**COMP9101 Ass03**

**Name: Zanning Wang**

**zID: z5224151**

文本

中度可信度描述已自动生成

**1.**

In order to solve this problem, we may use dynamic programming to solve this problem.

Suppose we have T(i, j), which means we have the number of ways to parenthesize the symbols from i to j to get TRUE result. The same for F(i, j), we have the number of ways to parenthesize the symbols from i to j to get FALSE result.

For the base case, we use cur\_sym to show the result of current symbol, therefore,

When cur\_sym[i] === True, obviously T[i][i] = 1, F[i][i] = 0;

When cur\_sym[i] === False, obviously F[i][i] = 0, F[i][i] = 1;

For T(i, j), the problem can be divide into the sub-problem T(i, k) and T(k+1, j), k∈[i, j];

For F(i, j), the problem can be divide into the sub-problem F(i, k) and F(k+1, j), k∈[i, j];

Suppose the symbol is “AND”:

In order to get the TRUE result, the result of each sub-problem should be TRUE

Therefore the T(i, j) = T(i, k) \* T(k+1, j), which k∈[i, j];

Suppose the symbol is “OR”:

In order to get TRUE result, at least one of sub-problem should be TRUE is enough

Both TRUE: T(i, j) => T(i, k) \* T(k+1, j), which k∈[i, j];

One TRUE: T(i, j) => T(i, k) \* F(k+1, j) + F(i, k) \* T(k+1, j), which k∈[i, j];

Therefore the T(i, j) = T(i, k) \* T(k+1, j) + T(i, k) \* F(k+1, j) + F(i, k) \* T(k+1, j), which k∈[i, j];

Suppose the symbol is “NAND”:

According to the definition, only TRUE NAND TRUE will get the False, all other situations will get the TRUE, therefore the result show below:

T(i, j) = F(i, k) \* T(k+1, j) + T(i, k) \* F(k+1, j) + F(i, k) \* F (k+1, j);

Suppose the symbol is “NOR”:

According to the definition, only TRUE NAND TRUE will get the TRUE;

Therefore T(i, j) = F(i, k) \* F(k+1, j);

Therefore the recursion show bolow:

The same for F(i, j), we can get the recursion show bolow:

Therefore we can count the number of TRUE result by T(i, j) show above.