1. The TSP

Recall the first approximation to the traveling salesman problem (TSP) which uses the MWST on a complete Euclidean graph to find a 2-approximation for the TSP problem.

(a) State exactly what a 2-approximation means for the TSP problem.
   Answer:
   2-approximation means, for any complete Euclidean Graph $I$, the TSP Approximation Algorithm (TSPALG) produces a tour that is at most twice the cost of the optimal tour (OPT). That is, $TSPALG(I) \leq 2 \times OPT(I)$.

(b) Give an example of a non-Euclidean complete graph $H$ where, when you use this approximation method, $H$, it fails to give a 2-approximation to the TSP. Say what the optimal TSP solution is for your graph and explain the answer you obtain when the 2 approximation algorithm is run on $H$.
   Answer:
   Consider the following non-euclidean graph $I$.

![Graph Image](image.png)

and a Minimum Weight Spanning Tree contained in $I$. 

Traversing the MWST yields the following path.
Following this path in the MWST on $I$ yields the following approximation for the Traveling Salesman Problem.

Observe that the cost of the approximate tour is 104, whereas the optimal tour given below has cost 8.
(c) (T or F) There is no approximation algorithm which is better than a 2-approximation for any complete Euclidean graph.
Answer: False, Christofide’s Algorithm is a $\frac{3}{2}$-Approximation Algorithm for the Traveling Salesman Problem.

2. Freivald’s Algorithm for Checking Non-Square Matrix Multiplication
You are given the 3 matrices $A$, $B$, and $C$ below where $A$ is $3 \times 2$, $B$ which is $2 \times 3$ and $C$ is $3 \times 3$.

\[
A = \begin{bmatrix}
5 & 1 \\
-1 & 0 \\
2 & -2
\end{bmatrix}, \quad
B = \begin{bmatrix}
0 & 1 & 0 \\
3 & 2 & 1
\end{bmatrix}, \quad
C = \begin{bmatrix}
3 & 7 & 1 \\
0 & -1 & 0 \\
-6 & -2 & -2
\end{bmatrix}
\]

Now you run Freivald’s algorithm as in the square matrix case but with a random binary column vector $V$ of 3 bits which is generated to test whether $A \times B = C$.

Answer the following 4 short answer questions about this algorithm:

(a) What is the output of Freivald’s algorithm when the column vector $V$ is

\[
V = \begin{bmatrix}
1 \\
0 \\
1
\end{bmatrix}
\]
Answer:
Recall that Freivald’s algorithm computes the following to determine whether \( AB = C \) given a column vector \( V \).

\[
\left\{
\begin{array}{c}
\text{yes if } 0 \\
ABV - CV \\
\text{no otherwise}
\end{array}
\right.
\]

Observe that \((AB - C) = ABV - CV\).

White:
\[
\begin{bmatrix}
0 & 0 & 0 \\
-3 & 0 & 0 \\
0 & 2 & 0
\end{bmatrix}
\begin{bmatrix}
0 \\
1 \\
1
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 2
\end{bmatrix} \neq 0
\]

Yellow:
\[
\begin{bmatrix}
0 & 0 & 0 \\
-3 & 0 & 0 \\
0 & 2 & 0
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix} = \begin{bmatrix}
-3 & 0 & 0
\end{bmatrix} \neq 0
\]

Freivald’s outputs “no” for \( V \).

(b) With \( V \) as in (i), is the output of the algorithm correct or incorrect? Explain why this is so.
Answer:
Since \( AB \neq C \), Freivald’s output is correct in this case.

(c) What is the exact error probability of Freivald’s algorithm for this particular \( A,B, \) and \( C \) when \( V \) is picked at random? Explain.
Answer:
We can compute the probability of error by enumerating all 8 possible choices of \( V \).

\[
V = \begin{bmatrix}
0 \\
1 \\
1
\end{bmatrix} \text{ correct}
\]

\[
V = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix} \text{ correct}
\]

\[
V = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix} \text{ is wrong Feivald outputs ”yes”}
\]

\[
V = \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix} \text{ is wrong Feivalds outputs ”yes”}
\]

\[
V = \begin{bmatrix}
1 \\
0 \\
1
\end{bmatrix} \text{ correct}
\]

\[
V = \begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix} \text{ correct}
\]

The probability of error = \( \frac{2}{8} = \frac{1}{4} \).
(d) (T or F) The error probability for this algorithm on some \( A, B, \) and \( C \) (of the given dimensions) could be bigger than \( \frac{1}{2} \) as the matrices are not square.

False, the probability of error \( \leq \frac{1}{2} \) as shown by the proof given in the book Randomized Algorithms.

(e) Is this algorithm a Monte Carlo algorithm or a Las Vegas algorithm, or neither? Explain.

