Динамика на дереве.

Динамическое программирование на дереве заключается в вычислении некоторой информации для каждой вершины дерева. Например, это может быть глубина вершины или размер ее поддерева.

Заполнять динамику можно снизу-вверх или сверху-вниз, в зависимости от задачи. Если динамика заполняется снизу-вверх, то переход динамики выполняется после вызовы dfs от сына, а если сверху-вниз, то перед вызовом dfs.

Примером заполнения динамики сверху-вниз является вычисление глубины вершины. Для корня глубина считается равной нулю. Код ниже решает эту задачу.

```
void dfs(int v) {
    used[v] = 1;
    for (int i = 0; i < g[v].size(); ++i) {
        int to = g[v][i];
        if (!used[to]) {
            dep[to] = dep[v] + 1;
            dfs(to);
        }
    }
}
```

Примером заполнения динамики снизу-вверх является вычисление размера поддерева вершины. Код ниже решает эту задачу.

```
void dfs(int v) {
    used[v] = 1;
    sz[v] = 1;
    for (int i = 0; i < g[v].size(); ++i) {
        int to = g[v][i];
        if (!used[to]) {
            dfs(to);
            sz[v] += sz[to];
        }
    }
}
```

В двух примерах, рассмотренных выше, мы заполняли одномерную динамику. Динамика на дереве может иметь и более высокую размерность. Давайте рассмотрим пример двумерной динамики.

Задача. Дано дерево из n вершин ($1 \le n \le 2000$). Требуется для всех вершин v дерева и всех значений числа $0 \le k < n$ найти количество вершин в поддереве вершины v, находящихся на расстоянии не более чем k от вершины v.

Через $dp[v][k]$ обозначим искомую величину. Код ниже решает поставленную задачу.

```
void dfs(int v) {
    used[v] = 1;
    for (int j = 0; j < n; ++j)
        dp[v][j] = 1;
    for (int i = 0; i < g[v].size(); ++i) {
        int to = g[v][i];
        if (!used[to]) {
            dfs(to);
            for (int j = 1; j < n; ++j)
                dp[v][j] += dp[to][j - 1];
        }
    }
}
```