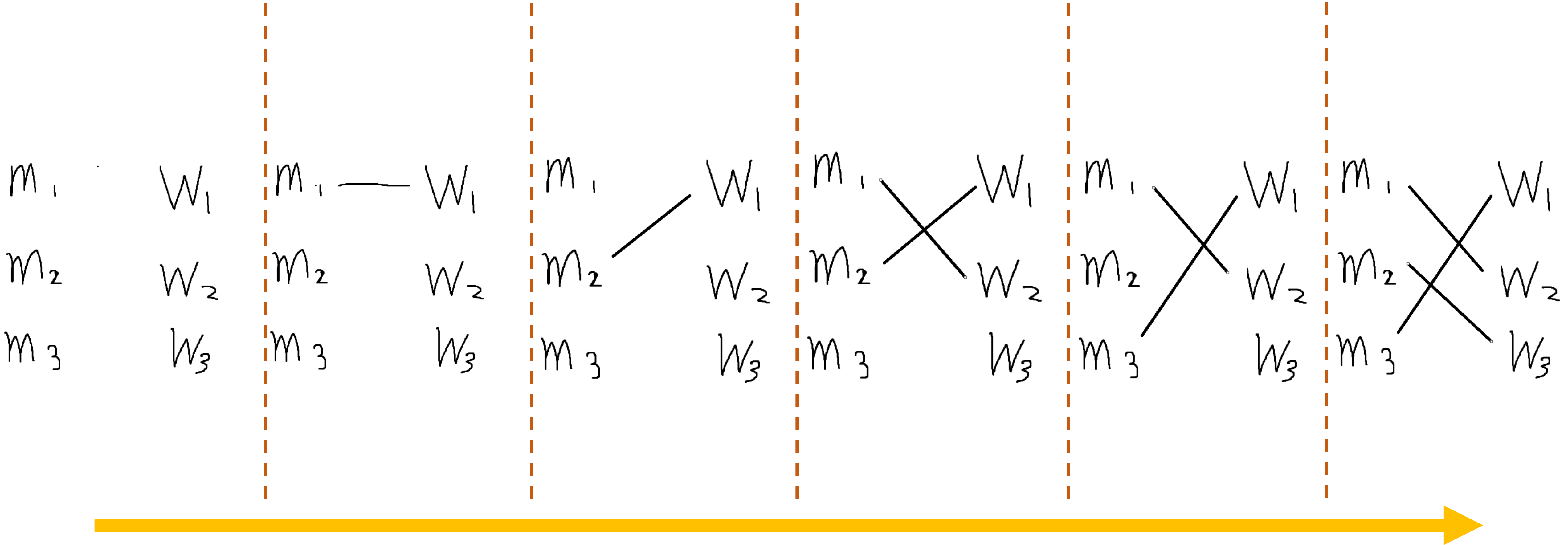
m_2 W_2

m_3 W_3

Example (GS Algorithm)

m1	$w_1 > w_2 > w_3$
m2	$w_1 > w_3 > w_2$
m3	$w_2 > w_1 > w_3$

w1	$m_3 > m_2 > m_1$
w2	$m_2 > m_1 > m_3$
w3	$m_1 > m_3 > m_2$



Observations

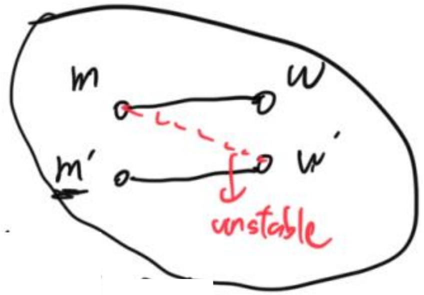
- 1. Once w is matched, she will never become unmatched.
- 2. w’s partner’s matching will always increase
- 3. m’s partner’s matching will always decrease

