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 parllel (with 2 HECs) we get a speedup of approximately 5.5, which is pretty amazing.

An example run

```
% ./Stock +RTS -N2
benchmarking sequential
time                 1.287 s    (1.164 s .. 1.403 s)
                     0.998 R²   (0.998 R² .. 1.000 R²)
mean                 1.295 s    (1.269 s .. 1.308 s)
std dev              22.22 ms   (0.0 s .. 23.82 ms)
variance introduced by outliers: 19% (moderately inflated)

benchmarking parallel
time                 232.9 ms   (216.9 ms .. 251.5 ms)
                     0.996 R²   (0.988 R² .. 1.000 R²)
mean                 224.7 ms   (214.2 ms .. 230.7 ms)
std dev              9.634 ms   (3.687 ms .. 13.16 ms)
variance introduced by outliers: 14% (moderately inflated)
```