ДП на подотрезках

Основные принципы

- Подотрезки (dp[l][r])
- Короткие отрезки
- а) Разрезаем отрезок на 2 части

  б) Отрезаем границу
- Ленивая динамика
- ДП от всего массива

Задача 1

Дана строка, состоящая из N символов (будем считать, что символы нумеруются с 0). Найти длину максимального подпалиндрома этой строки.

Примеры:

abca – 3 (aba, aca)

abcd – 1 (a, b, c, d)

abba – 4 (abba)

Задача 1

1. dp[l][r] – длина максимального подпалиндрома, состоящего из некоторых символов подстроки s_l, ..., s_r.

Задача 1

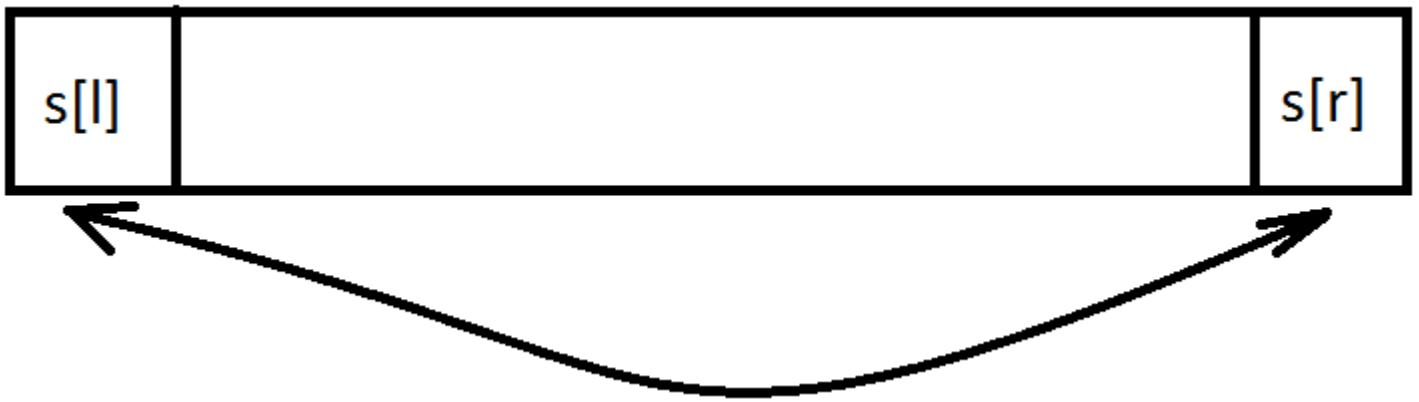
1. dp[l][r] – длина максимального подпалиндрома, состоящего из некоторых символов подстроки s_l, …, s_r.
2. dp[i][i] = 1, l = 0, …, n – 1, где n – длина строки

Задача 1

1. dp[l][r] – длина максимального подпалиндрома, состоящего из некоторых символов подстроки s_l, …, s_r.
2. dp[i][i] = 1, i = 0, …, n – 1, dp[i][j] = 0, i > j.