Theorem 1

- 1. GS algorithm will always output a **complete matching S:**
  - Proof (by contradiction):
    - **We want to show all m’s and w’s are matched**.
    - We assume there exists an m and a w unmatched.
    - If m hasn’t proposed to w:
      - The algorithm will not end since m can propose to w. **Contradiction**
    - Else, m has proposed to w in some time point
      - Based on observation 1: “once w is matched, she will never become unmatched”, the w should have been matched when the algorithm end. **Contradiction to observation 1.**
    - Both situation are contradictions.
    - Proved**.**

Theorem 2

- 2. GS algorithm will always output a **stable matching S:**
  - Proof(by contradiction) :
    - **We want to shown there isn't any unstable edge.**
    - We assume there exists an unstable edge in S, such that
      - 1) m prefers w' to w
      - 2) w' prefers m to m'
    - m prefer w' to w, but end up with matched with w. This means m proposed to w' before w.
    - Since now m is not with w', this means m was broken with w' at some time points.
    - This means w' matched with an m'' that has a higher ranking compared with m in the preference list of w'.
    - Recall: observation 2: w's partner's matching will always increase.
    - W's partner changed from m -> m'' -> m', in which m' is lower rank compared with m in the preference list of w'. **Contradiction to observation 2**
    - Proved.



Exercise

True or false? Consider an instance of the stable matching problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m, w) belongs to S.

