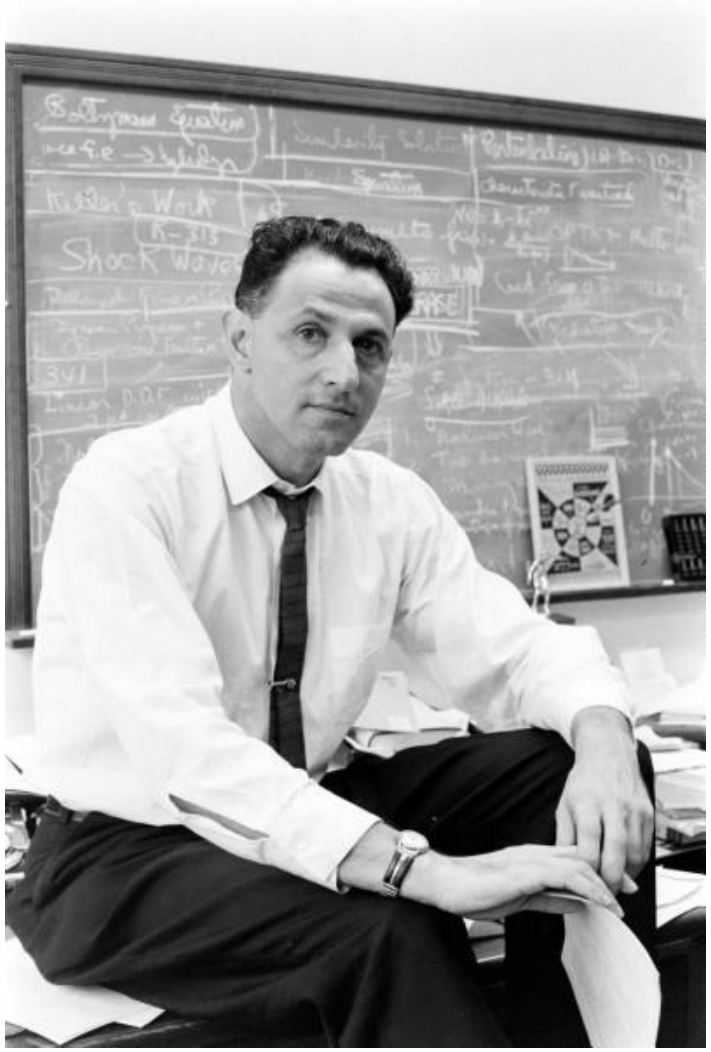
Динамическое программирование

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 $n > 1$.
- 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

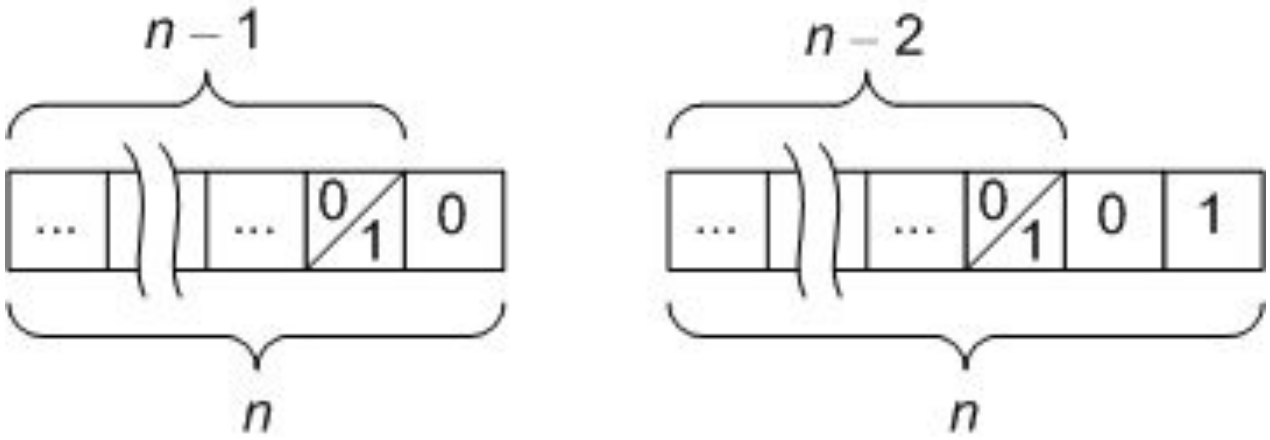
```
int F(int n) {
        if (A[n] != -1) return A[n];
        if (n < 2) return 1;
        else {
                A[n] = F(n - 1) + F(n - 2);
                return A[n];
        }
}
```