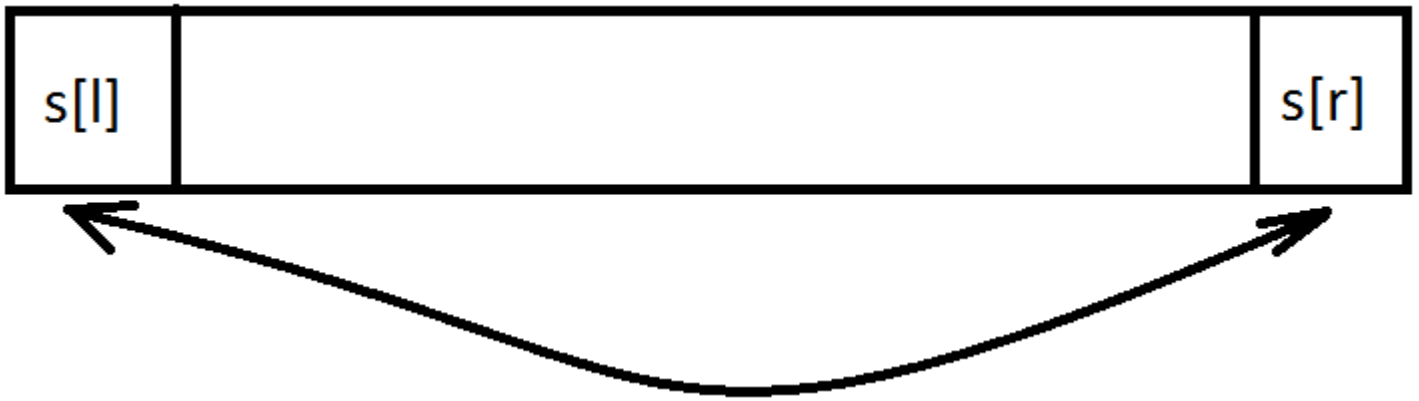
Задача 1

3. Рассмотрим подотрезок s_l, …, s_r



Задача 1

3. Рассмотрим подотрезок s_l, …, s_r



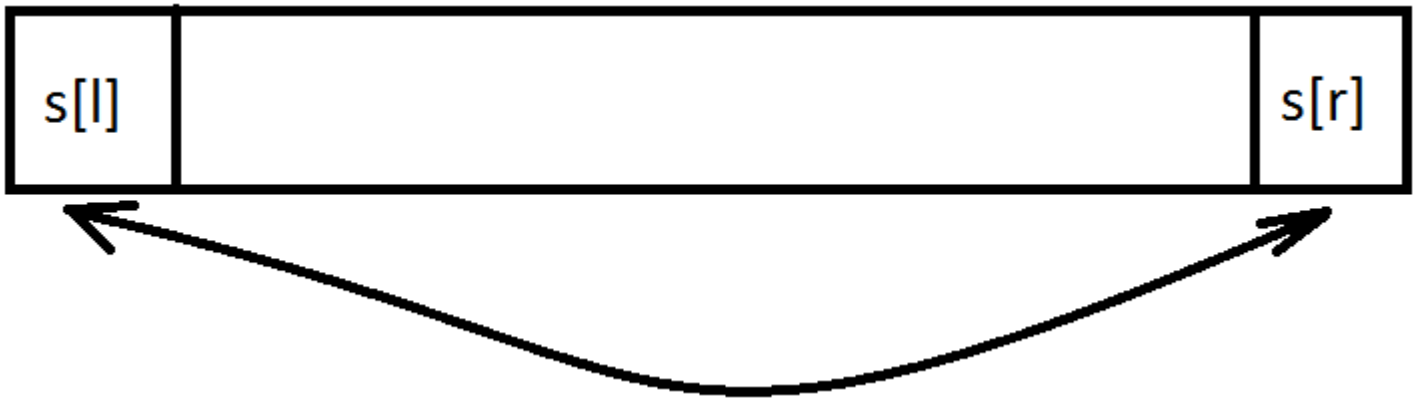
2 варианта:

а) s[l] = s[r]

б) s[l] ≠ s[r]

Задача 1

3. Рассмотрим подотрезок s_l, …, s_r



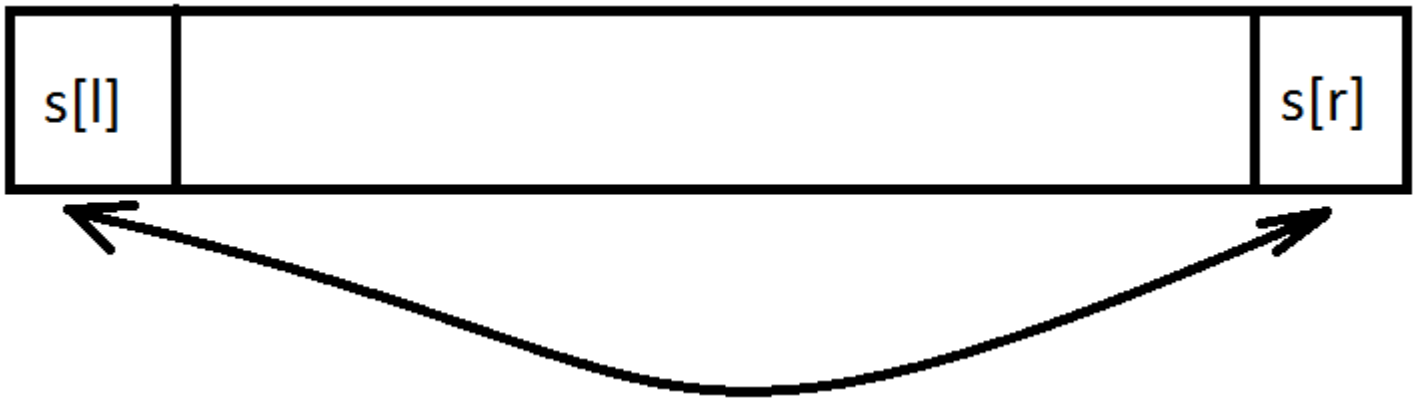
2 варианта:

а) s[l] = s[r] => dp[l][r] = 2 + dp[l + 1][r - 1]

б) s[l] $\neq$ s[r]

Задача 1

3. Рассмотрим подотрезок s_l, …, s_r



2 варианта:

а) s[l] = s[r] => dp[l][r] = 2 + dp[l + 1][r – 1]

б) s[l]≠s[r] => dp[l][r] = max(dp[l + 1][r], dp[l][r – 1])

Задача 1

Остается реализовать вычисление отрезков большей длины через отрезки меньшей длины (длины, меньшей либо на 1, либо на 2). Мы знаем значения для отрезков длины 1 и длины 0. Организуем перебор отрезков в порядке возрастания их длины: сначала отрезки длины 2, потом отрезки длины 3 и так далее.

Как это сделать?

Как это сделать?

Простым двойным циклом:

Как это сделать?

Простым двойным циклом:

```
for(int len = 2; len <= n; ++len){
    for(int l = 0; l < n - len + 1; ++l){
        int r = l + len - 1;
        // ... рассматриваем подотрезки ...
    }
}
```

Альтернативный способ

Перебор левой границы справа налево, правой – слева направо:

```
for(int l = n - 1; l >= 0; --l){
    for(int r = l; r < n; ++r){
        считаем динамику
    }
}
```

Как применить это в задаче?

Как применить это в задаче?

```
for(int len = 2; len <= n; ++len){
    for(int l = 0; l < n - len + 1; ++l){
        int r = l + len - 1;
        dp[l][r] = max(dp[l + 1][r], dp[l][r - 1]);
        if (s[l] == s[r]){
            dp[l][r] = 2 + dp[l + 1][r - 1];
        }
    }
}
```

Где хранится ответ?

```
for(int len = 2; len <= n; ++len){
    for(int l = 0; l < n - len + 1; ++l){
        int r = l + len - 1;
        dp[l][r] = max(dp[l + 1][r], dp[l][r - 1]);
        if (s[l] == s[r]){
            dp[l][r] = 2 + dp[l + 1][r - 1];
        }
    }
}
```

Где хранится ответ?

```
for(int len = 2; len <= n; ++len){
    for(int l = 0; l < n - len + 1; ++l){
        int r = l + len - 1;
        dp[l][r] = max(dp[l + 1][r], dp[l][r - 1]);
        if (s[l] == s[r]){
            dp[l][r] = 2 + dp[l + 1][r - 1];
        }
    }
}
```

Ответ хранится в dp[0][n - 1];

Задача 2

Дана строка, для каждой её непрерывной подстроки узнать, является ли она палиндромом.

Задача 2

Дана строка, для каждой её непрерывной подстроки узнать, является ли она палиндромом.

is_pal[l][r] = 0, если подстрока s_l, …, s_r не является палиндромом

is_pal[l][r] = 1, в противном случае

Задача 2

Дана строка, для каждой её непрерывной подстроки узнать, является ли она палиндромом.

Задача 2

```
int n = s.size();
vector < vector <char> > is_pal(n, vector <char> (n, 0));
// рассматриваем подотрезки длины 1
for(int i = 0; i < n; i++){
    is_pal[i][i] = 1;
}
// рассматриваем подотрезки длины 2
for(int i = 0; i < n - 1; i++){
    is_pal[i][i + 1] = (s[i] == s[i + 1]);
}
// рассматриваем остальные подтрезки
for(int i = 2; i < n; i++){
    for(int j = 0; j < n - i; j++){
        if (s[j] == s[i + j] && is_pal[j + 1][i + j - 1]){
            is_pal[j][i + j] = 1;
        }
    }
}
```

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя всё просто перемножить?

