Practice for Midterm 1.

The following instructions will also be on the exam:

During this exam you may not use electronic devices, books, lecture notes, or any other source that may assist in solving the questions.

Unless stated otherwise, you do not need to repeat any proofs or running time analysis that we did in class. For example, you may sort in $O(n \log n)$ time without describing how. You may not use algorithms that we did not learn, such as hash tables or DFS.

Unless stated otherwise, all graphs are simple and undirected. Running times should use $O(\cdot)$-notation.

The problems below are for practicing the material for midterm 1. Some of the problems are harder than you’ll see on the midterm. Others are easier than the midterm questions. The midterm will consist of exactly 6 problems.

Problem 1. Let $G = (V, E)$ be a graph containing vertex $s$. We run BFS from $s$ to obtain a BFS tree $T$. What is the sum of the degrees of the vertices in $T$?

Problem 2. Design an algorithm that receives a list $L$ of $n$ numbers and checks if there exist three numbers $a, b, c$ in $L$ such that $(a + b)/c = 1234$. The algorithm returns True or False according to whether such $a, b, c$ exist in $L$. The running time should be $O(n^2 \log n)$.

Problem 3. The following recursive function is very silly, and does not make any sense. Write a recurrence relation for the running time of this function (with respect to $n$). Solve the relation to obtain the running time of the function.

```python
def blabla(n):
    if n==1:
        return 1
    else:
        print(blabla(int(n/2)))
        print(blabla(int(n/2)))
        print(blabla(int(n/2)))
        return 1
```

Problem 4. Denote by $T(n)$ the running time of some algorithm, when running on input of size $n$. We know that $T(1) = O(1)$ and that $T(n) = T\left(\frac{n}{2}\right) + n$. Find the running time of the algorithm.
Problem 5. We are given a graph $G = (V, E)$ and three vertices $s, t, r$. We wish to find the shortest simple path that starts at $s$, ends at $t$, and contains $r$. Here is an algorithm for the problem:

- Run BFS from $s$ to find a shortest path to $r$.
- Run BFS from $r$ to find a shortest path to $t$.
- Connect the two paths to obtain a path from $s$ to $t$.

Is this algorithm correct? If yes, explain why. If not, give an example of a graph where it fails.

Problem 6. (a) We are given a list $L$ of $n$ distinct integers. We are also given a list $M$ that contains $n - 1$ integers from $L$. In other words, exactly one integer appears in $L$ but not in $M$. Design an algorithm that finds the missing integer in $O(n)$ time.

Hint: The solution is simple and does not require any method we learned in class.

(b) Now assume that $M$ contains $n - 2$ integers from $L$. That is, there are two integers that are in $L$ but not in $M$. Design an algorithm that finds the missing integer in $O(n \log n)$ time.

(part (b) can be done in $O(n)$ time, but this is a very challenging problem.)

Problem 7. The following recursive function does not make much sense. Analyze the running time of the function with respect to $n$. (Careful! Adam may be trying to trick you.)

```python
def silly(n):
    if n%5==1:
        return 1
    elif n%3==1:
        return 2
    else:
        return n*silly(n-1)
```

Problem 8. Difficult problem!

Let $G = (V, E)$ be a graph containing the vertices $s$ and $t$. Describe an algorithm for finding a shortest path between $s$ and $t$ that has an even length. In other words, out of the paths of even length between $s$ and $t$, we wish to find the shortest one. Hint: create a new graph whose vertex set consists of two copies of $V$. 

Problem 9. Consider an integer $k \geq 2$ and set $n = 2k$. We consider sets of exactly $k$ distinct integers between 1 and $n$. For example, when $n = 4$ and $k = 2$ we consider sets of two distinct integers between 1 and 4. That is, we have the six sets

$$\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}.$$ 

We build a graph $G = (V, E)$ such that $V$ contains a vertex for every set. There is an edge between two vertices if the corresponding sets have at least one number in common. For example, there is an edge between $\{1,2\}$ and $\{1,4\}$, but there is no edge between $\{1,2\}$ and $\{3,4\}$.

For very large $k$ and $n$, find the number of connected components in $G$. You may use the expressions $|V|$ and $|E|$ in your answer without calculation their exact values.

(b) We change the definition of an edge, as follows. There is an edge between two vertices if no number appears in both corresponding sets. That is, if the two sets are disjoint. What is the number of connected components in $G$ now?

Problem 10. Difficult problem!

Let $G = (V, E)$ be a graph. For vertices $v, u$ in $V$, we define by $\delta(v, u)$ the distance between $u$ and $v$. Design an algorithm that finds $\max_{u,v} \delta(u, v)$, where the maximum taken over all pairs of vertices $u$ and $v$. In other words, the algorithm is looking for the maximum distance between two vertices in $G$. For example, in the figure the maximum distance is 3 (since $\delta(s, e) = 3$ and $\delta(b, c) = 3$). The running time should be $O(|V| + |E|)$.

Problem 11. Let $G = (V, E)$ be a graph. We run the BFS algorithm from a vertex $s$ in $V$. Let $d[v]$ denote the distance of a vertex $v$ from $s$ (equivalently, $d[v]$ is the level of $v$ in the BFS tree created by the algorithm).

- Claim. We stop the BFS algorithm during some step (before the algorithm is done). Let $v_1, v_2, v_3, \ldots, v_k$ be the vertices in the queue during that step. Then $d[v_1], d[v_2], d[v_3], \ldots, d[v_k]$ have at most two different values.

Either explain why the claim is true or give a counterexample to it.

Problem 12. Let $G = (V, E)$ be a rooted tree with root $r$. For a vertex $v$ in $V$, let $s[v]$ be the height of the subtree whose root is $v$. In other words, when keeping only $v$ and its descendants (and removing every other vertex), then $s[v]$ is the height of the remaining tree.

Write an algorithm that computes $s[v]$ for every vertex of $V$. The running time should be $O(|V| + |E|)$. Hint: Run BFS to obtain a good order for going over the vertices of the tree.
For each of the following statements, state whether it is true or false. Explain your answers. There will not be such a problem in the midterm, but this is a good way to see that you understand the material.

- Every tree is bipartite.
- Every cycle is almost-bipartite, in the sense that it becomes bipartite after removing at most one edge.
- Every bipartite graph has a 5-coloring.
- There exists an algorithm that sorts \( n \) numbers at time \( O(n \cdot \log(\log n)) \).
- There exists an algorithm that sorts \( n \) numbers at time \( O(n \cdot (\log n)^2) \).
- The recurrence relations \( T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n \) and \( T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n \) lead to the same result.