CME 323: Distributed Algorithms and Optimization

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#1 – Due at the beginning of class Thursday April 19

1. A computation takes 250 seconds to run on a single core, i.e. $T_1 = 250$. Is it possible to design a multithreaded version of the computation that takes time 40 seconds on 5 processors with greedy scheduling? How about 60 seconds on 5 processors? Justify your answer.

2. The Karatsuba algorithm multiplies two integers x and y. Assuming each has n bits where n is a power of 2, it does this by splitting the bits of each integer into two halves, each of size $n/2$. For any integer x we will refer to the low order bits as x_l and the high order as x_h. The algorithm computes the result as follows:[1]

   function **km**(x, y, n):
   if $n = 1$ **then**
   return $x \times y$
   else
   $a \leftarrow$ **km**(x_l, y_l)
   $b \leftarrow$ **km**(x_h, y_h)
   $c \leftarrow$ **km**$(x_l + x_h, y_l + y_h)$
   $d \leftarrow c - a - b$
   return $(b2^n + d2^{n/2} + a)$
   end if

   Note that multiplying by 2^k can be done just by shifting the bits over k positions.

   (a.) Assuming addition, subtraction, and shifting take $O(n)$ work and $O(n)$ depth what is the work and depth of **km**?

   (b.) Assuming addition, subtraction, and shifting take $O(n)$ work and $O(\log n)$ depth what is the work and depth of **km**?

3. Suppose a square matrix is divided into blocks:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

   where all the blocks are the same size. The *Schur complement* of block D of M is $S = A - BD^{-1}C$. The inverse of the matrix M can then be expressed as:

$$M^{-1} = \begin{bmatrix} S^{-1} & S^{-1}BD^{-1} \\ -D^{-1}CS^{-1} & D^{-1} + D^{-1}CS^{-1}BD^{-1} \end{bmatrix}$$

   This basically defines a recursive algorithm for inverting a matrix which makes two recursive calls (to calculate D^{-1} and S^{-1}), several calls to matrix multiply, and one each to elementwise add and subtrace two matrices. Assuming that matrix multiply has work $O(n^3)$ and depth $O(\log n)$ what is the work and depth of this inversion algorithm?

[1] If you have seen this before, you might have thought of it as a sequential algorithm, but actually it is a parallel algorithm, since in particular, the three recursive calls to **km** can be made in parallel.

4. Describe a divide-and-conquer algorithm for merging two sorted arrays of lengths n into a sorted array of length $2n$. It needs to run in $O(n)$ work and $O(\log^2 n)$ depth. You can write the pseudocode for your algorithm so that it looks like your favorite sequential language (C, Java, Matlab, ...), but with an indication of which loops or function calls happen in parallel. For example, use `parallel for` for a parallel for loop, and something like:

```
parallel {
    foo(x,y)
    bar(x,y)
}
```

to indicate that `foo` and `bar` are called in parallel. You should prove correctness at the level expected in an algorithms class (e.g. CME305 or CS161).

5. Given the price of a stock at each day for n days, we want to determine the biggest profit we can make by buying one day and selling on a later day. For example, the following stock prices have a best profit of 5:

   $[12, 11, 10, 8, 5, 8, 9, 6, 7, 7, 10, 7, 4, 2]$

   since we can buy at 5 on day 5 and sell at 10 on day 11. This has a simple linear time serial solution. Give an algorithm to solve this problem that runs in $O(n)$ work and $O(\log n)$ depth. Give pseudocode as in the previous problem.

6. In this problem, we'll look at how fast the maximum of a set of n elements can be computed when allowing for concurrent writes. In particular we allow the arbitrary write rule for "combining" (i.e. if there are a set of parallel writes to a location, one of them wins). Show that this can be done in $O(\log \log n)$ depth and $O(n)$ work.

   (a.) Describe an algorithm for maximum that takes $O(n^2)$ work and $O(1)$ depth (using concurrent writes).

   (b.) Use this to develop an algorithm with $O(n)$ work and $O(\log \log n)$ depth. Hint: use divide and conquer, but with a branching factor greater than 2.

For the next few problems, we expect that you can learn SQL on your own and answer the below questions. Some good guides for learning SQL:

(a) `https://www.w3schools.com/sql/sql_intro.asp`

(b) `https://www.youtube.com/watch?v=7Vtl2WggqOg`

7. In this set of problems, we review how to *select* data from relational databases.

   (a.) Write a SQL statement to find the total purchase amount of all orders. Sample table: `orders`.

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002

   (b.) Write a SQL statement which selects the highest grade for each of the cities of the customers. Sample table: `customer`.

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001

   (c.) Write a SQL statement to find the highest purchase amount on a date "2012-08-17" for each salesman with their ID. Sample table: `orders`, used in (a).

8. In this problem, we review how to *merge* two tables together.

   (a.) Write a SQL statement to know which salesman are working for which customer. Use the sample table `customer`, used in previous problem, and also `salesman`.

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12

   (b.) Write a query to display all salesmen and customers located in London.

(c.) Write a SQL statement to make a cartesian product between `salesman` and `customer` i.e. each salesman will appear for all customer and vice versa for those salesmen who belongs to a city and the customers who must have a grade.

(d.) Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to find that either any of the existing customers have placed no order or placed one or more orders. Use `customer` and `orders` tables.

9. In this problem, we consider *aggregation* of data.

   (a.) Write a SQL statement to find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount. Sample table: `orders`, used in problem 6 part (a).

   (b.) Write a SQL query to display the average price of each company's products, along with their code. Sample table: `item_mast`.

PROD_ID	PROD_NAME	PROD_PRICE	PROD_COMPANY
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12

10. **Joins with multiple keys** The point of this question is to explore how SQL handles cases where a join is performed on tables containing duplicate rows. Consider the following table `item_mast`.

PROD_ID	PRODUCT	PROD_PRICE	PROD_COMPANY
101	Mother Board	3200	1
101	Mother Board	2900	999
103	ZIP drive	250	14
106	DVD drive	900	12

and a corresponding table of customer purchases, `purchases`.

PROD_ID	CUSTOMER	PRODUCT	city
101	James Hoog	Mother Board	New York
101	James Hoog	ZIP drive	Los Angeles
103	Mc Lyon	ZIP drive	Pittsburgh

Notice that in `item_mast`, the same product can appear multiple times (listed under different manufacturers). Also, in database `purchases` the same customer can appear

multiple times. If we `join` carefully using select columns, we can identify observations uniquely in the resulting output table. However, suppose we join the two tables only on `item_mast: product, product price` and `purchases: customer, product`.

Draw a sample table describing what the output looks like, and explain the result.