Assignment 3

Unmarked Questions

Remember to comment out this section when submitting your assignment.

Here are a few simple warm-up problems. Make sure you are able to do them before proceeding to the marked questions.

Q1. Define the set $S \subset \mathbb{N}^2$.

- $2 \in S$.
- If $k \in S$, then so is k^2.
- If $k \in S$ and $k \geq 2$, then so $\frac{k}{2}$.

Prove that every element of S is a power of two.

1. **Predicate.** $P(s) : s$ is a power of two. Show that $P(s)$ is true for all $s \in S$.
2. **Base Case.** 2 is clearly a power of two.
3. **Inductive Step.** Suppose that $k \in S$. By IH, $k = 2^m$ for some $m \in \mathbb{N}$. It follows that $k^2 = (2^m)^2 = 2^{2m}$ is also a power of two. If we further suppose that $k \geq 2$, then we have that $k = 2^m$ with $m \geq 1$. Then $\frac{k}{2} = 2^{m-1}$ with $m - 1 \geq 0$ so $\frac{k}{2}$ is also a power of two.

   Thus, by the Principle of Structural Induction, $P(s)$ is true for all $s \in S$.

Q2. Binary search.

Consider Algorithm 1. We will prove that it is correct. The pre- and post-conditions are stated in the algorithm so we will not repeat them.

1. Variables: let s_i and t_i be the values of variables s and t after the i^{th} iteration of the while loop. Our loop invariant will be: $1 \leq s_i \leq t_i \leq n$, and if x is in A then $x \in A[s..t]$ where $A[s..t]$ is the slice of A from index s to index t inclusive.
2. Proof-of-correctness. We prove the loop-invariant by induction on the number of iterations.

   Base case. At the beginning $s_0 = 1$ and $t_0 = n$ so $1 \leq s_0 \leq t_0 \leq n$. If $x \in A$, then $x \in A[1..n]$.

   Inductive step. For $k \geq 0$, suppose $P(0) \wedge \cdots \wedge P(k)$ is true. We will show that $P(k+1)$ is true. If the while loop terminated before the $k + 1^{\text{st}}$ iteration, then the predicate is trivially true. Thus we can assume that the while loop requires at least $k + 1$ iterations. By IH, we have that $1 \leq s_k \leq t_k \leq n$. Further, if $s_k = t_k$, the while loop would have terminated so it must be the case that $s_k < t_k$. During the $k + 1^{\text{st}}$ iteration, $m = \lfloor \frac{s_k + t_k}{2} \rfloor$. Since $s_k < t_k$, $s_k \leq m < t_k$. Either $s_{k+1} = m$ or $t_{k+1} = m$ so we still have $1 \leq s_{k+1} \leq t_{k+1} \leq n$. It remains to show that $x \in A[s_{k+1}..t_{k+1}]$. There are three cases to consider:

(a) $A[m] = x$. $s_{k+1} = s_k$ and $t_{k+1} = m$ so $x \in A[s_{k+1}..t_{k+1}]$.

(b) $A[m] > x$. Since A is sorted, $A[a] > x$ for all $a \in \{m, ..., t_k\}$. Since $x \in A[s_k..t_k]$ but $x \notin A[m..t_k]$, it must be that $x \in A[s_k..m] = A[s_{k+1}..t_{k+1}]$.

(c) $A[m] < x$. Since A is sorted, $A[a] < x$ for all $a \in \{s_k, ..., m\}$. Since $x \in A[s_k..t_k]$ but $x \notin A[s_k..m]$, it must be that $x \in A[m+1..t_k] = A[s_{k+1}..t_{k+1}]$.

By the principle of Mathematical Induction, $P(n)$ is true for all $n \in \mathbb{N}$. Since $t_k - s_k$ decreases by at least one every iteration, the while loop must terminate. Consider $x \in A$ and $x \notin A$ upon termination. In the first case $s = t$ and $x \in A[s..t]$ so $x = A[s]$. In the second case $x \neq A[s]$ so we output zero as required.

Algorithm 1 BinSearch(A, x)

Require: A is a sorted array of length n where $n \geq 1$.
Ensure: Returns an integer t such that $1 \leq t \leq n$ and $A[t] = x$ if such a t exists; returns 0 otherwise.

```
s ← 1
t ← n
while s ≠ t do
    m ← ⌊(s+t)/2⌋
    if A[m] ≥ x then
        t ← m
    else
        s ← m + 1
    end if
end while
return s if A[s] = x else 0
```

Marked Questions

Q1. [10 Points] Binomial Trees *(maximum 2 pages)*

Let $\mathcal{B}$ be the family of binomial trees. A *binomial tree of order k* is defined recursively as follows:

- A binomial tree of order zero is a single node, denoted B_0, and is in $\mathcal{B}$.
- For all $k > 0$, a binomial tree of order k consists of two binomial trees of order $k-1$ with the root of one tree connected as a new child of the root of the other and is in $\mathcal{B}$.

