Exploration

Feb 2021

Given the triangle of consecutive odd numbers:

```
               1
            3     5
         7     9    11
      13    15    17    19
   21    23    25    27    29
31    33    35    37    39    41
...
```

Calculate the row sums of this triangle from the row index (starting at index 1) e.g.:

```
row_sum_odd_numbers(1) # 1
row_sum_odd_numbers(2) # 3 + 5 = 8
```

Can we build this triangle?

- n = 1, [1]
- n = 2, [3, 5] [n+1, n+3] n is even
- n = 3, [7, 9, 11] [n+4, n+6, n+8] n is odd
- n = 4, [13, 15, 17, 19] [n+9, n+11, n+13, n+15] n is even
- n = 5, [21, 23, 25, 27, 29] [n+16, n+18, n+20, n+22, n+24]
- n = 6, [31, 33, 35, 37, 39, 41]
- n = 7, [43, 45, 47, 49, 51, 53 55]

Notice the upper limit is given by: $n^2 + n - 1$ which can be cheked on the follwoing (this is not a demonstration, but one can devise a recurrence)...

- $$n = 1 \Rightarrow n^2 + n - 1 = 1^2 + 0 = 1$$
- $$n = 2 \Rightarrow n^2 + n - 1 = 2^2 + 1 = 5$$
- $$n = 3 \Rightarrow n^2 + n - 1 = 3^2 + 2 = 11$$
- $$n = 4 \Rightarrow n^2 + n - 1 = 4^2 + 3 = 19$$
- $$n = 5 \Rightarrow n^2 + n - 1 = 5^2 + 4 = 29$$
- $$n = 6 \Rightarrow n^2 + n - 1 = 6^2 + 5 = 41$$
- $$n = 7 \Rightarrow n^2 + n - 1 = 7^2 + 6 = 55$$

...

Also notice that the lower bound is then given by formulae: $(n-1)^2 + n - 2 + 2 \equiv (n-1)^2 + n$

- $$n = 1 \Rightarrow (n-1)^2 + n = 0 + 1 = 1$$
- $$n = 2 \Rightarrow (n-1)^2 + n = 1 + 2 = 3$$
- $$n = 3 \Rightarrow (n-1)^2 + n = 4 + 3 = 7$$
- $$n = 4 \Rightarrow (n-1)^2 + n = 9 + 4 = 13$$
- $$n = 5 \Rightarrow (n-1)^2 + n = 16 + 5 = 21$$

...

```
begin
  using Test
  using BenchmarkTools

  using PlutoUI
end
```

gen_odd_seq

Calc. sequence of odd integer from (n-1)^2 + n..n^2 + (n-1)

```
"""
Calc. sequence of odd integer from (n-1)^2 + n..n^2 + (n-1)
"""
function gen_odd_seq(n::T) where T <: Integer
    T[ix for ix in (n-1)^2+n:2:n^2+(n-1)]
end

# type stable - check with @code_warntype
```

```
@code_warntype gen_odd_seq(2)  # check output in console...
```

```
MethodInstance for Main.var"workspace#3".gen_odd_seq(::Int64)
  from gen_odd_seq(n::T) where T<:Integer in Main.var"workspace#3" at /home/pascal/Projects/ML_DL/Notebooks/jul
ia-notebooks/Julia Kata/02_sum_odd_numbers_exploration.jl#==#a2253dfa-6805-11eb-14ca-a58592832cdf:4
Static Parameters
  T = Int64
Arguments
  #self#::Core.Const(Main.var"workspace#3".gen_odd_seq)
  n::Int64
Locals
  @_3::Union{Nothing, Tuple{Int64, Int64}}
  @_4::Int64
  ix::Int64
Body::Vector{Int64}
1 ─ %1  = (n - 1)::Int64
    %2  = Core.apply_type(Base.Val, 2)::Core.Const(Val{2})
    %3  = (%2)()::Core.Const(Val{2}())
    %4  = Base.literal_pow(Main.var"workspace#3".:^, %1, %3)::Int64
    %5  = (%4 + n)::Int64
    %6  = Core.apply_type(Base.Val, 2)::Core.Const(Val{2})
    %7  = (%6)()::Core.Const(Val{2}())
    %8  = Base.literal_pow(Main.var"workspace#3".:^, n, %7)::Int64
    %9  = (n - 1)::Int64
    %10 = (%8 + %9)::Int64
    %11 = (%5:2:%10)::Core.PartialStruct(StepRange{Int64, Int64}, Any[Int64, Core.Const(2), Int64])
    %12 = Base.IteratorSize(%11)::Core.Const(Base.HasShape{1}())
    %13 = (%12 isa Base.SizeUnknown)::Core.Const(false)
    %14 = Base._array_for($(Expr(:static_parameter, 1)), %11, %12)::Vector{Int64}
    %15 = Base.LinearIndices(%14)::LinearIndices{1, Tuple{Base.OneTo{Int64}}}
          (@_4 = Base.first(%15))
          (@_3 = Base.iterate(%11))
    %18 = (@_3 === nothing)::Bool
    %19 = Base.not_int(%18)::Bool
```