It’s a Monte Carlo algorithm since its runtime is always efficient, and makes errors with some probability.

3. NP Problems and Graph Coloring

For an integer \( k \), a graph is \( k \)-colorable if you can color the vertices of the graph with \( k \) different colors in such a way that no two adjacent vertices have the same color. Let \( GC \) (Graph Coloring) be the problem of finding, given a graph \( G \), the smallest \( k \) for which \( G \) is \( k \)-colorable. This problem is NP-hard.

(a) The decision version of \( GC \), called \( DGC \) is: Given a graph \( G \), does \( G \) have a coloring using \( k \) or fewer colors.

Describe an efficient verifier \( V \) which shows that \( DCG \) is in NP.

Specifically, you need not include every detail here but you should say what the input and outputs of \( V \) are, then give a brief version of the algorithm defining \( V \), and then say what makes \( V \) a correct verifier for \( DCG \).

Answer:

\[
V(G, c, k) = \begin{cases} 
\text{accept} & \text{if } c \text{ is a valid coloring of } G \text{ and uses at most } k \text{ colors} \\
\text{reject} & \text{otherwise}
\end{cases}
\]

\( V \) is efficient since it takes \( O(|V|^2) \) steps to check whether the given coloring \( c \) is valid or not, by iterating all vertices and checking that all neighbors have different colors.

(b) For a graph \( G = (V, E) \) define \( \deg(G) = \max \{ \deg(v) | v \text{ is a vertex in } V \} \).

Give a proof or a counterexample for the following statement:

Any graph \( G \) has a coloring of no more than \( 1 + \deg(G) \) colors.

Answer:

This statement is true and we can define an algorithm to assign a coloring to the graph using at most \( 1 + \deg(G) \) colors.

Let \( G = (V, E) \), \( \text{colors} = 1, 2, 3, \ldots, 1 + \deg(G) \), and \( v \in V \). For \( i = 1, 2, \ldots, n \) color \( v_i \) with the least color not used by any of \( v_i \)'s neighbors. There is always such a color since \( \deg(v_i) < 1 + \deg(G) \).
4. Polynomial Identity Testing (PIT)

Recall the randomized PIT algorithm which can be used to test if two single variable polynomials \(f(x)\) and \(g(x)\) both of degree 9 are identical. Which of the following is true about our randomized test when the random inputs in the 5 questions below are chosen from a set \(S\) of 100 different numbers.

Answer the following 5 True/False questions about this PIT problem. No partial credit is given for this one.

(a) If we pick (white 12) (yellow 19) random points from \(S\) and find that \(f(x) \times g(x) = 0\) for all 19 chosen points then we know (with probability 1) that \(f(x) \times g(x) \equiv 0\).

Answer:

White: False
With some small probability we could have selected twelve of the \(9 + 9 = 18\) roots for the polynomial \(f \times g\). Therefore, we cannot claim \(f(x) \times g(x) \equiv 0\) with probability 1.

Yellow: True
Since we sample without replacement we’re guaranteed to select one input that is neither a root of \(f\) nor \(g\). Therefore, we know with probability 1 that \(f(x) \times g(x) \equiv 0\).

(b) If in fact \(f(x) \neq g(x)\), and we choose 1 random \(r\) from \(S\) to use for the identity test for \(f(x)\) and \(g(x)\) then the polynomial identity test has error probability \(\leq 0.1\).

Answer:

True, the algorithm errs if it selects one of the 9 roots which happens with probability at most \(9/100 < 0.1\).

(c) If we pick 6 random inputs \(r_1, r_2, \ldots, r_6\) from \(S\) and we test and find that \(f(r_i) = g(r_i)\) for all 6 inputs, then we know that \(\text{prob}(f(x) \equiv g(x)) \geq \frac{1}{2}\).

Answer:

False, after observing \(f(r_i) = g(r_i)\) on six inputs we can’t claim that \(f(x) = g(x)\) with any certainty.

(d) If we pick 6 random inputs from \(S\) and we test if \(f(x) = g(x)\) on these 6 inputs and find them not equal on exactly three of them then we only know with probability \(9/100\) that \(f(x) \neq g(x)\).

Answer:

False. Observe that with Polynomial Identity Testing (PIT), as soon as you find an input \(x\) such that \(f(x) \neq g(x)\), you know with probability 1 that \(f(x) \neq g(x)\).

(e) If we pick 12 random inputs from \(S\) and we test if \(f(x) = g(x)\) on these 12 inputs, then we know with certainty (prob = 1) that \(f(x) = g(x)\) for all inputs \(x\).
True. $f(x) - g(x)$ has degree $\leq 9$ so if we find 12 inputs where they are equal, we know with probability 1 that $\forall x. f(x) = g(x)$. 