please grade

CSE 548 – Analysis of Algorithms, Spring 2013

Assignment #2b

Duo Xu (#108662210)

partner: Yu-Yao Lin (#109090793)

Problem 6

(a) Multiplication of two n-bit numbers is the same as addition of n $(2n-1)$-bit numbers. As the same idea of merge sort, every time we do addition two by two. So in total there are $\log n$ levels. Because in each level we do addition of $O(n)$-bit numbers two by two, the depth in each level is $O(\log n)$. So totally, this design has depth of $O(\log^2 n)$.

The size of addition circuits of two n-bit numbers is $\Theta(n^{\log 3})$, in all the levels we do $(n + \frac{n}{2} + \frac{n}{4} + ... + 1)$ times of addition, so the size of this design is $\Theta(n^{1+\log 3})$.

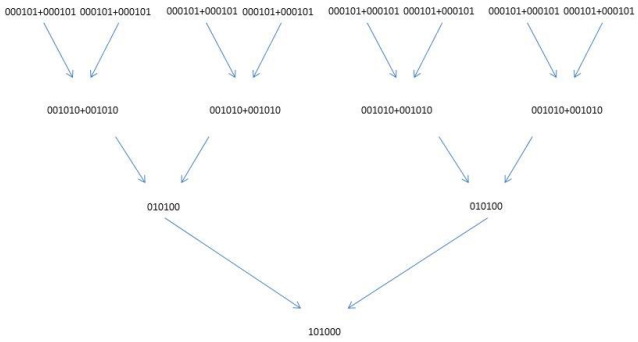

Figure 1: Circuits for multiplying by Yu-Yao Lin

(b) Multiplication of two n-bit numbers is the same as addition of n $(2n-1)$-bit numbers. As the same idea of merge sort, every time we do addition two by two. So in total there are $\log n$ levels. In each level we do addition of $O(n)$-bit numbers two by two. However, the depth in each level is $O(1)$. So totally, this design has depth of $O(\log n)$.

The size of addition circuits of two n-bit numbers is $\Theta(n)$, in all the levels we do $(n+\frac{n}{2}+\frac{n}{4}+...+1)$ times of addition, so the size of this design is $\Theta(n^2)$.

Problem 7

(a) Refer to the problem 5 of problem set 1, the algorithm is that:

We change $a_n a_{n-1} a_{n-2} ... a_1 * b_n b_{n-1} b_{n-2} ... b_1$ to the form of $(a_{\frac{n}{2}} a_{\frac{n}{2}-1} ... a_1 + a_n a_{n-1} a_{n-2} ... a_{\frac{n}{2}+1} * i) * (b_{\frac{n}{2}} b_{\frac{n}{2}-1} ... b_1 + b_n b_{n-1} b_{n-2} ... b_{\frac{n}{2}+1} * i)$, where $i = 1 << \frac{n}{2}$. This is just the same format as $(a + bi)(c + di)$ in the problem 5 of problem set 1.

Thus we need to compute $(a + b)(c + d)$, ac and bd. Each of them are the multiplication of two $\frac{n}{2}$-bit numbers. We can see that the size is reduced half. So we do it recursively and the base case is the multiplication of two 1-bit numbers.

The recurrence relation for depth is $T(n) = T(n/2) + 1$, so the depth of this design is $O(\log n)$

(b) $S(n) = 3S(n/2) + n$

$S(1) = 1$

(c)
$$\begin{aligned} S(n) &= 3S(n/2) + n \\ &= 3(3S(n/4) + n/2) + n \\ &= 3^2 S(n/4) + \tfrac{3}{2}n + n \\ &= 3^3 S(n/8) + \tfrac{3^2}{2^2}n + \tfrac{3}{2}n + n \\ &\quad \vdots \\ &= 3^{\log n} + \tfrac{3}{2}^{\log n - 1} n + \tfrac{3}{2}^{\log n - 2} n + ... + \tfrac{3}{2}^2 n + \tfrac{3}{2}n + n \\ &= \Theta(3^{\log n}) \\ &= \Theta(n^{\log 3}) \\ &= o(n^2) \end{aligned}$$

Problem 8

When the lieutenant begins the firing process, we define a rule that:

1. Assume the information pass speed is 1 person/second.

2. If you are the first time to hear from your neighbor (either left or right), you pass it to your another neighbor at once.

