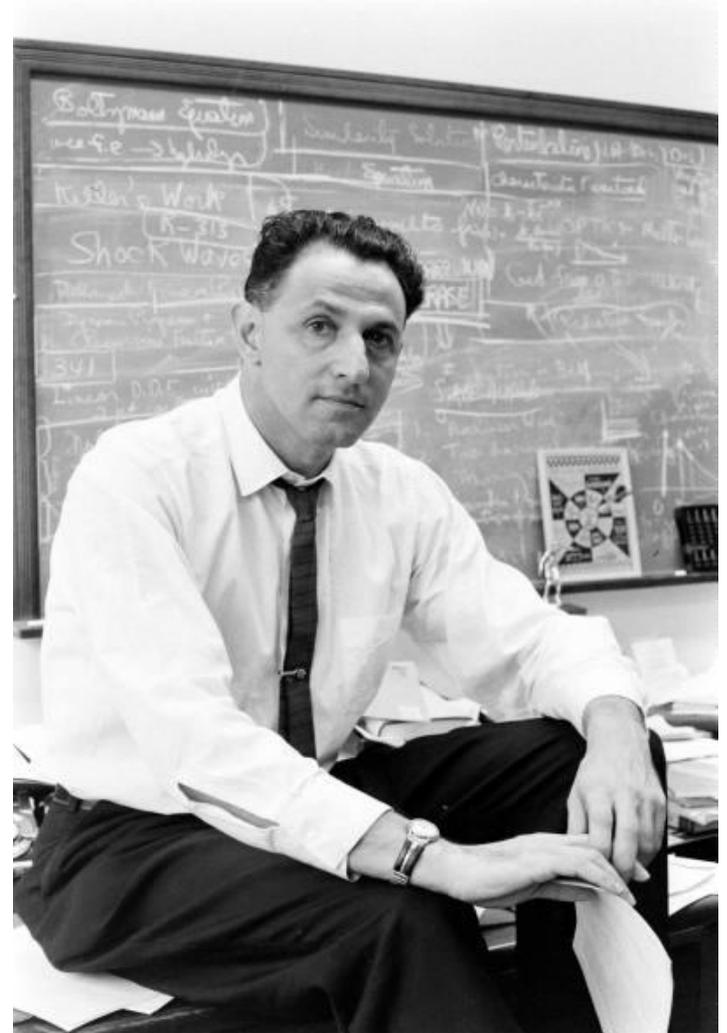


Динамическое программирование

Ричард Беллман

- **Ричард Эрнст Беллман** (англ. *Richard Ernest Bellman*; 1920—1984) — американский математик, один из ведущих специалистов в области математики и вычислительной техники.



Последовательность Фибоначчи

- Последовательность Фибоначчи F_n задается формулами: $F_1 = 1$, $F_2 = 1$,
 $F_n = F_{n-1} + F_{n-2}$ при $n > 2$.
- Необходимо найти F_n по номеру n .

Рекурсия

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

# Сохранение промежуточных результатов

```
int F(int n) {
 if (A[n] != -1) return A[n];
 if (n < 2) return 1;
 else {
 A[n] = F(n - 1) + F(n - 2);
 return A[n];
 }
}
```

