ASSIGNMENT 1

Name : Preksha Yadav
NUID : 002787954

Q1. Arrange the following functions in increasing order of growth
Ans :

- $0.99^n + 1$
- log(log n)
- log 1000n
- log (n3)
- log n + n1/3
- √n
- n / log(n)
- 15n
- 10n + 20n
- n! en

Q2. Give a brief definition for the following:

1) Algorithms : An algorithm is a set of instructions for solving a specific problem or performing a specific task. In the context of data structures, an algorithm is a step-by-step procedure for manipulating a data structure, such as adding or removing elements, searching for a specific element, or sorting the elements in a specific order. Algorithms can be expressed in many forms, such as natural language, pseudocode, or a programming language. They are often analyzed for their time and space complexity, which measures the amount of resources (e.g. memory and time) required to execute the algorithm.

2) Computational tractability:
   The efficiency of an algorithm is determined by the quickest way it can reach the desired solution.
3) Stable Matching : The Stable Matching Problem involves finding a pairing between two equal-sized groups of elements that satisfies specific preferences for each element. The goal is to match individuals in a way that no two people would prefer to be paired with each other instead of their current partner. When there is no such pair, the pairing is considered to be stable. For example, if there are n men and n women, each person would rank members of the opposite sex according to their preferences, and the goal is to match them in a way that results in a stable pairing.
4) Big O notation : The Big O notation is a mathematical notation used to describe the upper bound of the growth rate of a function. It provides a measure of the efficiency of algorithms, specifically the maximum amount of resources (e.g. time, memory) an algorithm requires as the size of its input increases. The Big O notation is a way to describe the asymptotic behavior of a function,

and can be used to compare different algorithms and determine the best one for a particular problem.

5) **Asymptotic order of growth:**

- **Upper Bound:** T(n) is O(n) if there exists constants c and n0 where c>0 and n0 >= 0 such that for all n>=n0 we have T(n)<=c.f(n)
- **Lower Bound:** T(n) is Omega(f(n)) if there exists constants c>0 and n0>= 0 such that for all n>=n0 we have T(n) >= c.f(n)
- **Tight Bound:** T(n) is Theta(f(n)) if T(n) is both O(f(n)) and Omega(f(n))

Q3. The preferences are:
Leslie: Pat > Chris >_ Dana
Chris: Leslie _> Pat _> Dana
Pat: Chris > Leslie> _ Dana
Dana: Chris> _ Leslie> _ Pat
Show that no stable matching exists. (That is, no matter who you put together, they will always be two potential roommates who are not matched but prefer each other to their current roommate.) give examples to justify your solution.

Ans:
Stable Matching : A stable matching is defined as a matching where there does not exist any pair of unmatched individuals who prefer each other to their current match. In this case,
If we start with Lislie, He prefers Pat.
And Pat is currently not paired with any one so he will accept the offer.

LesLie	Pat		
Chris			
Pat	Leslie		
Dana			

Next is Chris, his first preference is Listlie, But as he is paired with other and also with his first preference he will not switch. Next in priority we have Pat, Pat is paired but with his priority number 2, his first priority is Chris. So he will break his pairing and get paired up with Chris.

LesLie	~~Pat~~	Dana	
Chris	Pat		
Pat	~~Leslie~~	Chris	
Dana	Leslie		

Next id Dana, Dana's first priority is Chris, he is paired with a higher priority candidate and next in priority is Leslie. And Leslie is not currently paired so he will pair up with Leslie.

So finally we have,
Lislie and Dana
Chris and Pat paired together.
But this is not an ideal match as Lislie's first priory Pat and Dana's first priority Chris they are not paired correctly. And hence this is an unstable pairing. Therefore, no stable matching exists in this scenario.

Q4. Prove the following:
In the Gale-Shapley algorithm, run with n men and n women, what is the maximum number of times any woman can be proposed to?
Ans: The Gale-Shapley algorithm is a matching algorithm that is used to find stable matchings in a two-sided market. The algorithm is run with n men and n women, and the goal is to find a stable matching between the men and women such that there are no blocking pairs.

In the algorithm, each man starts by proposing to his top choice woman, and each woman can only accept one proposal at a time. If a woman is proposed to by a man that she prefers over her current match, she will reject her current match and accept the new proposal. This process continues until there are no more changes in the matching and it becomes stable.

To prove that the maximum number of times any woman can be proposed to is n^2 - n + 2, we can use the following reasoning:

- At the start of the algorithm, each woman has n men to choose from.
- As the algorithm progresses, each woman will receive at most n-1 proposals (since she can only accept one proposal).
- However, once a woman has accepted a proposal, she will no longer receive proposals from the men she rejected, so the number of proposals she receives will decrease.
- Therefore, the total number of proposals received by any woman will be n + (n-1) + (n-2) + ... + 2 + 1 = n(n+1)/2
- Since each man makes exactly one proposal to each woman, the total number of proposals made by all men is n*n
- Therefore, the maximum number of proposals received by any woman is n(n+1)/2

So the maximum number of times any woman can be proposed to is n^2 - n + 2.

Q5.
Ans:
a. Show that there is always a stable assignment of students to companies, and devise an algorithm to find one.

The task at hand is to match graduating Computer Science and Game Design students to gaming companies, where each company has a specific number of available positions and the students have their own preferences for the companies based on factors such as work, pay, benefits and location. The problem is that there are more students graduating than there are available positions and companies generally want to hire more than one student. This creates a stable matching problem and the goal is to find a way to assign each student to at most one company in a way that all available positions are filled and the assignment is stable.

