1. Общие принципы динамики.

1.1) Граф состояний. Пять основных пунктов динамики

1.1.1) Состояние динамики: параметр(ы), однозначно задающие подзадачу.

1.1.2) Значения начальных состояний.

1.1.3) Переходы между состояниями: формула пересчёта.

1.1.4) Порядок пересчёта.

1.1.5) Положение ответа на задачу: иногда это сумма или, например, максимум из значений нескольких состояний.

1.2) Динамика вперед и динамика назад(Нисходящее и восходящее динамическое программирование).

Нисходящее динамическое программирование: задача разбивается на подзадачи меньшего размера, они решаются и затем комбинируются для решения исходной задачи. Используется запоминание для решений часто встречающихся подзадач.

Восходящее динамическое программирование: все подзадачи, которые впоследствии понадобятся для решения исходной задачи просчитываются заранее и затем используются для построения решения исходной задачи. Этот способ лучше нисходящего программирования в смысле размера необходимого стека и количества вызова функций, но иногда бывает нелегко заранее выяснить, решение каких подзадач нам потребуется в дальнейшем.

1.3) Правильный подбор аргументов и значения динамики.

Важно отличать аргументы и пересчитываемое значение динамики. Совсем не всегда пересчет динамики является ответом на задачу: бывает, что искомое значение нужно задать в качестве аргумента для лучшего пересчета.

2. Динамика по поддеревьям

2.1) Общая постановка задач.

Параметром состояния динамики по поддеревьям обычно бывает вершина, обозначающая поддерево, в котором эта вершина – корень. Для получения значения текущего состояния обычно нужно знать результаты всех своих детей. Чаще всего реализуют лениво – просто пишут поиск в глубину из корня дерева.

Пусть задано дерево корневое T на n вершинах. Подвесим T за какую-нибудь вершину r. Обозначим $Ch(x)$ как множество сыновей вершины x. А $lca(u, v)$ – это самый далёкий от корня общий предок вершин u и v.

2.2) Пример задачи: Сумма длин всех путей в дереве.

Пусть P путь в дереве между вершинами u и v. Существуют 2 случая:

1) $lca(u, v) \in \{u, v\}$

2) $lca(u, v) \notin \{u, v\}$, тогда P можно разбить на 2 части, каждая из которых имеет первый вид.

s_u — размер поддерева u.
d_u — сумма длин всех путей между u и вершинами в поддереве u.
r_u — сумма длин всех путей между вершинами a и b, для которых $lca(a, b) = u$.

Если вершина u лист, то $s_u = 1$ и $d_u = r_u = 0$,
в противном случае:
$s_u = 1 + \sum\limits_{v \in Ch(u)} s_v$
$d_u = \sum\limits_{v \in Ch(u)} (d_v + s_v) = s_u - 1 + \sum\limits_{v \in Ch(u)} d_v$
$r_u = d_u + \sum\limits_{\substack{v,w \in Ch(u) \\ v \neq w}} (s_v(d_w + s_w)) = d_u + \sum\limits_{v \in Ch(u)} s_v(d_u - (d_v + s_v)) =$
$= d_u s_u - \sum\limits_{v \in Ch(u)} s_v(d_v + s_v)$
Ответом будет $\sum\limits_{u=1}^{n} r_u$. Время работы $O(\sum\limits_{u=1}^{n} |Ch(u)|) = O(n)$.
Альтернативное решение: для каждого ребра мы можем посчитать сколько раз оно войдет в сумму. Это количество вершин слева, умноженное на количество вершин справа.

2.3*) Пример задачи: Задача о вершинном покрытии в случае деревьев.
Вершинное покрытие для неориентированного графа $G = (V, E)$ это множество его вершин S, такое что, у каждого ребра графа хотя бы один из концов входит в S. Решение в случае деревьев состоит в применение динамики по поддеревьям.

$a_{u,0}$ — размер минимального вершинного покрытия поддерева с корнем в вершине u среди всех покрытий, содержащих вершину u.
$a_{u,1}$ — размер минимального вершинного покрытия поддерева с корнем в вершине u среди всех покрытий, не содержащих вершину u.

Если вершина u лист, то $a_{u,0} = 0$ и $a_{u,1} = 1$,
в противном случае:
$a_{u,0} = \sum\limits_{v \in Ch(u)} a_{v,1}$
$a_{u,1} = 1 + \sum\limits_{v \in Ch(u)} \min(a_{v,0}, a_{v,1})$
Ответом будет $\min(a_{r,0}, a_{r,1})$. Время работы $O(\sum\limits_{u=1}^{n} |Ch(u)|) = O(n)$.

3. Стандартные задачи на динамику.

3.1) НВП. НОП.