# Самое простое решение

```
F[0] = 1;
```

```
F[1] = 1;
```

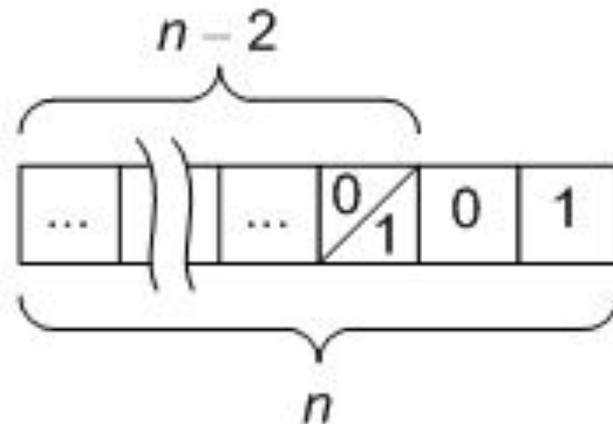
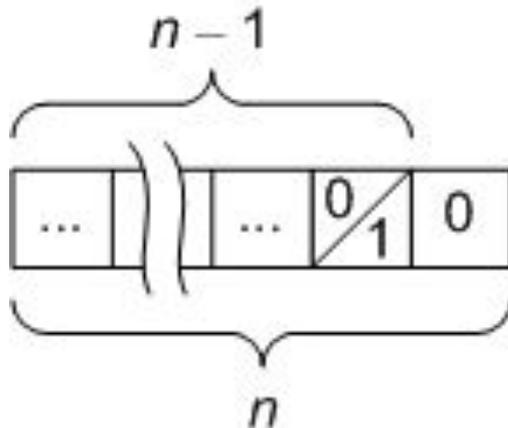
```
for (i = 2; i < n; i++) {
```

```
 F[i] = F[i - 1] + F[i - 2];
```

```
}
```

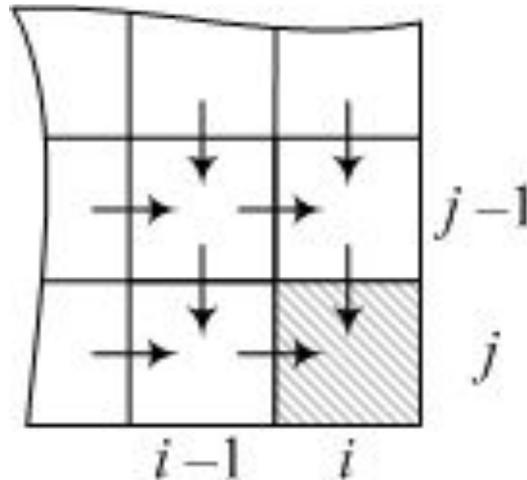
# Одномерное динамическое программирование

- **Задача 1.** Посчитать число последовательностей нулей и единиц длины  $n$ , в которых не встречаются две идущие подряд единицы.



# Двумерное динамическое программирование

- **Задача 2.** 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.



# Задача о рюкзаке

- Имеется набор из  $N$  предметов, каждый предмет имеет массу  $W_i$  и стоимость  $P_i$ ,  $i=(1,2..N)$ , требуется собрать набор с максимальной полезностью таким образом, чтобы он имел вес не больше  $W$ , где  $W$  – вместимость ранца.  $W_i$ ,  $P_i$ ,  $W$  – целые неотрицательные числа.

