Teacher instructions:

Students should note that there is a loop carried dependency throughout the problem, and that there is no way to parallelize the algorithm without changing it.

In changing the algorithm, discuss with the students the need for thorough testing--if the algorithm is different the solution is potentially different. Especially given that this will effectively "shorten" the most successful walk, which potentially could be equivalent to training too fast.

Note that this is a random algorithm, and that multiple tests must be done to ensure consistency.

Discuss with students in which cases it might be more prudent to use parallelism to perform a large ensemble of complete runs, as opposed to speeding up a single run, given the random nature of the results.

Thread safe random number generation.

When using random numbers in shared memory threaded programs, be sure to use a thread safe implementation of your random number generator.

In C, you will want to use drand48_r (reentrant) instead of rand, and the accompanying srand48_r instead of srand.

https://man7.org/linux/man-pages/man3/drand48_r.3.html

Depending on your code structure, this may require storage of multiple random number streams for each thread. If this is occuring outside of a single #pragma omp parallel block, that would require the use of some passed or global memory structure.

Note the use of global

```
struct drand48_data randThread[MAX_THREADS];
```

in the ebsa.c code, initialized in main() and used, along with a call to omp_thread_num() in each random number call.

Custom memory structures

For anything more than simple typed variable, you will need to provide a method of initializing shared data structures across threads.

When the annealing problem splits into threads to determine step size, separate copies of the annealer's data structure are needed for each thread, initialized from the current

state. Note that a copy routine is written and provided for this purpose, and that memory is allocated and stored as a global variable for multiple copies of the annealer data structure.

SAstruct localModel[MAX_THREADS];