151554_Alicja_Augustyniak_biostrukturRNA2

April 14, 2025

Zaimplementuj miarę INF dla struktur w formacie dot-bracket. Do raportu dołącz kod oraz przedstaw wyniki dla struktur otrzymanych wcześniej (ocena 4.0)

[17]:
```
def dotbracket_pairs(dotbracket):
    stack = []
    pairs = set()
    for i, char in enumerate(dotbracket):
        if char == '(':
            stack.append(i)
        elif char == ')':
            if stack:
                j = stack.pop()
                pairs.add((j, i))
    return pairs

def calculate_inf(x, z):
    a = dotbracket_pairs(x)
    b = dotbracket_pairs(z)

    TP = a & b
    FN = a - b
    FP = b - a

    if len(TP) == 0:
        return 0

    INF = (len(TP) / (len(TP) + len(FN))) * (len(TP) / (len(TP) + len(FP)))
    return INF

turbofold = ".................(((((((((.....)))))))))..(((((....))))).(((((...
 ↪....))))).........."
locarna = ".........(((.....(((((((.........)))))))..((((......)))).(((((...-.
 ↪...)))))).....)))."



#def read_file(filepath):
#    with open(filepath, 'r') as file:
```

```
#         return file.readline().strip()

#turbofold = read_file('turbofold.txt')
#locarna = read_file('locarna.txt')


inf = calculate_inf(turbofold, locarna)

print(f"INF: {inf:.2f}")
```

```
INF: 0.36
```

Zaimplementuj algorytm wyznaczania odległości Levenshteina dla struktur w formacie dot-bracket. Do raportu dołącz kod oraz wyniki dla struktur otrzymanych wcześniej (ocena 4.5)

[18]:
```
def levenshtein(s, t):
    m, n = len(s), len(t)
    d = [[0] * (n+1) for _ in range(m+1)]

    for i in range(m+1):
        d[i][0] = i
    for j in range(n+1):
        d[0][j] = j

    for i in range(1, m+1):
        for j in range(1, n+1):
            cost = 0 if s[i-1] == t[j-1] else 1
            d[i][j] = min(d[i-1][j] + 1,
                          d[i][j-1] + 1,
                          d[i-1][j-1] + cost)

    return d[m][n]

result = levenshtein(turbofold, locarna)
print(f"Levenshtein distance: {result}")
```

```
Levenshtein distance: 13
```