# Методы

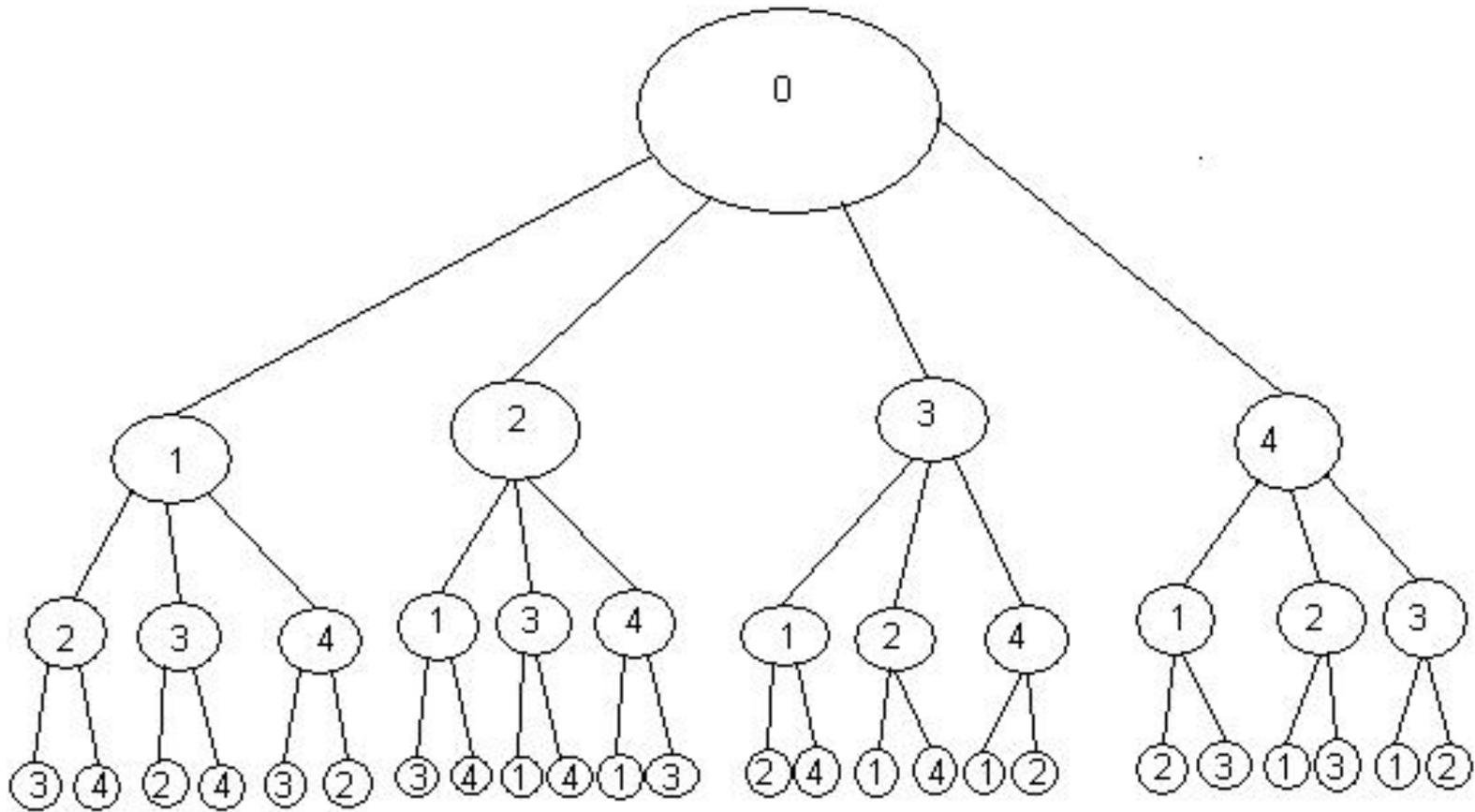
- Полный перебор
- Динамическое программирование
- Метод ветвей и границ
- Жадный алгоритм