Test Passed

```
begin
    a = gen_odd_seq(1)
    @test a == [1]
end
```

```
begin
    a₂ = gen_odd_seq(3);
    println(a₂)             # output console: [7, 9, 11]
end
```

```
[7, 9, 11]
```

Test Passed

```
begin
    a₁₀ = gen_odd_seq(10)
    @test a₁₀ == Int[91, 93, 95, 97, 99, 101, 103, 105, 107, 109]
end
```

```
20
```

```
with_terminal() do
    println(length(gen_odd_seq(20)))
end
```

```
[Vector{Int64}]
```

```
Base.return_types(gen_odd_seq, (Int64,))
```

```
row_sum_odd_numbers (generic function with 1 method)
```

```
function row_sum_odd_numbers(n::T) where T <: Integer
  gen_odd_seq(n) |> ary -> sum(ary)
end
```

```
Test Passed
```

```
begin
    @test row_sum_odd_numbers(1) == 1
    @test row_sum_odd_numbers(2) == 8
    @test row_sum_odd_numbers(42) == 74088
end
```

```
MethodInstance for Main.var"workspace#3".row_sum_odd_numbers(::Int64)
  from row_sum_odd_numbers(n::T) where T<:Integer in Main.var"workspace#3" at /home/pascal/Projects/ML_DL/Notebooks/ju
Static Parameters
  T = Int64
Arguments
  #self#::Core.Const(Main.var"workspace#3".row_sum_odd_numbers)
  n::Int64
Locals
  #1::Main.var"workspace#3".var"#1#2"
Body::Int64
1 ─ %1 = Main.var"workspace#3".gen_odd_seq(n)::Vector{Int64}
│        (#1 = %new(Main.var"workspace#3".:(var"#1#2")))
│   %3 = #1::Core.Const(Main.var"workspace#3".var"#1#2"())
│   %4 = (%1 |> %3)::Int64
└──      return %4
```

```
with_terminal() do
    @code_warntype row_sum_odd_numbers(5)
end
```

Performance

```
  43.204 ms (17952 allocations: 382.38 MiB)
```

```
with_terminal() do
    @btime for i in 1:10_000
        row_sum_odd_numbers(i)
    end
end

# Note: a lot of allocations!
```

```
row_sum_odd_numbers_2 (generic function with 1 method)
```

```
# A new version more efficient ?
function row_sum_odd_numbers_2(n::T) where T <: Integer
  s = zero(eltype(n))   # pre-alloc
  for ix in (n-1)^2+n:2:n^2+(n-1)
      s += ix
  end
  s
end
```

```
  13.046 ms (0 allocations: 0 bytes)
```

```
with_terminal() do
    @btime for i in 1:10_000
        row_sum_odd_numbers_2(i)
    end
end

# Note: faster, still more memory allocations!
```

```
alloc_gen_odd_seq_sum (generic function with 1 method)
```

```
function alloc_gen_odd_seq_sum(m::T) where T <: Integer
    ary = Vector{T}(undef, m)                       ## Allocate

    gen_fn = function (n::T) where T <: Integer     ## Populate
        for (ix, v) ∈ enumerate((n-1)^2+n:2:n^2+(n-1))
            ary[ix] = v
        end
    end

    sum_fn = function (n::T) where T <: Integer     ## Sum...
        gen_fn(n) |> ary -> sum(ary)
    end

    sum_fn
end
```

```
  4.829 ms (17952 allocations: 382.38 MiB)
```

```
with_terminal() do
    @btime for ix in 1:10_000
        alloc_gen_odd_seq_sum(ix)
    end
end
```