Динамическое программирование

Непрерывный рюкзак — вариант задачи, в котором возможно брать любую дробную часть от предмета, при этом удельная стоимость сохраняется.
Сортируем предметы по уменьшению удельного веса и берём поочерёдно столько дорогих, сколько можно унести или сколько есть в наличии до тех пор пока не заполнится рюкзак.
Стоит отметить, что жадный алгоритм не всегда выдаёт оптимальное решение. Например, при разложении натурального числа на сумму квадратов натуральных чисел с наименьшим количеством слагаемых:
$32 = 25 + 4 + 1 + 1 + 1$ (жадно)$= 16 + 16$ (оптимально)

Методы поиска глобального минимума

1. Метод бисекции (основанный на следствии из I теоремы Больцано-Коши)
   Делим отрезок на две части и сравниваем центр с 0. Выбираем правую или левую часть и переобозначаем границы. Продолжаем поиск пока не попадем в корень или не приблизимся к нему на заданную точность.

2. Метод Ньютона - это итерационный численный метод нахождения корня заданной функции с помощью касательной к графику функции.

3. Градиент $f(x, y, z)$ – заданная функция. Тогда $\nabla f = (\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz})$ – градиент функции
   Градиентный способ -– движение в сторону/против направления вектора роста для поиска максимума/минимума функции.

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta) + f(x)}{\Delta x}$$

$$f(x_0, y_0, z_0)$$

$$\nabla f = (x_0, y_0, z_0) \simeq (\frac{f(x_0+\Delta x, y_0, z_0) - f(x_0, y_0, z_0)}{\Delta x}, \frac{f(x_0, y_0+\Delta y, z_0) - f(x_0, y_0, z_0)}{\Delta y}, \frac{f(x_0, y_0, z_0+\Delta z) - f(x_0, y_0, z_0)}{\Delta z})$$

Динамическое программирование

Последовательность Фибоначчи.
Количество вычислений каждого члена выражается через последовательность Фибоначчи. Последовательность растёт со скоростью $O(\varphi^n)$, где φ - золотое сечение. Создадим массив из $n+1$ элемента, где будут лежать промежуточные значения функции Фибоначчи: $a[n+1] = \{-1, -1, -1, ..., -1\}$
$f(n) : if(a[n] == -1)$
$a[n] = f(n-1) + f(n-2)$;
return $a[n]$;
Получаем линейную скорость работы алгоритма. Это называется обратным

ходом динамического программирования. Но у данного алгоритма есть существенный недостаток - большой проигрыш в памяти. Однако в памяти можно хранить только два последних значения.

Вычислим биномиальный коэффициент $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$
Заметим, что если повернуть трекгольник Паскаля на $90°$, то получим матрицу, значением каждой клетки которой будет являться сумма значений клетки сверху и слева:

1	1	1	1	1
1	2	3	4	
1	3	6		
1	4			
1				