Вычисление чисел Фибоначчи в стратегии сверху вниз:

```
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
zipWith f (x:xs) (y:ys) = f x y : zipWith f xs ys
zipWith _ _ _           = []

fibs = 0:1:zipWith (+) fibs (tail fibs)

fib n = fibs !! n

(!!) :: Int -> [a] -> [a]
(x:xs) !! 0 = x
(x:xs) !! n = xs !! (n-1)

fib 3
2

fibs !! 3
(0:1:zipWith (+) fibs (tail fibs)) !! 3
(1:zipWith (+) fibs (tail fibs)) !! 2
(zipWith (+) fibs (tail fibs)) !! 1
(0 + 1 : zipWith (+)
    (1:zipWith (+) fibs (tail fibs))
    (zipWith (+) fibs (tail fibs))) !! 1
(zipWith (+)
    (1:zipWith (+) fibs (tail fibs))
    (zipWith (+) fibs (tail fibs))) !! 0
(zipWith (+)
    (1:zipWith (+) fibs (tail fibs))
    (zipWith (+)
        (0:1:zipWith (+) fibs (tail fibs))
        (1:zipWith (+) fibs (tail fibs)))) !! 0
(zipWith (+)
    (1:zipWith (+) fibs (tail fibs))
    (0 + 1 : zipWith (+)
        (1:zipWith (+) fibs (tail fibs))
        (zipWith (+) fibs (tail fibs)))) !! 0
(1 + 1 : zipWith (+)
    (zipWith (+) fibs (tail fibs))
    (zipWith (+)
        (1:zipWith (+) fibs (tail fibs))
        (zipWith (+) fibs (tail fibs)))) !! 0
2
```

В Haskell применяется измененная версия этой стратегии. Создается ссылка на fibs и fibs будет вычисляться только один раз, на каждом шаге добавляя новые элементы.

Merge sort

```
sequences (a:b:xs)
  | a > b      = descending b [a]  xs
  | otherwise = ascending  b (a:) xs
sequences xs  = [xs]

descending a as bs@(b:bs')
  | a > b          = descending b (a:as) bs'
descending a as bs = (a:as): sequences bs

ascending a as bs@(b:bs')
  | a <= b        = ascending b (\ys -> as (a:ys)) bs'
ascending a as bs = as [a]: sequences bs

mergeAll [x] = x
mergeAll xs  = mergeAll (mergePairs xs)

mergePairs (a:b:xs) = merge a b: mergePairs xs
mergePairs xs       = xs

merge as@(a:as') bs@(b:bs')
  | a > b      = b:merge as  bs'
  | otherwise = a:merge as' bs
merge [] bs   = bs
merge as []   = as

mergeSort :: Ord a => [a] -> [a]
mergeSort = mergeAll . sequences
```