IBM benchmarking framework was overkill, didn’t expect you to use framework

OK if you did

Problem: use framework to time sort, reuse array, but array sorted after first iteration

benchmarking time to sort a sorted array
Setup vs load

- For benchmarking, need to do two kinds of calculations
  - setup
  - computation to be timed
- May need to do setup each time, not time it
- Use new Random(seed) to get identical random data
Form data

- Please enter performance data into form
- Earlier version were confused about micro/milli seconds, please use microseconds
  - milliseconds is too coarse for $O(n \log n)$ sorts
Multithreaded sorts

- On smaller arrays, cost to spawn threads made it not worthwhile
- Using a thread pool executor worked
- Did recursive/multithreaded merge sort down to level required to saturate machine, then invoked quicksort
Using Futures with an Executor

```java
Future<?> future = executor.submit(new Runnable() {
    @Override
    public void run() {
        sort(start, middle, parallelLevels - 1); } });

sort(middle, end, parallelLevels - 1);
future.get();
```
Final project

• Everyone does Sudoko logic, including recursive solution

• Pick either:
  • KenKen logic
  • demo project
Sudoku

• You need to figure out how to implement the internal logic
• lots of choices
Sudoku logic

• A value can’t be placed in a square if that value already occurs in the same square, row or column

• Your computation of legal moves needs to take, at a minimum, this into account

• Next heuristic: if, in a row, there is only one place a 5 can be placed, that place needs to be a 5

• same for columns, squares, other values
KenKen

- In a number of ways, much harder than Sudoku
- Regions are known as cages
- Cages that span more than one row and more than one column can contain duplicates
Project extensions

- People not doing demo’s will be required to choose a project extension, such as:
  - Add undo
  - Add puzzle difficulty evaluation
For Wednesday

• Come up with a written plan for how to implement logic for Sudoku

• And either a plan for how to implement KenKen, or a plan for a demo project

• Will distribute Sudoku project (no code) after class
  • update plan.txt, submit it

• we will spend Wednesday in groups, discussing them
• Text file format for stored kenken problems
• How are cages represented in memory
• How to we compute cage constraints?
• How do we take into account cages that contain duplicates?
• Constraints on more than just legal values that can appear in a cage