Algorithms Homework Guidelines

The following are the most basic requirements of any homework you submit:

1. Your handwriting is legible
2. Your name is clearly written at the top
3. The names of anyone you worked with are also indicated
4. The questions are answered in order
5. There are no scratched out or crossed out sections. What you submit should be the final "camera-ready" version.
6. You used complete sentences in paragraph format

For any question that asks you to "design an algorithm" or "give an algorithm" your solution must include the following sections:

- A written description of your algorithm. This is similar to what you would say if you were asked to describe your answer. Consider including an example to make your description clearer.
- Pseudocode for your algorithm. The pseudocode should give sufficient detail to make an analysis straightforward (e.g. don't hide a for-loop in an English phrase) while still being high-level enough that it can be easily read. Please add comments to your pseudocode where necessary to explain any non-obvious steps.
- An analysis of the worst-case running time of your algorithm in Big-O notation. You should justify your running time using your pseudocode. If your pseudocode does *not* specify which data structures are used, you should state that here. (Note that the choice of data structures will change the running time of your algorithm.)
- If appropriate, a proof that your algorithm correctly solves the problem it was intended to solve. This will mainly apply to greedy algorithms.

Following is an example write-up for the Gale-Shapley Algorithm:

Description: Assume there are an equal number of men and women. Initially, all men and women are free. We then repeat the following steps until all men (and therefore, all women) are paired. A free man m proposes to the highest woman on his list who he has not yet proposed to. If the woman is free, she accepts m's proposal and they are engaged. If the woman is engaged but she prefers m to her current partner, then she breaks her engagement with her current partner (who becomes free) and becomes engaged to m. If the woman is engaged but she prefers her current partner to m, then no change is made and m remains free.

Pseudocode:

```
procedure GALE-SHAPLEY( )
    for all men $m$ do
        mark $m$ as free
    end for

    for all women $w$ do
        mark $w$ as free
    end for

    while there is a free man $m$ who has not proposed to all women do
        $w \leftarrow$ the highest-ranked woman on $m$'s list that he has not proposed to
        if $w$ is free then
            $m$ and $w$ get engaged
        else if $w$ is engaged to $m'$ but prefers $m$ then
            $w$ and $m$ get engaged
            $m'$ is now free
        end if
    end while

    Return the set of engaged pairs
end procedure
```

Running Time:
Let N be the number of men and the number of women. We can store the men's preference lists in an NxN matrix. Similarly, we can store the women's preference lists in an NxN matrix[1]. This allows us to look up preferences in $O(1)$ time. For each man, we also store a single integer index into his preference list that points to the highest-ranked woman he has yet to propose to. This allows us to find the highest-ranking woman not yet proposed to for each man in $O(1)$ time.

We also need a data structure to store the state (free or engaged) for each person. We can use a vector of length $2N$ to store this information where a value of 0 indicates free and a value between 1 to N indicates engaged (to the person with the corresponding id). We can update the status of a person in $O(1)$ time.
Finally, we need a way to keep track of all free men. We can use a queue for this purpose. Pushing and popping to a queue takes $O(1)$ time.

Since we can mark each man and woman as free or engaged in $O(1)$ time, the initial two for-loops takes only $O(N)$ time. Each iteration of the while loop, a man proposes to a woman he has never proposed to before. There are only N men and each man can propose to at most N women, thus there are at most N^2 iterations of the while loop. Since every operation performed in the while loop takes only $O(1)$ time, the while loop has a running time of $O(N^2)$. Thus, the Gale-Shapley algorithm has a worst-case running time of $O(N^2)$.

[1]In reality, the men's and women's preference lists are stored in a slightly different format. However, it does not change the running time and so I did not detail that here.

A Listing of Some Mathematical Notation and Terminology

Note: This document is a work in progress. If you come across mathematical notation that you don't understand, please let me know and I will add it.

Terms

Counterexample A *counterexample* is an example that disproves a claim. A counterexample must provide an actual example. For example, consider the following claim: *Every set of* 5 *numbers must contain the number* 0.