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя всё просто перемножить?

Примеры:

0 2 3

1 2 3

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя всё просто перемножить?

Примеры:

0 2 3

1 2 3

1 * 2 * 3 = 6

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя всё просто перемножить?

Примеры:

0 2 3

1 2 3

1 * 2 * 3 = 6

(1 + 2) * 3 = 9

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Пример:

1 3 1 5

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Пример:

1 3 1 5

1 + 3 1 5

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Пример:

1 3 1 5

1 + 3 + 1 5

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Пример:

1 3 1 5

(1 + 3 + 1) * 5 = 25

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Контрпример:

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Контрпример:

1 1 1 1 1 1 4

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Контрпример:

1 1 1 1 1 1 4

(1 + 1) * (1 + 1) * (1 + 1) * 4 = 32

Задача 3

Дана последовательность, состоящая из n неотрицательных чисел. Требуется расставить знаки +,× и скобки в последовательности, чтобы значение полученного арифметического выражения было максимальным.

Почему нельзя сделать так:

Прибавляем единицу к тому числу, которое меньше, заменяем число на (число + 1), а всё остальное перемножаем?

Контрпример:

1 1 1 1 1 1 4

(1 + 1) * (1 + 1) * (1 + 1) * 4 = 32

(1 + 1 + 1) * (1 + 1 + 1) * 4 = 36

Решение

- dp[l][r] – максимальное значение выражения, которое можно получить, действуя указанным способом.

Решение

- dp[l][r] – максимальное значение выражения, которое можно получить, действуя указанным способом.
- dp[i][i] = a[i], i = 1, …, n

Решение

- dp[l][r] – максимальное значение выражения, которое можно получить, действуя указанным способом.
- dp[i][i] = a[i], i = 1, …, n

  Переберем позицию последней арифметической операции на отрезке.

Решение

- dp[l][r] – максимальное значение выражения, которое можно получить, действуя указанным способом.
- dp[i][i] = a[i], i = 1, …, n

  Переберем позицию последней арифметической операции на отрезке.

  Тогда:

  dp[i][j] = max(dp[i][k] + dp[k + 1][j], dp[i][k] * dp[k + 1][j]), k = 1, …, j – 1

Решение

- dp[l][r] – максимальное значение выражения, которое можно получить, действуя указанным способом.
- dp[i][i] = a[i], i = 1, …, n

  Переберем позицию последней арифметической операции на отрезке.

  Тогда:

  dp[i][j] = max(dp[i][k] + dp[k + 1][j], dp[i][k] * dp[k + 1][j]), k = 1, …, j – 1

  Ответ хранится в dp[1][n]

Задача 4

Дана последовательность, состоящая из круглых и квадратных скобок. Дополнить её минимальным число скобок так, чтобы скобочная последовательность оказалась правильной.

Правила ПСП

1. Пустая последовательность – ПСП
2. S – ПСП => [S], (S) – тоже ПСП
3. A – ПСП, B – ПСП => AB и BA – тоже ПСП

Решение

dp[l][r] – минимальное число скобок, которое надо добавить, чтобы скобочная подпоследовательность s_l, …, s_r оказалась правильной.

Решение

dp[l][r] – минимальное число скобок, которое надо добавить, чтобы скобочная подпоследовательность s_l, …, s_r оказалась правильной.

dp[i][i] = 1, l = 0, …, n – 1.

dp[l][r] = 0, l > r.

Решение

1. Разбиваем нашу последовательность на 2 части.

Решение

1. Разбиваем нашу последовательность на 2 части.
2. Находим ответ для каждой из 2 частей

Решение

1. Разбиваем нашу последовательность на 2 части.
2. Находим ответ для каждой из 2 частей
3. Складываем получившиеся значения

Решение

1. Разбиваем нашу последовательность на 2 части.
2. Находим ответ для каждой из 2 частей
3. Складываем получившиеся значения
4. Находим минимальное из получившихся значений

Решение

1. Разбиваем нашу последовательность на 2 части.
2. Находим ответ для каждой из 2 частей
3. Складываем получившиеся значения
4. Находим минимальное из получившихся значений

Как разбивать?