See Figure 1 for the first few binomial trees.

a. Prove that for all $k \in \mathbb{N}$, a binomial tree of order k has exactly 2^k nodes *using structural induction.*

Proof. □

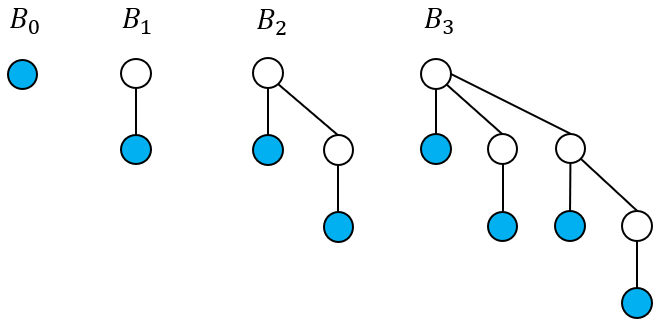

Figure 1: The first four binomial trees. The shaded nodes are leaves.

b. Prove that for all integer $k \geq 1$, attaching a new leaf to every node in a binomial tree of order $k-1$ results in a binomial tree of order k *using structural induction.*

Proof. □

c. Prove that for all non-negative integers k and d, a binomial tree of order k has exactly $\binom{k}{d}$ nodes at depth d *using structural induction.*

If you use any binomial identities not shown in the lecture, you must prove it.

Proof. □

Q2. [10 Points] Bipartite Graphs *(maximum 3 pages)*

Let $G = (V, E)$ be a graph with n vertices and m edges. G is *bipartite* if the vertices V can be partition into two disjoint parts A and B such that all edges have one end-point in A and the other in B. More formally, G is bipartite if $\exists A, B \subset V$ such that $\forall v \in V$ either $v \in A$ or $v \in B$ and $A \cap B = \emptyset$ and for all $(u, v) \in E$ either $u \in A$ and $v \in B$ or $u \in B$ and $v \in A$.

Come up with an algorithm which outputs the partition (A, B) if G is bipartite and outputs `None` **otherwise.** Assume that $n \geq 2$ and the input to your algorithm is an *adjacency list* i.e. G is a list containing n lists where list i stores the indices of the neighbours of vertex i. For a list L, the time required to check the length of L ($|L|$), add an element to the end of L ($L.add(i)$), remove and return the element at the end of L ($L.pop()$) is $O(1)$. The output should be a pair of sets A and B. Generally, for a set S, the time required to check if an element i is in S ($S.contains(i)$), check the length of S ($|S|$) , add to S ($S.add(i)$), and remove element i from S if it exists ($S.remove(i)$) is $O(1)$.

Your algorithm should run in $O(m + n)$ time.

a. Prove that G is bipartite if and only if G does not have any odd cycles.

Proof. □

b. Write the pseudo-code of your algorithm in Algorithm 2.

Algorithm 2 Bipartition(G)

Require: G is a graph with at least two nodes given as an adjacency list.
Ensure: Returns sets A and B such that $A \cap B = \emptyset$, $\forall i \in [n]$ either $i \in A$ or $i \in B$, and for every edge (u, v) in G, one of u or v is in A and the other is in B.

```
1:  u ← 1                                          ▷ this is how you assign variables
2:  for i ∈ [n] do                                 ▷ this is a for-loop
3:      while i > 0 do                             ▷ this is a while-loop
4:          if i < 10 then                         ▷ this is an if-else-statement
5:              u ← 2
6:          else
7:              u ← 3
8:          end if
9:      end while
10: end for
11: return A, B                                    ▷ this how you return something
```

c. Proof-of-correctness setup. List the variables use in the proof, and the proof invariant(s).
Solution.

d. Proof-of-correctness. Prove that your algorithm outputs the desired values.

Proof. □

e. Running time. Evaluate the running time of your algorithm and justify your answer.
Solution.

Additional Questions

Remember to comment out this section when submitting your assignment.

If you would like more exercises consider trying the following problems from your primary and supplementary textbooks. *We will not be providing solutions to these questions* though you are free to find the solution online and discuss them with your peers.

1. David Liu's notes Chapter 2: Exercises 26, 27, 28, 29.
2. Vassos' notes Chapter 2: Exercises 4, 5, 7.