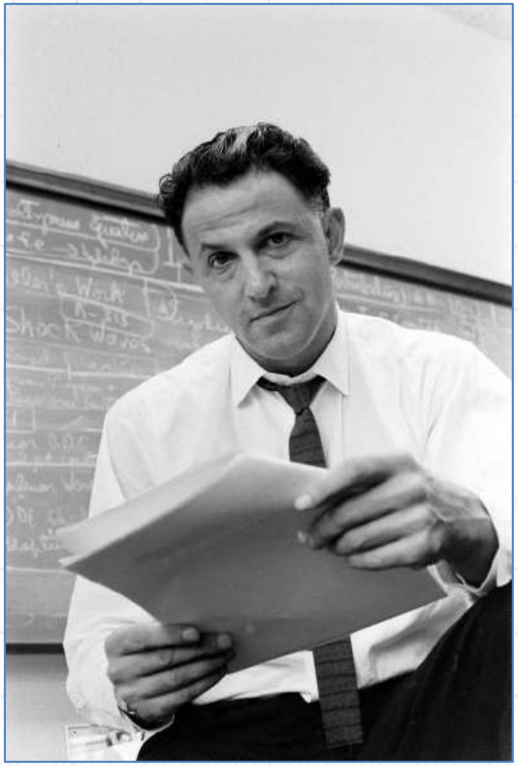
18.12.2021

Тема доклада:

Динамическое программирование. Классические задачи



Основоположник динамического программирования

Ричард Беллман (1920—1984)

I. ВВЕДЕНИЕ

Среди переборных и некоторых других задач можно выделить класс задач, обладающих одним хорошим свойством: имея решения некоторых ***подзадач*** (например, для меньшего числа *n*), можно практически без перебора найти ***решение исходной задачи***. Такие задачи решают методом динамического программирования.

Под динамическим программированием понимают ***сведение задачи к подзадачам***.

II. ПОСЛЕДОВАТЕЛЬНОСТИ

Задача №1.
Последовательность Фибоначчи F_n задается формулами: $F_1 = 1$, $F_2 = 1$, $F_n = F_{n-1} + F_{n-2}$ при $n > 1$.
Необходимо найти F_n по номеру n.

```
int F(int n) {
 if (n < 2) return 1;
 else return F(n - 1) + F(n - 2);
}
```

- Решение с помощью **рекурсии**
- Решение «с конца»
- Число операций нарастает экспоненциально

- Решение с помощью **ДП**
- Решение «с начала»
- Сначала решили все подзадачи

```
F[0] = 1; F[1] = 1;
for (i = 2; i < n; i++)
    F[i] = F[i - 1] + F[i - 2];
```

III. ОДНОМЕРНОЕ ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Пусть исходная задача заключается в нахождении некоторого числа T при исходных данных $n_1, n_2, \ldots, n_k$.

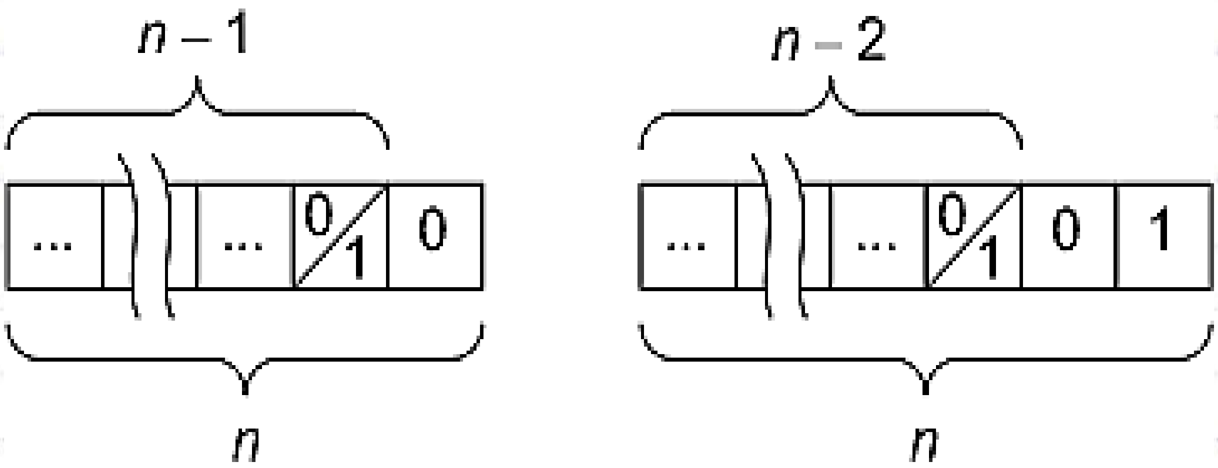
Имеем функцию $T(n_1, n_2, \ldots, n_k)$, значение которой и есть необходимый ответ.

Тогда подзадачами будем считать задачи $T(i_1, i_2, \ldots, i_k)$ при $i_1 < n_1, i_2 < n_2, \ldots, i_k < n_k$.

Одномерное ДП: $k = 1$
Двумерное ДП: $k = 2$
Многомерное ДП: $k > 2$

Задача №2. Посчитать число последовательностей нулей и единиц длины *n*, в которых не встречаются две идущие подряд единицы.

При $n = 1$, $n = 2$ ответ очевиден.
Пусть K_{n-1}, K_{n-2} — число таких последовательностей длины $n - 1$ и $n - 2$.

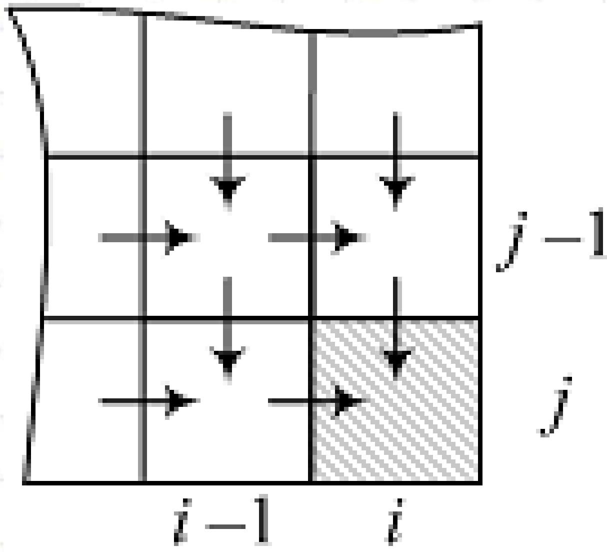


$K_1 = 2$
$K_2 = 3$

$$K_n = K_{n-1} + K_{n-2} \text{ при } n > 2$$

IV. ДВУМЕРНОЕ ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Задача №3. Дано прямоугольное поле размером *n***m* клеток. Можно совершать шаги длиной в одну клетку вправо или вниз. Посчитать, сколькими способами можно попасть из левой верхней клетки в правую нижнюю.



В клетку с координатами **(*i*,*j*)** можно прийти только сверху или слева ⇒ из клеток с координатами **(*i* – 1, *j*)** и **(*i*, *j* – 1)**

A[i][j] = A[i – 1][j] + A[i][j – 1]

A[0][0] = 1

18.12.2021

Спасибо за внимание