1. Стоит заметить, что некоторые отрезки будут рассмотрены несколько раз.

Как разбивать?

1. Стоит заметить, что некоторые отрезки будут рассмотрены несколько раз.
2. Чтобы не рассматривать каждый отрезок несколько раз, воспользуемся мемоизацией (запоминаем ответ и больше отрезок не рассчитываем)

Как разбивать?

1. Стоит заметить, что некоторые отрезки будут рассмотрены несколько раз.
2. Чтобы не рассматривать каждый отрезок несколько раз, воспользуемся мемоизацией (запоминаем ответ и больше отрезок не рассчитываем)
3. Для это для всех $l < r$ установим $dp[l][r] = \infty$

Как разбивать?

1. Стоит заметить, что некоторые отрезки будут рассмотрены несколько раз.
2. Чтобы не рассматривать каждый отрезок несколько раз, воспользуемся мемоизацией (запоминаем ответ и больше отрезок не рассчитываем)
3. Для это для всех $l < r$ установим $dp[l][r] = \infty$
4. Удобно установить $dp[l][r] = -1$, $l < r$

Как разбивать?

1. Стоит заметить, что некоторые отрезки будут рассмотрены несколько раз.
2. Чтобы не рассматривать каждый отрезок несколько раз, воспользуемся мемоизацией (запоминаем ответ и больше отрезок не рассчитываем)
3. Для это для всех $l < r$ установим $dp[l][r] = \infty$
4. Удобно установить $dp[l][r] = -1$, $l < r$

Как разбивать?

1. Разбиение на 2 части будем производить следующим образом:

   если s[l] и некоторое s[i] являются скобками одного типа (s[l] – открывающая, s[i] – закрывающая), то разбиваем нашу последовательность на s_l, ..., s_i и s_{i+1}, ..., s_r.

Как разбивать?

1. Разбиение на 2 части будем производить следующим образом:

   если s[l] и некоторое s[i] являются скобками одного типа (s[l] – открывающая, s[i] – закрывающая), то разбиваем нашу последовательность на s_l, ..., s_i и s_{i+1}, ..., s_r.

2. Для каждой считаем ответ и суммируем.

Как разбивать?

1. Разбиение на 2 части будем производить следующим образом:

   если s[l] и некоторое s[i] являются скобками одного типа (s[l] – открывающая, s[i] – закрывающая), то разбиваем нашу последовательность на s_l, ..., s_i и s_{i+1}, ..., s_r.
2. Для каждой считаем ответ и суммируем.
3. Ответ будет храниться в dp[0][n – 1]

Реализация

Реализация

```
int dp[113][113];
string s;
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){




}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];









}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){









    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);









    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);
        if (s[l] == '[' || s[l] == '('){









        }

    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);
        if (s[l] == '[' || s[l] == '('){
            for (int i = l + 1; i <= r; i++){



            }
        }

    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);
        if (s[l] == '[' || s[l] == '('){
            for (int i = l + 1; i <= r; i++){
                if ((s[l]=='[' && s[i]==']') || (s[l]=='(' && s[i]==')')){
                    ans = min(ans, solve(l + 1, i - 1) + solve(i + 1, r));
                }
            }
        }

    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);
        if (s[l] == '[' || s[l] == '('){
            for (int i = l + 1; i <= r; i++){
                if ((s[l]=='[' && s[i]==']') || (s[l]=='(' && s[i]==')')){
                    ans = min(ans, solve(l + 1, i - 1) + solve(i + 1, r));
                }
            }
        }
        dp[l][r] = ans;
    }

}
```

Реализация

```
int dp[113][113];
string s;

int solve(int l, int r){
    if (l > r)
        return 0;
    int ans = dp[l][r];
    if (ans < 0){
        ans = 1 + solve(l + 1,r);
        if (s[l] == '[' || s[l] == '('){
            for (int i = l + 1; i <= r; i++){
                if ((s[l]=='[' && s[i]==']') || (s[l]=='(' && s[i]==')')){
                    ans = min(ans, solve(l + 1, i - 1) + solve(i + 1, r));
                }
            }
        }
        dp[l][r] = ans;
    }
    return ans;
}
```