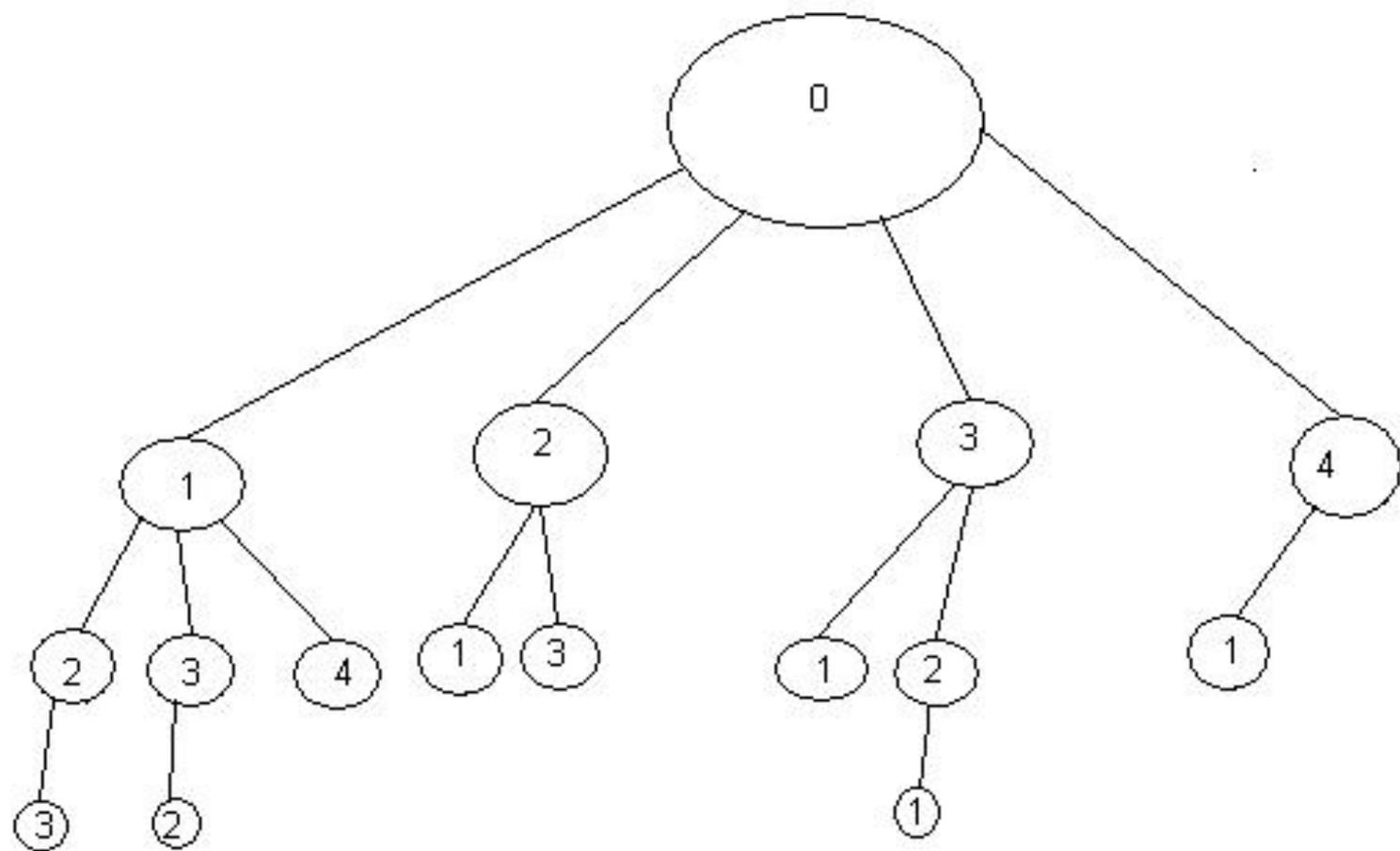
# Динамическое программирование

- Value [W, N] – максимальная сумма, которую надо найти.
- Суть метода– на каждом шаге по весу  $1 < W_i < W$  находим максимальную загрузку Value[W<sub>i</sub>, i], для веса W<sub>i</sub>. Допустим мы уже нашли Value[1..W, 1..i-1], то есть для веса меньше либо равного W и с предметами, взятыми из 1..i-1. Рассмотрим предмет i, если его вес W<sub>i</sub> меньше W проверим стоит ли его брать.

- Если его взять то вес станет  $W - W_i$  , тогда  $Value[W, i] = Value[W - W_i, i-1] + P_i$  (для  $Value[W - W_i, i-1]$  решение уже найдено остается только прибавить  $P_i$ ).
- Если его не брать то вес останется тем же и  $Value[W, i] = Value[W, i-1]$ .
- Из двух вариантов выбирается тот, который дает наибольший результат.

# Метод ветвей и границ





# Жадный алгоритм

| Номер предмета<br>(i) | Вес предмета (кг) | Цена (У.е) | Относительная<br>цена (У.е/кг) |
|-----------------------|-------------------|------------|--------------------------------|
| 1                     | 10                | 60         | 6                              |
| 2                     | 20                | 100        | 5                              |
| 3                     | 30                | 120        | 4                              |