Group 10

Nathan Beneke and Courtney Cook

Score

- 4th in class, 6th overall
- Sum of places: 11
- Sum of median: 2644.0
- Run 1
  - 1918
  - 1976
  - 1972
- Run 2
  - 673
  - 672
  - 672
- We were correct

The Algorithm

1. Go through String[] creating a KVPair<Long, Long> for every number
    a. The Key in this KVPair is the number, the value is the product of its first two prime numbers (or only the first prime factor if there is only one)
    b. We stored the first 200 primes in an int[], and converted them to longs when the Comparator is initialized, and first loop through those 200 to see if the first or second prime factors are in that array. This is true for the vast majority of numbers.
        i. Note that the int array is legal because the it uses the same amount of memory as 100 longs and is turned into longs once per sort.
    c. If the end of of the prime array is reached before finding both primes we enter a for loop from the last prime in the array + 2 to the bound, incrementing by 2.
        i. The bound starts at sqrt(n) and is set to min(sqrt(n), n / (firstPrime ^ m)) where m is the highest integer such that n / (firstPrime^m) is an integer
    d. The lowest factor is guaranteed to be prime, the next prime factor is the first factor not divisible by the first prime
    e. If the bound is reached before finding the second prime factor, and still equals sqrt(n), we check the potential prime greater than the bound using Elena’s method

Efficiency

- Sorting
  - Dual-Pivot Quicksort
    - Worst case: $\Theta(n^2)$
    - Best and expected: $\Theta(n \log_2(n))$
- Finding Primes
  - For an individual number k,
    - Worst case: Θ(sqrt(k)) when k's first 2 prime factors are very large, because we enter the for loop incrementing by 2 up to potentially sqrt(k)
    - Best and average case: Θ(k / ln(k)) as most of the time, k's first two prime factors are in the first 200 primes and k / ln(k) is approximately how many primes are less than or equal to k
  - Then for a list of n length, using k as an average of every number in the data set
    - Worst case: Θ(n * sqrt(k)) which is just Θ(n) when most numbers have large prime factors
    - Best and average case: Θ(n * (k / ln(k))) which is still Θ(n)
    - However, k is a very large number
- Total
  - Best / average: $\Theta(n \log_2(n) + n * (k / \ln(k))) = \Theta(n \log_2(n))$
  - Worst: $\Theta(n^2)$ for the same reasons as above
  - Note that k is still VERY large and so sorting seems slower

Interesting Features and Things Tried

- Before implementing the KVPair[] we tried to compute prime products in parallel with each other, so that if one number had a much larger product of two primes then we could exit the loop.
- Also before the KVPair[], we would compute the first 400,000 prime numbers once per sort in order to improve efficiency in finding prime factors.
  - Note that the larger the data set, the more primes it is efficient to compute to reduce repeated operations.
  - With the KVPair[] it was still worth it to have a smaller list of pre-computed primes but was inefficient to compute further primes.
- Should have chosen random pivots
- Should have stored first 400 prime numbers as shorts, rather than first 200 as ints.