Counterexample: Consider the set $\{1, 2, 3, 4, 5\}$. This is a set of 5 numbers however it does not contain the number 0. Therefore the claim is not true.

Non-negative This phrase means the quantity is greater than or equal to 0. In other words, it is not negative but it could be zero or positive.

Proof by contradiction A *proof by contradiction* is a proof technique. A proof by contradiction begins by assuming the proposition under consideration is false and then derives some contradiction, thus showing the proposition must have been true in the first place.

Transitive A relation over a set of elements (which we'll denote by $r(x, y)$) is called *transitive* if $r(x, y)$ and $r(y, z)$ implies that $r(x, z)$. In other words, if x is related to y and y is related to z then x is related to z. A good example of this is the relation "less than". If $x < y$ and $y < z$ then it must be that $x < z$.

Without loss of generality This phrase can be abbreviated as WLOG and is followed by an assumption that restricts the proof to a special case. "Without loss of generality" means that it is okay to restrict the proof to a special case because the proof of the other cases would follow in the same manner anyways.

Symbols

$\in$ The symbol $\in$ can be read as "is contained in". For example, $x \in A$ means that the element x is contained in the set A.

$\subset$ The symbol $\subset$ means that a set is contained inside of a larger set – i.e. it is a *subset* of a larger set. For example, if A is the set $\{2, 4, 6, 8\}$ then one possible subset is $\{4, 6\}$. We would write this as $\{4, 6\} \subset A$.

$n!$ The expression $n!$ can be read as "n factorial". The factorial of a number n is defined as

$$n! = n * (n - 1) * (n - 2) \dots 3 * 2 * 1$$

$\sum$ The symbol $\sum$ represents a summation of terms. The bounds of the summation are often written below and above the symbol. For example, the summation

$$\sum_{i=1}^{10} 2 * i$$

is a summation that starts at 1 and ends at 10. For each value of i, we are adding $2 * i$ to the total sum – i.e, this summation is equivalent to $2 + 4 + \ldots + 20$.

$|A|$ The vertical bars $|\ |$ represent the *cardinality* of a set. The cardinality of a set is the number of elements in the set. For example, the set $A = \{a, b, c\}$ has cardinality 3. We denote this by writing $|A| = 3$.

Lists and Sets

$\boldsymbol{X} = \{x_1, \ldots, x_n\}$
X is a set that contains n elements. Those elements can be enumerated using the subscripts x_1, x_2, etc.

$X \times Y$
$X \times Y$ is called the *cartesian product* of X and Y. The cartesian product is also a set. It is the set of all pairs (x, y) where $x \in X$ and $y \in Y$. For example, if $X = \{1, 3\}$ and $Y = \{2, 4\}$ then the cartesian product $X \times Y$ is the set $\{(1, 2), (1, 4), (3, 2), (3, 4)\}$.

$\boldsymbol{x_i, x_j \in X}$ **where** $\boldsymbol{i \neq j}$
This statement says that x_i and x_j are 2 *distinct* elements contained in the set X. In other words, x_i cannot be the same element as x_j.

Functions

$\mathbf{f}(\boldsymbol{x})$, $\boldsymbol{T(n)}$, etc.
This notation is used to denote a *function.* A function is a relation that uniquely assigns each element in the input to an element of the output. For example, $f(x) = 2.5x$ is a function that takes a real-valued number x (i.e. an input) and assigns it the real-valued number that is 2.5 times as big (i.e. the output).

Sometimes you will just see the notation $f(x)$ or $T(n)$ by itself. In this case, it is being used to represent some generic function. Traditionally, x is used to represent a real-valued number whereas n is used to represent an integer-valued number.

$\mathbf{lim}_{n\to\infty}\boldsymbol{h(n) = c}$
This notation represents a *limit.* This means that as the value of n approaches infinity (i.e. denoted as $n \to \infty$), the value of $h(n)$ approaches the number c. A formal definition for a limit is as follows:

$$\lim_{n\to\infty} h(n) = c \text{ if for every } \epsilon > 0 \text{ there exists an } n_0 \text{ such that}$$

$$|h(n) - c| < \epsilon \quad \text{for all } n > n_0$$