Наибольшая возрастающая подпоследовательность: Дана последовательность $a_1, ..., a_n$. Требуется найти ее возрастающую подпоследовательность наибольшей длины.

d_i — длина наибольшей возрастающей подпоследовательности, оканчивающейся в элементе с индексом i. Для удобства, будем считать, что: $a_0 = -\infty$ и $d_0 = 0$. Тогда:

$d_i = 1 + \max\limits_{\substack{j=0,...,i-1 \\ a[j]<a[i]}} d_j$

Будем считать d_i слева направо. Ответ: $\max\limits_{1\le i\le n} d_i$. Время работы $O(n^2)$.

Наибольшая общая подпоследовательность: Даны две последовательности $a_1, ..., a_n$ и $b_1, ..., b_m$. Требуется найти их наибольшую общую подпоследовательность.

$f_{i,j}$ — длина наибольшей общей подпоследовательности префиксов a и b, длины i и j соответсвенно.

$f_{0,j} = f_{i,0} = 0$
$f_{i,j} = 1 + f_{i-1,j-1}$ если $a[i] = b[j]$
$f_{i,j} = \max(f_{i-1,j}, f_{i,j-1})$ если $a[i] \neq b[j]$

Ответ: $f_{n,m}$. Время работы: $O(nm)$.

3.2*) НВП за O(N log N)

d_i — число, на которое оканчивается возрастающая подпоследовательность длины i (а если таких чисел несколько — то наименьшее из них).

Изначально мы полагаем $d_0 = -\infty$, а все остальные элементы $d_i = \infty$. Считать эту динамику мы будем постепенно, обработав число a_1, затем a_2, и т.д.

Для обработки числа a_i, найдем наименьшее число j, для которого выполняется $a_i < d_j$ и поставим $d_j = a_i$.

Заметим, что в любой момент времени $d_{i-1} < d_i, 1 \le i \le n$, и это означает, что последний шаг можно делать бинарным поиском.

Ответ: $\max\limits_{\substack{1\le i\le n \\ d_i\neq\infty}} i$. Время работы: без бинарного поиска $O(n^2)$, с бинарным поиском $O(n \log n)$.

3.3) Пример задачи на рюкзак: дана очередь, но некоторые люди не знают, за кем они стоят(таким образом, очередь разбивается на цепочки). Требуется для конкретного человека определить все возможные варианты, когда он станет первым.

Для начала нужно перебрать все варианты перестановок этой цепочки. Точнее, нас интересует лишь количество человек перед цепочкой, в которой стоит наш x-ый человек.

Это все эти варианты считаются с помощью рюкзака. Всё, что потом остается – к возможным значениям добавить порядок нашего человека в очереди.

4. Динамика на подотрезках

4.1) Общая постановка и принцип решения задачи.

Это класс динамики, в котором состояние — это границы подотрезка какого-нибудь массива. Суть в том, чтобы подсчитать ответы для подзадач, основывающихся на всех возможных подотрезках нашего массива. Обычно перебираются они в порядке увеличения длины, и пересчёт основывается, соответственно на более коротких отрезках.

4.2) Пример задачи: посчитать количество подпалиндромов.

Дана строка s длины n. Требуется посчитать количество подпалидромов этой строки.

$p_{i,j}$ — количество подпалиндромов подстроки $s[i...j]$ $(i \leq j)$.

$p_{i,i} = 1, 1 \leq i \leq n$
$p_{i,i+1} = 2 + [s[i] = s[i+1]], 1 \leq i < n$
$p_{i,j} = p_{i+1,j} + p_{i,j-1} - p_{i+1,j-1} + [s[i] = s[j]](p_{i+1,j-1} + 1)$

где [утверждение] определяется как 1, если утверждение истинно, и 0 в противном случае.

Ответ: $p_{1,n}$. Время работы: $O(n^2)$.

4.3) Пример задачи: дана последовательность букв, которая является обходом какого-то дерева(считаем, что на ребрах дерева записаны буквы). Обходом считаем рекурсивный проход по вершинам из корня дерева, который постоянно идет в "самую левую"непосещенную вершину. Посчитать количество возможных деревьев.

Пример для понимания:

Строке $ABBA$ соответствует только одно дерево(Из трех вершин, ребра идут друг за другом из корня). А строке $AAAA$ – два дерева.

Тогда ответом на задачу будет

$p_{i,i-1} = 1, 1 < i \leq n$ (Пустое поддерево)
$p_{i,i} = 0, 1 \leq i \leq n$
$p_{i,i+1} = [s[i] = s[i+1]], 1 \leq i < n$
$p_{i,j} = \sum\limits_{i<t\leq j} [s[i] = s[j]](d[i+1][t-1] * d[t+1][j])$

Ответ: $p_{1,n}$. Время работы: $O(n^3)$.