Name: 涂世龙 ArslanTu
ID: 2022214334
E-mail: tsl$_{a}rslan$@163.$sufe.dedu.cn$

Algorithm Design and Complexity Analysis (3978) Spring 23
Assignment 2

Submission Instructions

- The solution must be submitted as a PDF or image file to Blackboard by the deadline.
- All the solutions must be written in English.
- You must write your name in English and Chinese, your Student ID and your e-mail (optional) at the top of the first page.
- Later on, there will be a lot of mathematical formulas. Thus, you are recommended to use LaTex to write your solutions. Microsoft Word version 2016 and above also have a feature for converting a Tex-like script into a mathematical formula.
- If it is too troublesome to use LaTex, handwriting notes are always welcome. You can just write your solutions on paper and scan them using your phone camera. But, please make sure that you write them clearly; otherwise, it will be difficult to grade your submissions correctly.
- There is no need to be too careful about English grammar and typos. It is acceptable as long as it is understandable. You may use some grammar-checking tools, like, Grammarly to help fix your English writing.
- ChatGPT is not allowed. Please don’ t use ChatGPT to solve your problems. It can be easily spotted because the writing is generally **too perfect**.
- Should you have any questions regarding the assignment, please contact me via WeChat or by email at LBundit+gradalgoS23@gmail.com.

Academic Integrity Policy

1. All collaborations must be clearly indicated, e.g., discussed with student A.
2. All outside sources including papers, textbook, lecture note or website must be cited properly. There is no restriction in reference format.
3. Everything except definitions must be **paraphrased**. You are supposed to write everything on your own. Please also try to write mathematical calculations on your own unless it does not seem possible to avoid copying the known calculations.
4. Any submission with $\geq 10\%$ verbatim copied from outside sources is also considered plagiarism.

You will receive the most severe punishment for not abiding to the Academic Integrity Policy. Fail to do so, you will receive a zero score for the entire course, and the incident will be reported to the academic committee, which could end up in being expelled from the program.

Part I: Test Your Understanding (50 points)

Question 1 (30 points)

Arrange the following functions in ascending order of growth rate, where the function $f(n)$ has order higher than $g(n)$ if $f(n) = O(g(n))$.

$$f_1(n) = n^2 f_2(n) = 2^{\sqrt{\log n}}, f_3 = (\log n)^1 00, f_4(n) = 2^{10n}, f_5(n) = n, f_6(n) = n!$$

(Your score is evaluated from the length of the subsequence that is in the correct order.)

Answer:

$$f_3(n)\ f_2(n)\ f_5(n)\ f_1(n)\ f_4(n)\ f_6(n)$$

Question 2 [Exercise 3.1 p.52-53 in [CLRS09]] (30 points)

- **Question (2a).** [Exercise 3.1-1] (5 points) Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using basic definition of Θ-notation, prove that

  $$\max\{f(n), g(n)\} = \Theta(f(n) + g(n)).$$

  Answer:

- **Question (2b).** [Exercise 3.1-3] (5 points) Explain why the statement, "The running time of algorithm A is at least $O(n^2)$.", is meaningless.

  Answer:

  We may assume WLOG that $f(n) \geq g(n)$ such that $max\{f(n), g(n)\} = f(n)$. To prove the equation, we must prove that there exist constants $c_1 > 0$ and $c_2 > 0$, and $n_0 \geq 0$ such that $0 \leq c_1(f(n) + g(n)) \leq f(n) \leq c_2(f(n) + g(n))$. We may assume that c_1 is any constant satisfying $c_1 \leq \lim \frac{f(n)}{f(n)+g(n)}$ such that $0 \leq c_1(f(n)+g(n)) \leq f(n)$. And we can just set $c_2 = 2$ then $f(n) \leq c_2(f(n) + g(n))$. So the equation holds.

- **Question (2c).** [Exercise 3.1-4] (5 points) Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$? Give your explanation for each statement.

  Answer:

  Obviously, $2^n = O(2^n)$. Then by the properties of big O notation, $2 * 2^n = O(2^n)$. What's more, $2^{2n} = 2^n * 2^n$ and they are both $O(2^n)$. So $2^{2n} = O(2^{2n})$ or in other words $O(4^n)$.

- **Question (2d).** [Exercise 3.1-7] (5 points) This question requires extra reading. We may define a strict upper bound and lower bound by change the inequalities in the definitions of O and Θ notations to strict inequalities $<$ and $>$, which gives us the notations of o (little Oh) and ω (little Omega). Show that $o(g(n)) \cap \omega(g(n))$ is an empty set.

Answer:

We could prove this by contradiction. We may assume that exist $f(n)$, $c_1 < c_2$ such that $0 \leq c_1 g(n) < f(n) < c_2 g(n)$ holds. It's obvious that $f(n) = c_3 g(n), c_1 < c_3 < c_2$. So we have $f(n) = \Theta(g(n))$ which means the equality also holds. So such $f(n)$ does not exist, which means the set is empty.

Part II: Challenge Questions (20 points)

Question 3 (20 pt) [Review of Mathematics Induction; See p.59 of CLRS09]
The Fibonacci numbers are defined by the following recurrence.

$$F(0) = 0, F(1) = 1, F(i) = F(i-1) + F(i-2), \text{ for } i \geq 2.$$

That is, each number in the sequence is the sum of the two previous numbers. We wish to express the Fibonacci sequence in terms of the golden ratio ϕ and its conjugate $\bar{\phi}$, which is the root of the equation

$$x^2 = x + 1.$$

In particular,

$$\phi = \frac{1+\sqrt{5}}{2} \text{ and } \bar{\phi} = \frac{1-\sqrt{5}}{2}.$$

Show that $F(i) = \frac{(\phi)^i - (\bar{\phi})^i}{\sqrt{5}}$.