To address these differences, we can use a variant of the Gale-Shapley algorithm called the Deferred Acceptance algorithm. The Deferred Acceptance algorithm is an extension of the Gale-Shapley algorithm that can handle the case where one side of the market has more agents than the other side.

The basic idea of the Deferred Acceptance algorithm is as follows:

- Each student proposes to their most preferred company that has not yet filled all its available positions.
- Each company then accepts the top k students according to their ranking, where k is the number of available positions for that company.
- The students who were not accepted by any company are then removed from the market, and the remaining students propose to their next most preferred company.
- This process is repeated until all available positions are filled.

It can be proven that the Deferred Acceptance algorithm always results in a stable matching, and it is guaranteed that no student will be left unmatched. The reason is that, in any unstable matching, there must exist a student s1 and a company c that prefer each other over their current match. But since c accepts only k students, and all k positions are filled, s1 can't be matched with c. Therefore, there cannot be an unstable matching.

In terms of the implementation, we can use a priority queue to store the students' preferences and a queue to store the companies' preferences. We can also use a list to store the students who were not accepted by any company.

Overall, the Deferred Acceptance algorithm can be used to find a stable assignment of students to companies in the given scenario, and the algorithm guarantees that all available positions will be filled.

Q6.
A. False.

A counterexample to this statement is when there is an instance of the Stable Matching Problem where there is an individual m who is not ranked first on the preference list of any individual f, and similarly, there is an individual f who is not ranked first on the preference list of any individual m. In this case, no stable matching can be formed that contains a pair (m,f) where m is ranked first on the preference list of f and f is ranked first on the preference list of m.

For example, let's consider 4 individuals, 2 men (m1, m2) and 2 women (f1, f2).

m1 prefers f2, f1
m2 prefers f1, f2
f1 prefers m1, m2
f2 prefers m2, m1
In this case, the matching between m1 and f2 and m2 and f1 is stable but there is no pair (m,f) such that m is ranked first on the preference list of f and f is ranked first on the preference list of m.

It is worth noting that the Stable Matching Problem can have multiple stable solutions, but not all of them will satisfy the statement given in the question.

B.
True.

In any stable matching problem where there is a man m who is the first choice of all women, m must be paired with his first choice in any stable matching.

The reasoning behind this is that, in a stable matching, no woman can be matched with a man she prefers over her current match. Since m is the first choice of all women, any woman who is matched with a man other than m must prefer that man over m. But since m is the first choice of all women, no woman can prefer any other man over m. Therefore, in order for the matching to be stable, m must be matched with his first choice.

Q7.

Q8.

The Gale-Shapley algorithm works by having each man propose to the woman they prefer the most, who is still available to them. The woman then either accepts or rejects the proposal. If she accepts, the two are matched and the woman is removed from the pool of available women. If she rejects, the man moves on to the next woman on his list. This process continues until all men and women are matched.

Suppose a woman, w, prefers man m to m’, but both m and m’ are low on her list of preferences. If she switches the order of m and m’ on her list of preferences, falsely claiming that she prefers m’ to m, and runs the algorithm with this false preference list, it is not necessarily the case that w will end up with a man m’’ that she truly prefers to both m and m’.

This is because if man m’ is low on w's preference list, it is likely that he will be rejected by the women he initially proposes to and will eventually propose to w. At that point, w will reject him as she truly

prefers m. Since man m is higher on w's true preference list, he will likely be matched with a woman before m' gets to w. Therefore, w cannot improve her match by lying about her preferences.

Q9. One algorithm requires n log2(n) seconds and another algorithm requires √n seconds. Which one is asymptotically faster? What is the cross-over value of n? (The value at which the curves intersect?)
Ans:
The algorithm that requires n log2(n) seconds is asymptotically faster than the algorithm that requires √n seconds. The cross-over value of n can be calculated by equating the two time complexities and solving for n.

n log2(n) = √n

Squaring both sides, we get:

n^2 log2^2(n) = n

Isolating n, we get:

n^2 = n / log2^2(n)

Taking the square root of both sides, we get:

n = sqrt(n / log2^2(n))

At this point, the cross-over value of n can be approximated numerically or by using a numerical optimization method to find the root of the above equation. However, finding an exact cross-over value would require more advanced mathematical techniques and is beyond the scope of this answer.

Q10

C. Randomly assume certain teams win and lose each round and eliminate the losers from the preference lists for each team. Can the Gale-Shapley matching algorithm be applied over and over in each round (16 teams, 8 teams, 4 teams, 2 teams) to create stable matches? You can answer this with code or rhetoric.
Ans:
The Gale-Shapley algorithm can be applied in each round of elimination to create stable matches. This can be achieved by updating the preference lists of each team after each elimination round, so that teams that have been eliminated are removed from the preference lists of all other teams.

D. Now combine the lists so that any team can be matched against any other irrespective of conference. Can the Gale-Shapley matching algorithm still create stable matches? (With just one list matching against itself?) You can answer this with code or rhetoric.

Ans:

Yes, the Gale-Shapley matching algorithm can still create stable matches even when teams are matched against each other irrespective of conference. By combining the lists, we create a new set of preferences for each team where the teams can now be matched with any other team. The algorithm will still work as long as the preferences of each team are complete, transitive, and non-trivial.

To implement this with code, we can simply pass the new combined list to the Gale-Shapley algorithm and it will find a stable matching based on these preferences.