Problem 1 [20 points]

Recall that a semiring is a 5-tuple $(R, \oplus, \otimes, \bar{0}, \bar{1})$, where $R$ is a set, $\oplus$ and $\otimes$ are binary operations $R \times R \rightarrow R$, and $\bar{0}, \bar{1} \in R$, satisfying the semiring laws:

\[
\forall a, b, c \in R:
\]

\[ (a \oplus b) \oplus c = a \oplus (b \oplus c) \quad (1) \]
\[ \bar{0} \oplus a = a \oplus \bar{0} = a \quad (2) \]
\[ a \oplus b = b \oplus a \quad (3) \]
\[ (a \otimes b) \otimes c = a \otimes (b \otimes c) \quad (4) \]
\[ \bar{1} \otimes a = a \otimes \bar{1} = a \quad (5) \]
\[ a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \quad (6) \]
\[ (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \quad (7) \]
\[ \bar{0} \otimes a = a \otimes \bar{0} = \bar{0}. \quad (8) \]

We showed in class that the Forward algorithm can be used to compute aggregate statistics over all paths from a node $s$ to another node $t$ in a directed acyclic graph. By mapping edge weights into various semirings we can compute various statistics without modifying the algorithm. For instance the semiring

\[
(R_{\geq 0} \cup \{\infty\}, \min, +, \infty, 0)
\]

computes the length of the shortest path.

For each of the following quantities of interest, define a semiring which can be plugged into the Forward algorithm to calculate it. Be careful that the object you’ve defined satisfies the semiring laws (though you are not required to prove it).

1. [10 points] Whether there exists any path from $s$ to $t$.
   $R = \{\text{True, False}\}$. Define $\oplus$, $\otimes$, $\bar{0}$, and $\bar{1}$.
   How do you map weighted edges into your semiring?
   What is returned when there is no path from $s$ to $t$?
2. [10 points] The set of all paths from \( s \) to \( t \).

\[ R = 2^E, \] where \( E \) is the set of edges in the graph. Define \( \oplus, \otimes, \vec{0}, \) and \( \vec{1} \).

How do you map weighted edges into your semiring?

What is returned when there is no path from \( s \) to \( t \)?

**Problem 2 [15 points]**

Suppose \( G \) is a CFG and we are given a word \( w \in L(G) \) such that \( |w| = n \). What is the exact derivation length of \( w \) in \( G \) (that is, the number of derivation steps in which \( w \) is derived) if:

1. [7 points] \( G \) is in Greibach Normal Form (GNF)
2. [8 points] \( G \) is in Chomsky Normal Form (CNF)

Explain your answers in both cases.

**Problem 3 [30 points]**

Show that if \( L \) is a context free language that does not accept the empty word \( \epsilon \), then there exists a PDA \( M = (Q, \Sigma, \Gamma, \delta, q, Z, \emptyset) \) that accepts \( L \) by empty stack and has the following two properties:

(1) \( M \) has a single state \( q \) (i.e. \( |Q| = 1 \)).

(2) \( M \) has no \( \epsilon \)-transitions.

**Problem 4 [35 points]**

(1) \( S \rightarrow NP \ VP \)
(2) \( S \rightarrow NP \ VP \ PP \)
(3) \( NP \rightarrow det \ n \)
(4) \( NP \rightarrow n \)
(5) \( NP \rightarrow NP \ PP \)
(6) \( VP \rightarrow aux \ VP \)
(7) \( VP \rightarrow v \ NP \)
(8) \( PP \rightarrow p \ NP \)

1. [10 points] The grammar given above is not in CNF. Modify it into a CNF grammar that accepts the same language. (Modify rules 2, 3, 6, 7, and 8.) Your modification should be reversible, meaning that a derivation from your new grammar can be deterministically converted into a derivation from the original grammar.

2. [20 points] Run the CYK parsing algorithm on the input “\( det \ n \ aux \ v \ n \ p \ det \ n \)”, showing the two-dimensional table that is constructed by the algorithm. Each entry in your table should have the form \( A_{i,k} \rightarrow B_{i,j}C_{j+1,k} \), indicating the backpointers of the constructed parse tree or trees. You do not need to do ambiguity packing.

3. [5 points] How many different parse trees does the algorithm construct? How is this indicated in your table?