Work and span of buySell algorithm

Assuming input of length N.

Our algorithm has 3 steps:

- `buildMatrix`: builds a matrix of all possible buy and sell dates
  - work: N^2
  - span: 1, there are no dependencies between elements of the matrix
- `getSell`: computes the best sell date for each buy date

  A `fold` has work N and span $\log N$. We have N folds folding over N elements:
  - work: N^2
  - span: $\log N$, it is possible to run all N folds in parallel.
- `getBuy`: computes the best buy date
  - work: N
  - span: $\log N$

In total we have

- work: $2N^2 + N$ and
- span: $2 \log N + 1$

Speedup of parallelism!

Running the same algorithm sequentially and in parallel (with 2 HECs) we get a speedup of almost 2, which is pretty good.

An example run

```
% ./Stock +RTS -N2
benchmarking sequential
time                 22.66 ms   (22.18 ms .. 23.14 ms)
                     0.998 R²   (0.997 R² .. 0.999 R²)
mean                 23.41 ms   (22.99 ms .. 24.61 ms)
std dev              1.470 ms   (436.5 us .. 2.567 ms)
variance introduced by outliers: 24% (moderately inflated)

benchmarking parallel
time                 12.19 ms   (11.92 ms .. 12.67 ms)
                     0.992 R²   (0.982 R² .. 0.998 R²)
mean                 12.69 ms   (12.38 ms .. 13.49 ms)
std dev              1.222 ms   (610.9 us .. 2.055 ms)
variance introduced by outliers: 50% (severely inflated)
```

The code is compiled with `-Odph -rtsopts -threaded -fno-liberate-case -funfolding-use-threshold1000 -funfolding-keeness-factor1000 -fllvm -optlo-O3` as is recommended in the Repa documentation.