Самое простое решение

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

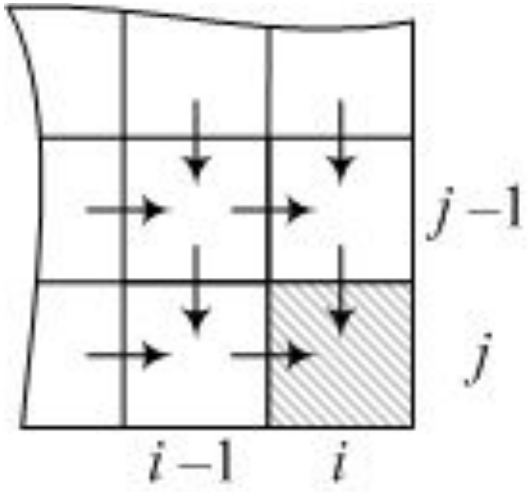
Одномерное динамическое программирование

- **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 Wi и стоимость Pi, i=(1,2..N), требуется собрать набор с максимальной полезностью таким образом, чтобы он имел вес не больше W, где W – вместимость ранца. Wi , Pi , W – целые неотрицательные числа.

Методы

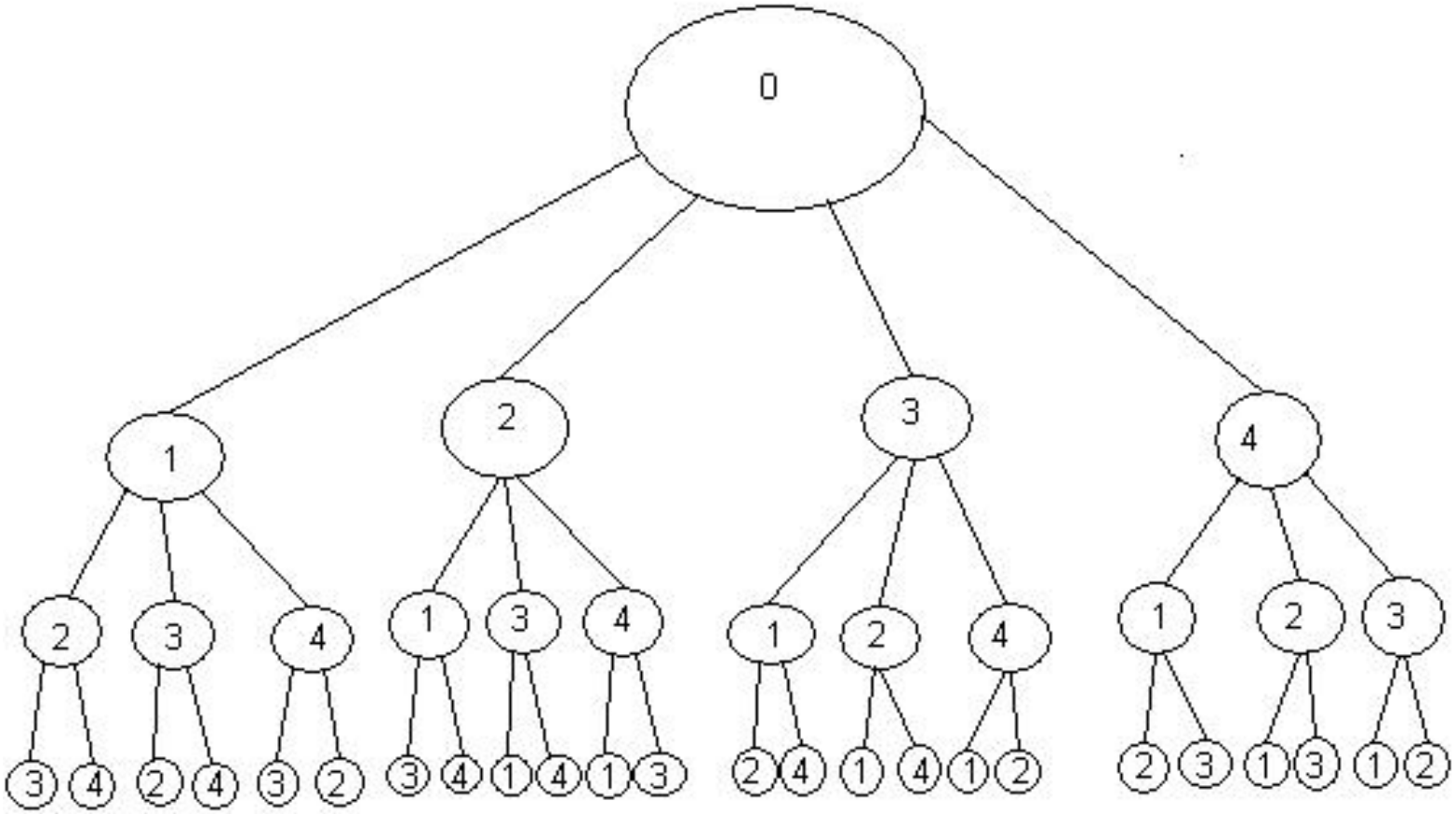
- Полный перебор
- Динамическое программирование
- Метод ветвей и границ
- Жадный алгоритм

