CSE525 Monsoon 2020 — Ashima Garg (PhD19003)
Homework 10 Problem 1 — Pramil Panjwani (PhD19008)

**Solution:**

- **Problem Definition**: For given string S[1 ...n] of parentheses and brackets. editDistanceBP(i, j) = minimum edit distance required for balancing parentheses of string S[i ...j]. Let diff(i,j) be number of edits required to form S[i] and S[j] a pair of '()' or '[]'.
- **Formula for DP**: diff(i, j) = 0 if ($S[i] =' ('$ and $S[j] =')'$) or ($S[i] =' ['$ and $S[j] =']'$). (i.e. S[i] and S[j] forms set of balanced parentheses), else if diff$(i, i+1)$ = 1 if ($S[i] =' ('$ and $S[j] =']'$) or ($S[i] =' ['$ and $S[j] =')'$), else diff$(i, i+1)$ = 2 if (($S[i] =')'$ or $S[i] =']'$) and ($S[j] =' ['$ or $S[j] =' ('$))

  Case 1: editDistanceBP$(i, j) = 0$ if $i > j$. Case 2: editDistanceBP$(i, i) = 1$ if $\forall i = 1 \ldots n$. Case 3: editDistanceBP$(i, i+1)$ = diff$(i, i+1)$. Case 4: editDistanceBP$(i, j)$ = min{min{(editDistanceBP$(i, j-1)$, editDistance$(i+1, j-1)$, editDistance$(i+1, j)$) **+ 1** if $S[i] \neq S[j]$. (i.e. S[i] and S[j] do not form set of balanced parentheses)}, editDistanceBP$(i, k)$ + editDistanceBP$(k+1, j)$ $\forall k$ in $i$ to $j-1$. }.
  **Deletion:** If diff$(i, j) = 2$, both the first and last characters are deleted, editDistanceBP$(i, j)$ = editDistanceBP$(i+1, j-1)$ + 2. If diff(i, j) = 1, if character deleted from last (S[j]), editDistanceBP$(i, j-1)$ + 1. If deleted from beginning(S[i]), editDistanceBP$(i+1, j)$ + 1.
  **Insertion:** If diff$(i, j) = 2$, characters both at start and end are inserted, editDistanceBP$(i, j)$ = editDistanceBP$(i, j)$ + 2. If character inserted after last (S[j]), editDistanceBP$(i+1, j)$ + 1. If inserted in the beginning before (S[i]), editDistanceBP$(i, j-1)$ + 1.
  **Replace:** If diff$(i, j) = 2$, both the characters are replaced such that they now match, and editDistanceBP$(i, j)$ = editDistanceBP$(i+1, j-1)$ + 2. If diff$(i, j) = 1$, one of the characters are replaced, and editDistanceBP$(i, j)$ = editDistanceBP$(i+1, j-1)$ + 1.
  Now, minimum of all the above case analysis will be editDistanceBP$(i, j)$ = min{(editDistanceBP$(i, j-1)$, editDistanceBP$(i+1, j-1)$, editDistanceBP$(i+1, j)$) **+ 1** if $S[i] \neq S[j]$} and min{editDistanceBP$(i, j)$=min(editDistanceBP( editDistanceBP$(i, k)$ + editDistanceBP$(k+1, j)$ $\forall k$ in $i$ to $j-1$.) } Case 5: editDistanceBP$(i, j)$ = min{editDistanceBP$(i+1, j-1)$ if ($S[i] =' ('$ and $S[j] =')'$) or ($S[i] =' ['$ and $S[j] =']'$) (i.e. S[i] and S[j] forms set of balanced parentheses), editDistanceBP$(i, k)$ + editDistanceBP$(k+1, j)$ $\forall k$ in $i$ to $j-1$. }
- **Memoization Structure**: Use 2D array $DP[0 \ldots n][0 \ldots n]$. $DP[i][j]$ stores editDistanceBP(i, j) i.e. the minimum edit distance required for balancing parentheses for string $S[i \ldots j]$.
- **How to fill the memo**: Initialize full 2D array DP[0 ...n][0 ...n] = Infinity. Using Case 1 defined above, dp[i][i] = 1 $\forall i = 1 \ldots n$. Then, using Case 2: dp[i][i+1] = diff(i, i+1) $\forall i = 1 \ldots n$. Rest of the values will be filled diagonally for $i < j$ starting from the main largest diagonal where $i = j$ and towards the upper triangle of the diagonal.

```
for l in $(2 \ldots n)$: ⟨⟨O(n)⟩⟩
  for i in $(1 \ldots n-l)$: ⟨⟨O(n)⟩⟩
      j = i + l, if $j == n$, continue
      if diff$(i, j)$ = 0: DP[i][j] = DP[i+1][j-1]
      else: DP[i][j] = min(DP[i+1][j], DP[i][j-1], DP[i+1, j-1]) + 1
          for k in $(i \ldots j-1)$: ⟨⟨O(n)⟩⟩
              DP[i][j] = min(DP[i][j], DP[i][k]+DP[k+1][j])
return DP[1][n]
```

- **How to solve original problem from memo**: To find the minimum edits between S[1 ...n] for balancing parentheses = editDistanceBP[1][n] = DP[1][n].
- **Space and time complexity**: Time complexity loop 1 ($l = 2 \ldots n$) $O(n)$, then for (loop 2)($i = 1 \ldots n-l$) is $(n-1)+(n-2)+(n-3)\ldots 1 = (n(n+1)/2 - n) = (n^2)/2 = O(n^2)$ and again for inner most loop (loop 3) $(j = 0 \ldots k)$(worst case) it is $[O(n^2)+O(n^2)\ldots O(n^2)] n times = n * O(n^2) = O(n^3)$ .Total $T(n) = O(n^3) + C$(for all the comparisons including the comparisons of in diff function). $T(n) = O(n^3)$.
  Since we need to create 2D matrix DP[0 ...n][0 ...n]. Therefore, space complexity is $O(n^2)$.

```
def editDistanceBP($S[1 \dots n]$):
    create empty DP[n][n]. Initialize DP[n][n] with infinity.
    for i in $(1 \dots n-1)$ : ⟨⟨O(n)⟩⟩
        DP[i][i] = 1, DP[i][i+1] =$diff$(i, i+1).
    for l in $(2 \dots n)$: ⟨⟨O(n)⟩⟩
        for i in $(1 \dots n-l)$: ⟨⟨O(n)⟩⟩
            j = i + l, if $j == n$, continue
            if diff$(i, j)$ = 0: DP[i][j] = DP[i+1][j-1]
            else: DP[i][j] = min(DP[i+1][j], DP[i][j-1], DP[i+1, j-1]) + 1
                for k in $(i \dots j-1)$: ⟨⟨O(n)⟩⟩
                    DP[i][j] = min(DP[i][j], DP[i][k]+DP[k+1][j])
    return DP[1][n]
```

■