1. This problem provides some experience with the Chomsky hierarchy (regular, context-free, context-sensitive, and unrestricted grammars). For each of the following languages (denoted L), provide a grammar G occupying the most restrictive level of the hierarchy possible.

   (a) $\Sigma = \{a, b\}$, $L(G) = \{a^n b^n \mid n \in \mathbb{N}\}$.

   Solution: $S \rightarrow aSb \mid \epsilon$
   This is a context-free grammar, but it cannot be rewritten as a regular grammar.

   (b) $\Sigma = \{a, b\}$, $L(G) = \{\alpha \in \Sigma^* \mid \alpha \text{ is a palindrome}\}$.

   Solution: $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$
   This is a context-free grammar, but it cannot be rewritten as a regular grammar.

   (c) $\Sigma = \{a, b\}$, $L(G) = \{a^n b^m \mid n \in \mathbb{N} \text{ odd}, m \in \mathbb{N} \text{ even}\}$.

   Solution: A regular grammar exists for this language:

   $$\begin{aligned} S &\rightarrow aX \mid aZ \\ X &\rightarrow aS \mid \epsilon \\ Z &\rightarrow bW \\ W &\rightarrow bZ \mid b \end{aligned}$$

   (d) $\Sigma = \{a, b, c\}$, $L(G) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$.

   Solution: A unrestricted grammar exists for this language (below), which can be transformed into a context-sensitive grammar using some variable substitutions (but a context-free grammar does not exist).

S	$\rightarrow$	$aSBC \mid \epsilon$	build up a string of a's followed by $BCBC$...
CB	$\rightarrow$	BC	sort the non-terminals that are out of order
aB	$\rightarrow$	ab	start transforming non-terminals at the a/B boundary
bB	$\rightarrow$	bb	continue transforming the B non-terminals
bC	$\rightarrow$	bc	start transforming non-terminals at the b/C boundary
cC	$\rightarrow$	cc	finish by transforming the C non-terminals

2. Transform the following context-free grammars into Chomsky normal form.

   (a) $\begin{aligned} S &\rightarrow aA \mid bA \mid aE \mid bE \\ A &\rightarrow S \\ E &\rightarrow \epsilon \end{aligned}$

   Solution: The first step is to remove all rules of the form $X \rightarrow \epsilon$. Then we need to remove rules of the form $X \rightarrow Y$. This leaves us with

   $$S \rightarrow aS \mid bS \mid a \mid b.$$

Finally we need to replace rules of the form $X \to aY$ with rules of the form $X \to ZY$ plus $Z \to a$. This gives us the CNF grammar

$$\begin{aligned} S &\to AS \mid BS \mid a \mid b \\ A &\to a \\ B &\to b. \end{aligned}$$

(b) $\begin{aligned} S &\to ab \mid A \\ A &\to bAaA \mid b \end{aligned}$

Solution: We can start by handling the rule $S \to A$ by replacing A with any expression X where $A \to X$ is one of our rules.

$$\begin{aligned} S &\to ab \mid bAaA \mid b \\ A &\to bAaA \mid b \end{aligned}$$

Then we can replace bA with X and aA with Y.

$$\begin{aligned} S &\to ab \mid XY \mid b \\ A &\to XY \mid b \\ X &\to bA \\ Y &\to aA \end{aligned}$$

Finally we can transform mixtures of non-terminals and terminals (and multiple terminals) as we did in part (a) to obtain the CNF grammar

$$\begin{aligned} S &\to ZW \mid XY \mid b \\ A &\to XY \mid b \\ X &\to WA \\ Y &\to ZA \\ Z &\to a \\ W &\to b. \end{aligned}$$

3. In this problem, we study some properties of the Nussinov algorithm for RNA folding. Recall that, given an RNA sequence $R = r_1 \ldots, r_L$, $B(i, j)$ is the maximal number of base-pairings for the substring $r_i \ldots r_j$, and satisfies the recursion,

$$B(i,j) = \max \begin{cases} B(i+1, j-1) + \sigma(r_i, r_j), \\ B(i+1, j), \\ B(i, j-1), \\ \max_{i<k<j-1}\{B(i,k) + B(k+1, j)\}, \end{cases}$$

for $i + 1 < j$, with base cases $B(i, i) = 0$ and $B(i, i + 1) = 0$. Here $\sigma(r_i, r_j) = 1$ if r_i and r_j can base pair, 0 otherwise.

(a) (modified from Durbin, question 10.3) What is the minimum length of a hairpin loop in the algorithm above? Modify the Nussinov folding algorithm so that hairpin loops have

a minimum length of h.

Solution: In the algorithm above, r_i and r_{i+2} can base pair together, but r_i and r_{i+1} cannot. So the minimum length of a hairpin loop is 1 (i.e. 1 unpaired base). To ensure a hairpin loop has at least h unpaired bases, we only need to modify the base case of the algorithm

$$B(i,j) = 0, \quad \text{if } |j-i| \leq h.$$

(b) How can you modify the Nussinov folding algorithm to handle *circular* RNA?

Solution: Before, we only filled in the DP table for $i \leq j$ (the upper triangular part). To handle circular RNA, we can fill in the table as before, then then continue to fill the lower triangular part in an anti-diagonal fashion starting from the lower left corner (or alternative this can be thought of as shifting over the lower triangular part as shown in the figure below). When initializing and searching over $k = i+1, \cdots, j-1$, we want to work modulo L. Finally, we want to return the maximum over all $B(i, i-1 \mod L)$.

More formally:

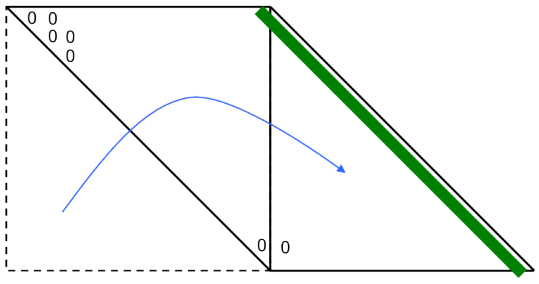
i. Initialization: $B(i,i) = 0$, $B(i, i+1 \mod L) = 0$, for $i = 1, 2, \cdots, L$ (zeros shown in the figure below, which carry on the entire diagonal).

ii. Recursion:

$$B(i,j) = \max \begin{cases} B(i+1 \mod L, j-1 \mod L) + \sigma(r_i, r_j), \\ B(i+1 \mod L, j), \\ B(i, j-1 \mod L), \\ \max_k \{B(i,k) + B(k+1, j)\}, \end{cases}$$

where $i < k < j-1$, if $i < j$, and $i < k \leq L$ and $1 \leq k < j$ otherwise.

iii. Final: return $\max_i B(i, i-1 \mod L)$ (green diagonal in the figure below).



Note: as pointed out in discussion, with the matching function σ above, all $B(i, i-1)$ should be equal, so we can just take $B(1, L)$ as before, with a slight modification:

$$\sigma(r_1, r_L) = 0.$$

This restriction ensures that we do not pair the "first" and "last" bases in our string, since they are adjacent for circular RNA. However, for a more complicated matching function (for example, one that has prefers local vs. long-range base pairings), this will no longer be the case.