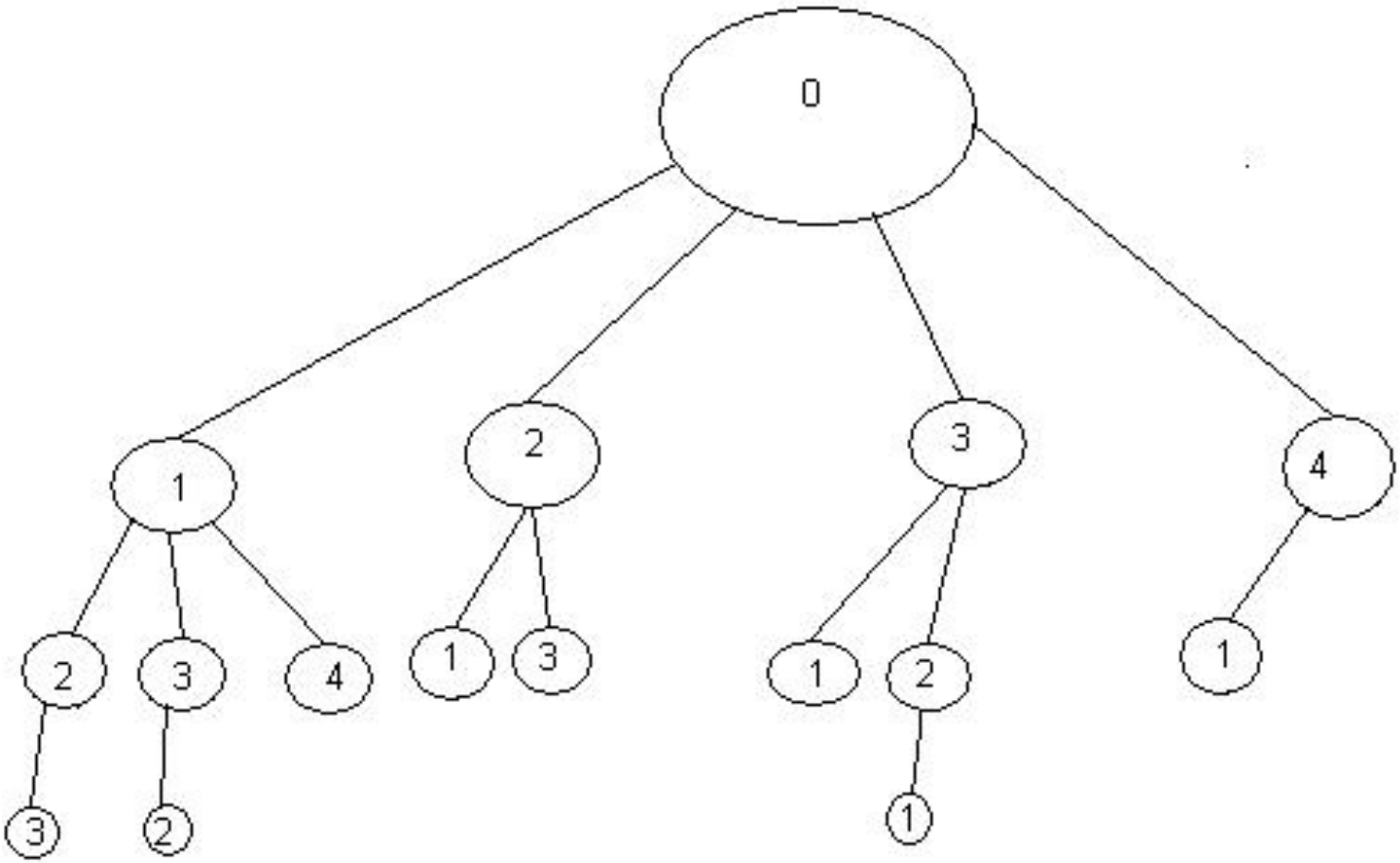
Динамическое программирование

- Value [W, N] – максимальная сумма, которую надо найти.
- Суть метода– на каждом шаге по весу 1<Wi<W находим максимальную загрузку Value[Wi, i], для веса Wi. 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 Wi меньше W проверим стоит ли его брать.

- Если его взять то вес станет W-Wi , тогда Value[W, i] = Value[W – Wi , i-1] + Pi (для Value[W – Wi , i-1] решение уже найдено остается только прибавить Pi).
- 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




0
1
2
3
4
2
3
4
1
3
1
2
1
3
2
1

Жадный алгоритм

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