Proof. We proceed the proof by mathematic induction.

Question 3 (a) (10 pt) Base Case: Show that the claim is true for $F(0)$ and $F(1)$. Note that since the recurrence involves two previous numbers, we have to show that the statement is true for the first two numbers $F(0)$ and $F(1)$.

Answer:
$F(0) = \frac{(\phi)^0 - (\bar{\phi})^0}{\sqrt{5}} = 0$
$F(1) = \frac{(\phi) - (\bar{\phi})}{\sqrt{5}} = \frac{\sqrt{5}}{\sqrt{5}} = 1$

Question 3 (b) (10 pt) Inductive Step: Assume inductively that the statement is true for some i and $i-1$. That is, $F(j) = \frac{(\phi)^j - (\bar{\phi})^j}{2}$, for $j = i-1$ and $j = i$. Show that the claim also holds for $F(i+1)$.

Answer:
Now we assume that the equation also holds for $i-1$ and $i-2$. That means $F(i-1) = \frac{(\phi)^{i-1} - (\bar{\phi})^{i-1}}{\sqrt{5}}$ and $F(i-2) = \frac{(\phi)^{i-2} - (\bar{\phi})^{i-2}}{\sqrt{5}}$ Then:

$$
\begin{aligned}
F(i) &= F(i-1) + F(i-2) && (1)\\
&= \frac{(\phi)^{i-1} - (\bar{\phi})^{i-1}}{\sqrt{5}} + \frac{(\phi)^{i-2} - (\bar{\phi})^{i-2}}{\sqrt{5}} && (2)\\
&= \frac{((\phi)^{i-1} + (\phi)^{i-2}) - ((\bar{\phi})^{i-1} + (\bar{\phi})^{i-2})}{\sqrt{5}} && (3)\\
&= \frac{(\phi)^{i-2}(\phi+1) - (\bar{\phi})^{i-2}(\bar{\phi}+1)}{\sqrt{5}} && (4)\\
& && (5)
\end{aligned}
$$

We know that ϕ and $\bar{\phi}$ both satisfy $x^2 = x + 1$, so the equation above is the same as $\frac{(\phi)^i - (\bar{\phi})^i}{\sqrt{5}}$. Now, we have proved that $F(i) = \frac{(\phi)^i - (\bar{\phi})^i}{\sqrt{5}}$.

□

Part III: Programming Challenge (30 points)

Question 4 (30 pt) In class, we have seen the pseudocode of the Gale-Shapley algorithm. However, there are some details in the implementation that were missing. A naive implementation would result in an $O(n^3)$-time algorithm, where n is the number of hospitals (or students). You task is to implement Gale-Shapley algorithm to run in $O(n^2)$.

Remark: If we consider the size of an input, say N is the size of the preference lists, then the $O(n^2)$-time implementation is actually linear on the size of the input.

You are supposed to attach your source code with your assignment to Blackboard. However, you also have an option to add your code with the solution if it is not too long.

```
fun StableMatch() {
// This is just an example.
}
```

Note: You are recommended to use high-level languages that have a lot of data-structure library, e.g., C++ (with STL), Python, Java, Kotlin, Go, etc. Since this is a simple implementation, you may want to take this opportunity to learn popular languages like Kotlin, Go or Rust.

```
#!/usr/bin/python3
# gale_shapley.py
from typing import Any, List

def SatbleMatch(preference_hospital: List[List[int]],
                preference_student: List[List[int]]) -> Any:
    '''
    The hospitals and students are numbered  from 0 to n - 1.
    We just simulate the Gale-Shapley algorithm,
    and all operation happend on list with O(1),
    so the upper bound of run time is n^2.
    '''
    # the number of hospitals (students)
    n = len(preference_hospital)
    # a list of mapping from hospital to its order in a student's preference
    # in the form of list
    h2idx = [[0] * n for _ in range(n)]
    for s in range(n):  # traverse students
        for idx in range(n):  # traverse student's preference
            h2idx[s][preference_student[s][idx]] = idx  # set mapping
    # a mapping from student to the hospital he has been matched
    # in the form of list
    res = [-1] * n
    # a list to store hospitals which still have no students
    hospitals = list(range(n))
    # a list to record the index (preference order)
    # of the next student a hospital should send an offer
    last_offer = [0] * n

    # if there are still some hospitals which have no students
    while hospitals:
        h = hospitals.pop()  # traverse these hospital
        matched = False  # init the state as not matched
        # as the current hospital is not matched,
        # we should keep on finding by the order of its preference list
        while not matched:
            # next student the hospital should send offer to
            next_student = preference_hospital[h][last_offer[h]]
            # when this student is not matched
            if res[next_student] == -1:
                res[next_student] = h
                matched = True
            # when the student is matched but he like h a bit more
            elif h2idx[next_student][res[next_student]] > h2idx[next_student][h]:
                hospitals.append(res[next_student])
                res[next_student] = h
                matched = True
            last_offer[h] += 1
    # return all pairs as a list of tuple
    return [(h, s) for s, h in enumerate(res)]


ph = [[0, 1, 2], [1, 0, 2], [0, 1, 2]]
ps = [[1, 0, 2], [0, 1, 2], [0, 1, 2]]
print(SatbleMatch(ph, ps))
```

References

[KT06] Jon Kleinberg and Éva Tardos. *Algorithm Design*: Pearson New International Edition (2006). Pearson Education Limited Kindle Edition (2014).

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*, The MIT Press (2009). Kindle Edition.