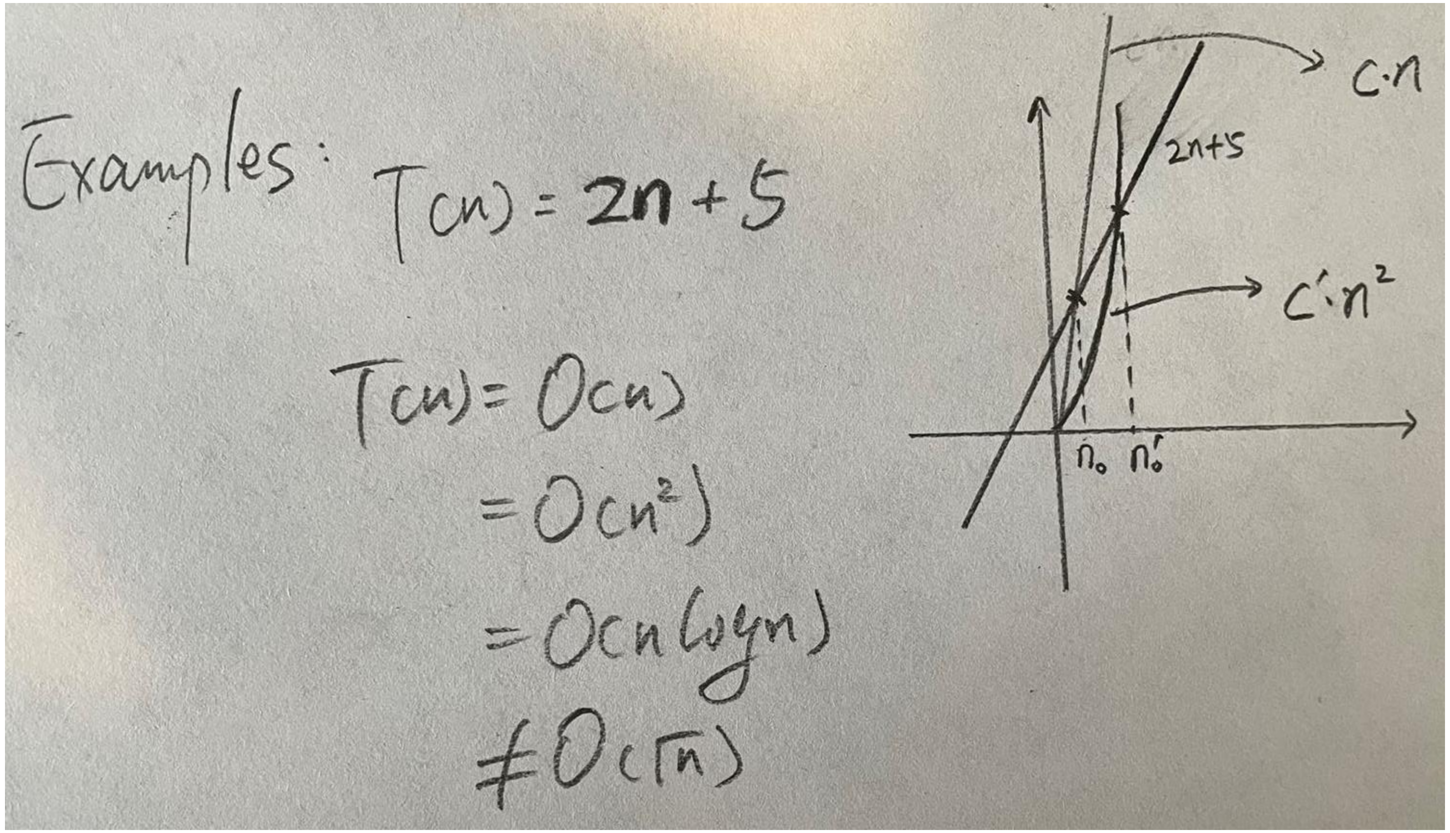
Time Complexity

- Treat algorithm's run time as a function of input size n, T(n)

- **O(f(n)):** T(n) is O(f(n)) if ∃ constant C>0, $n_0 \geq 0$, such that T(n)$\leq c * f(n)$ for $\forall n \geq n_0$
  - Upper bound

Examples

- **Big-O**: T(n) s O(g(n)) if ∃ constant C, $n_0 \geq 0$, such that T(n)≤ $c * g(n)$ for $\forall n \geq n_0$



Properties

- 1. If $f(n) = O\big(g(n)\big), then\ c * f(n) = O\big(g(n)\big) for\ \forall c > 0$
- 2. If $f(n) = O\big(h(n)\big)\ and\ g(n) = O\big(h(n)\big), then\ f(n) + g(n) = O(h(n))$

L’hopital’s Rule

$\log_a n$ vs. n^{ε} $\varepsilon > 0$ const.

$$\lim_{n\to\infty} \frac{\log n}{n^{\varepsilon}} = \lim_{n\to\infty} \frac{\frac{1}{n}}{\varepsilon \cdot n^{\varepsilon-1}} = \lim_{n\to\infty} \underbrace{\frac{1}{\varepsilon}}_{\text{const}} \cdot \underbrace{\frac{1}{n^{\varepsilon}}}_{\downarrow 0} = 0$$

Use Lhopital's rule:

If $f(n), g(n)$ are continuously differentiable,

If $\lim_{n\to\infty} f(n) = \infty$ and $\lim_{n\to\infty} g(n) = \infty$, $g'(n) \neq 0$ when n large enough

then $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$

Exercises

- T(n) = $10n^3 + 4n^2 - n$
- T(n) = $nlg(n) + n^2 + 12$
- T(n) = $c^n + n^c + \lg(n)$ ($c > 1$)

Exercises

3. Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

A $f_1(n) = n^{2.5}$

B $f_2(n) = \sqrt{2n}$

C $f_3(n) = n + 10$

D $f_4(n) = 10^n$

E $f_5(n) = 100^n$

F $f_6(n) = n^2 \log n$

Exercises

- Given a number n, check if n is a prime number or not.
  - Prime number: *a natural number greater than **1** and is divisible by only 1 and itself.*

Exercises

- Given a number n, check if n is a prime number or not.
  - Prime number: *a natural number greater than* ***1*** *and is divisible by only 1 and itself.*

- Algorithm 1:
  - Check from 2 to n-1, check if n can divide them or not. – $\mathbf{O(}\boldsymbol{n}\mathbf{)}$

Exercises

- Given a number n, check if n is a prime number or not.
  - Prime number: *a natural number greater than **1** and is divisible by only 1 and itself.*

- Algorithm 1:
  - Check from 2 to n-1, check if n can divide them or not. – **O($\boldsymbol{n}$)**

- Algorithm 2:
  - Check from 2 to $\sqrt{n}$, check if n can divide them or not. – **O($\boldsymbol{n^{\frac{1}{2}}}$)**