3. If you are the second time to hear from your neighbor (either left or right), you wait 2 second, then pass it to your another neighbor. We assume that the lieutenant wait 2 second after he first passes information to his right soldier.

The reason why I define the above rule is to make sure the fast information passing is 3 times the speed of the slow information passing so that they can meet at the middle of the array.

Then this problem is split into two small arrays.

We define the two soldiers at the middle of the array as the start soldier of each half array and do the information passing as the first round.

As they move and interact at exactly the same speed, we can assume that each sub array is splitting simultaneously.

This is a kind of divide and conquer, and the base case is only 1 soldier. Then all soldiers start firing.

The recurrence relation is $T(n) = T(n/2) + O(n)$

This algorithm is $O(n)$ running time.

	1	2	3	4	5	6	7	8	9
1	R								
2	R	m1							
3	R		m1						
4	R	m2		m1					
5	R				m1				
6	R					m1			
7	R		m2				m1		
8	R							m1	
9	R								R
10	R			m2				m1	R
11	R						m1		R
12	R					m1			R
13	R				meet				R
14	R			m1	R	m1			R
15	R		m1		R		m1		R
16	R	m1		m2	R	m2		m1	R
17	R				R				R
18	R	m1			R			m1	R
19	R		meet		R		meet		R
20	R	m1	R	m1	R	m1	R	m1	R
22	R		R		R		R		R
23	R	meet	R	meet	R	meet	R	meet	R
24	R	R	R	R	R	R	R	R	R
25	fire	fire	fire	fire	fire	fire	fire	fire	fire

Figure 2: Example by Yu-Yao Lin

Problem 9

Assume we have $n-1$ checkers.

1. If we place a checker on a piece where the Manhattan distance is larger than 2 from an existing rectangle on the board, or we place it on an empty board, it means we create a new $1 * 1$ rectangle.

2. If we place a checker which the Manhattan distance between the piece and an existing rectangle (size is $a * b$) is less or equal to 2, we can create a new bigger rectangle based on the neighbor filling rule, the size of which would be $a * (b+1), a * (b+2), (a+1) * b, (a+2) * b$ or $(a+1) * (b+1)$. We name the size of the new rectangle is $c * d$, and we find that $c + d \leq a + b + 2$.

From the above two checker placement strategies, we can find that the first strategy creates a rectangle of perimeter 4 and the second strategy increases the perimeter by at most 4.

If two rectangle can form a bigger rectangle, then the perimeter of the new rectangle is at most the sum of the two original ones.

So we can conclude that every time we place a checker on the board, we can increase the perimeter by at most 4.

If we have $n-1$ chekcers, the final covered rectangle would have perimeter of at most $4(n-1)$.

However, the whole checkerboard's perimeter is $4n$, thus we cannot cover the board by using only $n-1$ checkers.

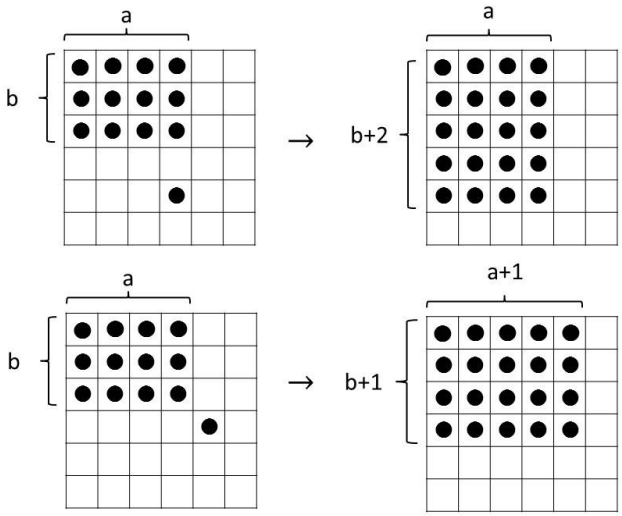

Figure 3: Perimeter increases by at most 4 by Yu-Yao Lin

Reference: `http://chaoxuprime.com/2011/12/fill-a-checkerboard`