CS 124 Programming Assignment 2

30943147

March 20, 2017

Caching

When multiplying two matrices $A * B = C$ using the naive matrix multiplication method, the elements of C are given as follows

$$C[i][j] = \sum_k A[i][k] * B[k][j]$$

This is naturally implemented with 3 nested for loops iterating over i, j, and k. There are $3! = 6$ possible permutations of their order. I tested all of them experimentally to determine which was best. Runtime measurements were taken by running the naive algorithm on $n \times n$ matrices with randomly generated entries for $n = 1200$.

ordering	runtime
i, j, k	5.81s
i, k, j	1.73s
j, i, k	5.54s
j, k, i	25.30s
k, i, j	2.05s
k, j, i	24.67s

The ordering i, k, j produces the best results. The caching behaviour of that ordering can be further optimized by saving the intermediate value of each $A[i][k]$ before we iterate over j. This allows the innermost loop, with iterates over j, to only access a single row of matrix B, in order, via calls to access $B[k][j]$. Since the matrices are stored in row-major order, this is very cache friendly. Saving the intermediate value of $A[i][k]$ prevents the cache from thrashing as it must repeatedly load up $A[i]$ to recalculate $A[i][k]$, which is wasted work since $A[i][k]$ isn't changing during the innermost loop over j.

Experimental Results for n_0

Testing Correctness

I used the unused first parameter as input for n_0 to determine what the experimental value for the optimal $n - 0$ is. The program will default to whatever I

determine that value to be if 0 is input for the first parameter (as the problem set description said it would be for grading). I wrote a bash script called check to help me test correctness, that would run the program first on an inputted $n_0 < n$, then on $n_0 > n$, and use the shell diff command to check that their respective outputs are identical. When $n_0 < n$, Strassen's algorithm will be used at least once. When $n_0 > n$, only the traditional matrix multiplication algorithm will be used. Therefore, this process checks my implementation of Strassen's algorithm against the traditional matrix multiplication algorithm.

Runtime for Different n_0

I wrote another bash script called testn0 to experiment efficiently with timing the runtime of different values of n and n_0. I was curious if the value of the optimal crossover point might vary with n as well as with n_0. It seemed natural to test powers of 2 for n_0, and I experimented with several different values of n. I averaged over 5 trials to reduce statistical variation. The first entry in the table, with $n_0 > n$, represents using the traditional multiplication algorithm.

n	n_0	runtime
1000	> 1000	0.82s
1000	64	1.38s
1000	128	0.83s
1000	256	0.82s
1000	512	0.67s

$n_0 \approx 512$ looks promising. We experiment with a few more values of n.

n	n_0	runtime
2000	> 2000	4.43s
2000	64	5.36s
2000	128	3.47s
2000	256	2.76s
2000	512	2.77s
2000	1024	3.83s

n	n_0	runtime
1500	> 1500	2.11s
1500	64	2.72 s
1500	128	2.10ss
1500	256	1.68s
1500	512	1.45s
1500	1024	1.78s