Name:_______________________________ Student Number:_______________________________

COMP4804 Assignment 1: Due Tuesday January 26, 23:59EDT

Print this assignment and answer all questions in the boxes provided. Any text outside of the boxes will not be considered when marking your assignment.

1 Frequency Assignment in Wireless Networks

We have a graph $G = (V, E)$ in which every vertex has degree 6, $|V| = n$ and $|E| = m$. For each vertex $v \in V$, we color v uniformly (and independently from all other vertices) at random with a color selected from the set $\{1, \dots, k\}$.

1. We say that an edge $e = (u, v)$ is *good* if u and v are assigned different colors in the above experiment and *bad* otherwise. What is the probability that an edge e is bad?

2. What are the expected numbers of bad edges and good edges?

3. We say that a vertex v is *dead* if all 6 of v's incident edges are bad. What is the probability that a particular vertex v is dead? What is the expected number of dead vertices?

4. How many colors k do we need if we want the expected number of dead vertices to be at most: (a) $n/10$, (b) $n/100$, and (c) $n/1000$

Name:______________________________ Student Number:______________________________

2 Approximating Max-2-Sat

A 2-CNF formula is the conjunction of a set clauses, where each clause is the disjunction of two (possibly negated, but distinct) variables. For example, the boolean formula

$$(a \vee b) \wedge (b \vee \neg d) \wedge (\neg a \vee c)$$

is a 2-CNF formula with 3 clauses. When we assign truth values to the variables (a, b, c and d above) we say that the assignment *satisfies* the formula if the formula evaluates to true. In general, it is not always possible to satisfy a 2-CNF-Formula, so we may try to satisfy most of the clauses.

1. Describe an analyze a very simple randomized algorithm that takes as input a 2-CNF formula with n clauses and ouputs a truth-assignment such that the expected number of clauses satisfied by the assignment is at least $3n/4$. (Prove that the running time of your algorithm is small and that the expected number of clauses it satisfies is at least $3n/4$. You may assume that the variables are named $a_1, \ldots, a_m$, $m \leq n$, so that you can associate truth values with variables by using an array of length m.)

2. Your algorithm implies something about all 2-CNF formulas having at most 3 clauses. What does it imply?

3. What does your algorithm guarantee for d-CNF formulas? (Where each clause contains d distinct variables.)

3 Computing the OR of a Bit String

We have are given a bit-string $B_1, \ldots, B_n$ and we want to compute the OR of its bits, i.e., we want to compute $B_1 \vee B_2 \vee \cdots \vee B_n$. Suppose we use the following algorithm to do this:

```
1: for i ← 1, ..., n do
2:   if B_i = 1 then
3:     return 1
4: return 0
```

1. In the worst case, what is the number of times line 2 executes, i.e., how many bits must be inspected by the algorithm? Describe an input $B_1, \ldots, B_n$ that achieve the worst case when the output is 0 and when the output is 1.

2. Consider the following modified algorithm:

```
Toss a coin c
if c comes up heads then
  for i ← 1, ..., n do
    if B_i = 1 then
      return 1
else
  for i ← n, ..., 1 do
    if B_i = 1 then
      return 1
return 0
```

   Assume that exactly one input bit $B_k = 1$. Then what is the expected number of input bits that the algorithm examines.

4 3-Way Partitioning

Suppose you are working on a system where two values can only be compared using the $<$ operator. (Sorting in Python is an example.) Here is an algorithm that, given an array $A[1], \ldots, A[n]$ and a value x classifies the elements of A as either less than, greater than or equal to x.

3-WAY-PARTITION(A, x)

```
for i = 1 to n do
    if A[i] < x then
        add A[i] to S<
    else if A[i] > x then
        add A[i] to S>
    else
        add A[i] to S=
```

1. Let $n_<$, $n_>$ and $n_=$ denote the number of elements of A that less than, greater than or equal to x, respectively. State the exact number of comparisons performed by 3-WAY-PARTITION.

2. Show that there exists a randomized algorithm that uses only 1 random bit (coin toss) and performs and expected number of comparisons that is $2n_= + \frac{3}{2}(n_< + n_>)$.

Name:_______________________________ Student Number:_______________________________

5 The height of a skiplist

Suppose we start with a list $L_0 = l_1, \ldots, l_n$. We obtain a new list L_1 by tossing a fair coin for each element l_i and adding l_i to L_1 iff the coin toss comes up heads.

1. What is the probability that l_i is in L_1? From this, compute the expected size of L_1.

2. Suppose we continue in this manner to obtain a list L_2 by tossing coins for each element of L_1. In general, to obtain L_i ($i > 0$), we toss a coin for each element in L_{i-1} and add that element to L_i iff the coin toss comes up heads.

   What is the probability that any particular element l_j is in L_i? From this, compute the expected size of L_i.

3. Show that the expected time required to build all the lists $L_1, L_2, L_3, \ldots$ is $O(n)$.

4. Define the indicator variable

$$I_i = \begin{cases} 1 & \text{if } L_i \text{ is not empty} \\ 0 & \text{otherwise.} \end{cases}$$

Observe that I_i never exceeds the size of L_i. The random variable $X = \sum_{i=0}^{\infty} I_i$ is the *height* of the skip list. Show that $\mathbf{E}[X] = \log_2 n + O(1)$. [Hint: You can use either inequality $I_i \leq 1$ or $I_i \leq |L_i|$ depending on the value of i.]

FYI: Skiplists are an efficient simple alternative to balanced binary search trees that support insertion and deletion in $O(1)$ expected time and searching in $O(\log n)$ expected time. Feel free to use Google for more information. Here's picture of one: