Hw5 writup

Problem1~2

Problem 1

1. precedence constraint $\Rightarrow T \geq \max\left(\frac{5}{1}, \frac{5}{1}\right) = 5$

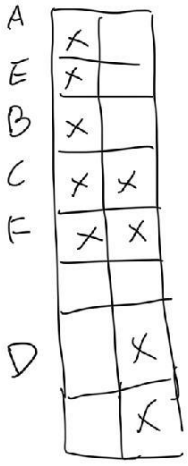
   resource constraint $\Rightarrow T \geq \max\left(\frac{5}{1}, \frac{4}{1}\right) = 5$

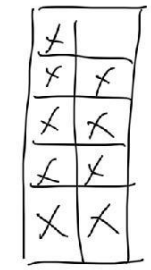
$\Rightarrow$ initial interval $T \geq 5$

2. $\begin{cases} S(C) - S(B) \geq 1 \\ S(B) - S(C) \geq 4 - T \end{cases} \Rightarrow S(B)+1 \leq S(C) \leq S(B)+T-4$

$\begin{cases} S(F) - S(E) \geq 3 \\ S(E) - S(F) \geq 2 - T \end{cases} \Rightarrow S(E)+3 \leq S(F) \leq S(E)+T-2$

$T=5 \Rightarrow S(C) = S(B)+1, \quad S(F) = S(E)+3$





modulo reservation table

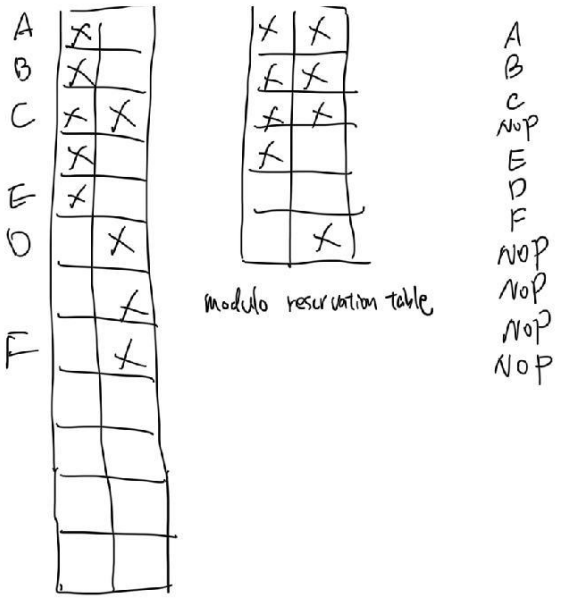
schedule :

A
E
B
C
F
NOP
D
NOP
NOP
NOP

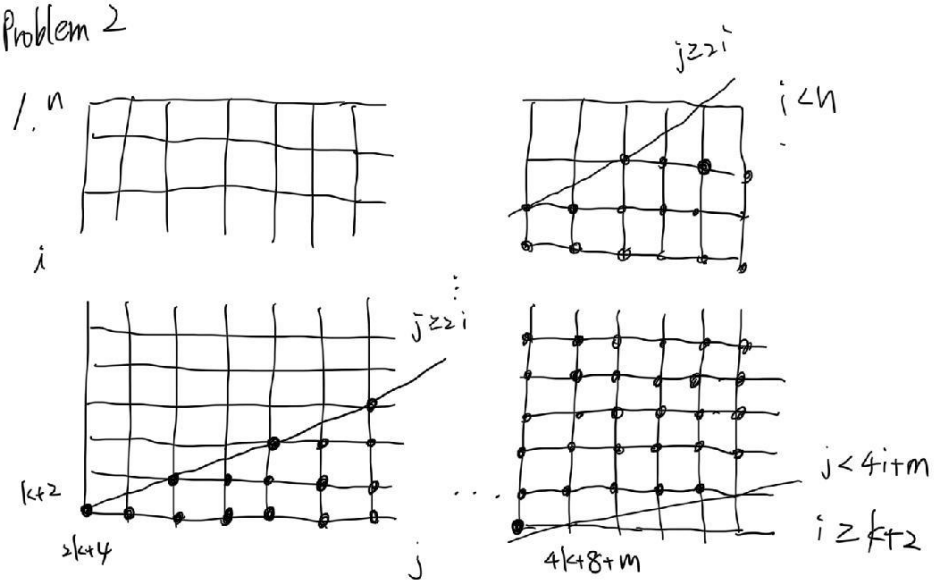
3. No, it can't

$T=6 \Rightarrow S(B)+1 \leq S(C) \leq S(B)+2, \quad S(E)+3 \leq S(F) \leq S(E)+4$

schedule :



Problem 2



2. ① write X[i, 4*j-2], read X[i, 4*j+1] (no dependency because they only access even/odd elements)

② write X[i, 4*j-2], read X[i, 2*j] (no dependence because we need $j'=1$ for $4j-2=2j'$, but we also have $j' \geq 2(k+2) = 4$)

③ write Y[i-1, j-3], read Y[i, j]

3. we have $k+2 \leq i, i' < n$

$2i < j < 4i+m$

$2i' < j' < 4i'+m$

$\begin{bmatrix} i \\ j \end{bmatrix} * \begin{bmatrix} i' \\ j' \end{bmatrix}$ as basic constraints

and add ① $i=i'$, $4j-2=4j'+1$

② $i=i'$, $4j-2=2j'$

③ $i-1=i'$, $j-3=j'$ to solve the three dependency in 2. seperately.

4. The loop is 1-d parallelizable because there's no dependence when either i or j fixed. It's not 2-d parallelizable because read Y[i, j] depend on write Y[i-1, j-3]

Problem3

3.1

3.1.1	Avg(us)	Std(us)	3.1.2	Avg(us)	Std(us)
serial	1260800	21624	serial	8626	1550
naïve_parallel	599116	101524	naïve_parallel	6531	1216
blocked_parallel	61969	3164	blocked_parallel	4239	1489

3.1.1 1000 by 1000 matrix multiplies 1000 by 1000 matrix

Blocked_parallel is significantly faster than other two implementation.

3.1.2 1000 by 10 matrix multiplies 10 by 1000 matrix

Blocked_parallel is slightly faster than other two implementation.

3.2.1

I tried implementing 3 version of parallel code – (1) Implementation of code in lecture slide. (2) Improve (1) to reduce synchronization overhead (3) Implementation of code in the hw5 handout. (See the comments in the code)

(3) performs the best among all three so I choose it as the final implementation. However, it is still slower than the serial code.

3.2.2

3.2.2(a)	Avg(us)	Std(us)	3.2.2(b)	Avg(us)	Std(us)	3.2.2(c)	Avg(us)	Std(us)
serial	1780710	108539	serial	87096	8493	serial	101382	9655
parallel	64076812	3355154	parallel	3283226	227331	parallel	3225109	159190

3.2.2(a) 20000 x 20000
3.2.2(b) 20000 x 1000
3.2.2(c) 1000 x 20000

3.3.1

First, I increase the block size from 16 to 128. Then, I fuse a matrix multiplication with elementwise addition. Finally, I fuse all the elementwise activation, multiplication, and addition together and compute c’ and h’ in the same loop to reduce memory access.

3.3.2

I make the matmul and matmul_add 2d-parallelized and final stage 1d-parallelized.

3.3.3

3.3.3(a)	Avg(us)	Std(us)	3.3.3(b)	Avg(us)	Std(us)
kernel(16)	2570931	340078	kernel(16)	9090332	468320
serial(128)	2947990	122967	serial(128)	10307270	461463
parallel(128)	1048930	88970	parallel(128)	2905316	178747

(.) after kernel/serial/parallel indicate the block size.

3.3.3(a) 500, 2000, 2000

3.3.3(b) 500, 100, 5000

3.3.4 I only have 8 cpus in my laptop so I draw the plot from 1 to 8 processors.