CS200 - Problem Set 9

Due: Monday, April 30 to Canvas

1. Graph Search. In this problem we will use the graph described by the following adjacency list:

A :

vertex	adjacency list
s	u, y
u	v, z, s
y	s, z
v	w, a, u
z	w, u, y
a	v, t
w	v, t, z
t	a, w

We will also use the algorithm `DepthFirstSearch`, which is a version of the Graph Search algorithm we saw in class. It's pseudocode is:

```
Algorithm 1: DepthFirstSearch($A, X, s, f$)
Input  : Adjacency list $A$ for a graph $G = (V, E)$, an array $X$
         of length $|V|$ such that $X[v] = 1$ if $v$ has been
         explored and 0 otherwise, a starting vertex $s$, a goal
         vertex $f$
Output: String "$f$ found!" or "$f$ not found" depending on
         whether $f$ can be found from $s$.
1  if $s == f$ then
2      Return "$f$ found!";
3  else
4      $X[s] = 1$;
5      $d = A[s].length$;
6      for $k = 1$ to $d$ do
7          if $X[A[s, k]] == 0$ then
8              DepthFirstSearch($A, X, A[s, k], f$);
9          end
10     end
11 end
12 Return "$f$ not found"
```

(a) **[6 points]** Please draw the graph the adjacency list corresponds to.

(b) **[6 points]** Suppose you start at s, and want to find t. In what order are vertices explored if you use `DepthFirstSearch`(A, X, s, t)? (Where X is initially set to all zeros)

(c) **[6 points]** Suppose you start at s, and want to find y. In what order are vertices explored if you use `DepthFirstSearch`(A, X, s, y)? (Where X is initially set to all zeros)?

(d) **[6 points]** Qualitatively describe how `DepthFirstSearch` explores the graph.

(e) **[6 points]** In our generic Graph Search algorithm, we did not specify how to choose the next explored edge, in the case that there were multiple edges we could explore. How does `DepthFirstSearch` choose which edge to explore next?

2. **[6 points]** Suppose you have a weighted coin such that the probability of heads is 0.3 and the probability of tails is 0.7. What is the probability of getting at least 3 heads, if you flip the coin 10 times?

3. **[11 points]** A compression algorithm is a function that maps input data to output data, where the output data ideally uses fewer bits than the original data, but there is a decompression algorithm that allows you to recover the original data from the compressed data. A lossless compression algorithm is an algorithm that allows perfect recovery of the original data from the compressed data. (Most real world compression algorithms are not lossless.) Prove that if a lossless compression algorithm makes at least one input file smaller, then it must also make at least one input file larger. (For concreteness, you can assume that the compression algorithm acts on bit strings of any length and compresses to, hopefully, smaller bit strings.)

4. (a) **[11 points]** Prove that for all $n \in \mathbb{Z}$ and $n \geq 0$, and any $r \in \mathbb{R}$ such that $r \neq 1$

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}. \tag{1}$$

(b) **[2 points]** What does the sum evaluate to when $r = 1$?

5. (a) Let $T(n)$ be the number of bit strings of length n that have two consecutive zeros. (A bit string of length n is an element of $\{0, 1\}^n$.) Consider a recurrence relation for $T(n)$.

   i. **[3 points]** What are the initial conditions for the recurrence relation?

   ii. **[6 points]** What are the recursive conditions for the recurrence relation?

   iii. **[6 points]** Use the recurrence relation to calculate $T(5)$.

(b) Consider the following variant to the Tower of Hanoi. We start with all disks on peg 1 and want to move them to peg 3, but we cannot move a disk directly between peg 1 and peg 3. Instead, we can only move disks from peg 1 to peg 2, or from peg 2 to peg 3. So in the case of $n = 1$ disk, we have to first move the disk from 1 to 2, and then from 2 to 3. Let $T(n)$ be the number of moves required to shift a stack of n disks from peg 1 to peg 3, if as usual, we can not put a larger disk on top of a smaller disk. Consider creating a recurrence relation for $T(n)$.

   i. **[3 points]** What are the initial conditions for the recurrence relation?

   ii. **[6 points]** What are the recursive conditions for the recurrence relation? (Explain)

   iii. **[6 points]** Use the iterative method to solve for $T(n)$ and give a big-O bound on $T(n)$.