HW1

Sch0.1

使用B数组中的某一列（或者对角线）代替M数组即可。

Ex5.6

① 根据教材，全局读写时间为d，同步时间为（和B叉树的B区分一下）$B(p)$

算法中的$\lceil \log_B p(B-1)+1 \rceil$是求p个节点的B叉树的高，记之为$h$。

算法每步时间：

（1）$O(n/p+d)$

（2）$B(p)$

（3）$(h-1)*$：

（3.1）$O(B+2d)$

（3.2）$B(p)$

最后结果忽略一些杂项，结果大概是$\propto B(p)log_B p + dlog_B p$

② 同步，避免脏读

Ex5.7

① 算法中的B改成d

以下两种写法都算对：

a. 根据书上BSP模型的的参数，一个超级步的时间为L，假设每次同步前各处理器均完成了操作，那么就相当于每个处理器均在L内完成了操作（同步时间也算在内），那么一共h（树高）个超级步，时间复杂度为hL。

b. 像Ex5.6那样分析。带宽因子为g，发送d条信息时间为gd。算法每步时间：

(1) $O(n/p+gd)$

(2) t_B（设一个同步时间）

(3) $(h-1)*$:

  (3.1) $g(d+1)+O(d)$

  (3.2) t_B

最后忽略一些杂项，结果大概是$\propto dlog_dp+t_Blog_dp$

② 首先，根据教材，一个超级步内处理器最多接收及发送h（不是树高）条消息，因此，算法中要接收d个孩子的消息并且发送一条消息，要求$d\leq h-1$，这是d的上界；

然后，由①可知，在一个超级步内，耗时函数不是单调的，所以在满足d的上界时使耗时函数取极小值的d就是最佳的d。

HW2

Ex6.3

算法解释：$root$中存放根，每个处理器中f_i存放的是父节点的下标，LC、RC分别存放左右子树的根节点的下标。

过程略，符合要求即可。需要给出最后的$root$，各处理器的f_i以及LC、RC

2

参考

思路：串行程序中，中序遍历就能获得有序数组；并行程序中，需要对中序遍历路径上“有用”的边赋值1，然后计算一个前缀和就能得到节点的位序。

算法（没有严格的写end）：

```
// 二叉树节点用原数组中下标表示，Pi,j表示处理边(i, j)的处理器，我们使用n*n个处理器(某些处理器实际上没用到)
// 共享变量：succ[1...n][1...n]，保存边(u,v)的后继边
// w[1...n][1...n]保存迭代中的权值
// suffix_sum[1...n][1...n]保存迭代中的后缀和
// 输入：序列A,算法6.3得到的root,f,LC,RC,A的大小n
// 输出：A排序后的有序数组B
// step 1: 每个处理器求succ
for all Pi,j par-do
    if f[j] == i        // case 1: j是i的孩子节点
        if LC[j] != n + 1  succ[i][j] = (j, LC[j])
        else if RC[j] != n + 1  succ[i][j] = (j, RC[j])
        else succ[i][j] = (j, i)
    else if f[i] == j   // case 2: j是i的父节点
        if i == LC[j]
            if RC[j] != n + 1  succ[i][j] = (j, RC[j])
            else if j != root  succ[i][j] = (j, f[j])
```

```
            else  succ = NIL
        else
            if j != root  succ[i][j] = (j, f[j])
            else succ[i][j] = NIL
    else succ[i][j] = NIL
endfor
// step 2: 对中序遍历路径上“有用”的边赋值1，并初始化后缀和
for all Pi,j par-do
    if f[j] == i && RC[i] == j          // case 3.1 (父节点, 右节点)
       || LC[j] == i && RC[i] == n + 1 // case 3.2 (右空左节点, 父节点)
       || RC[j] == i && RC[i] == n + 1 // case 3.3 (右空右节点, 父节点)
        w[i][j] = 1
        suffix_sum[i][j] = 1
    else
        w[i][j] = 0
        suffix_sum[i][j] = 0
endfor
// step 3: 计算后缀和
for k = 1 to ceil(log(n + 1)) do
    for all Pi,j par-do
        tmp = succ[i][j]     // 只是为了方便表示，临时变量不需要
        if tmp != NIL
            suffix_sum[i][j] += w[tmp.u][tmp.v]
            succ[i][j] = succ[tmp.u][tmp.v]
            Barrier // 避免w和succ脏读
            w[i][j] = suffix_sum[i][j]
    Barrier
endfor
// step 4: 用n + 1减去后缀和得到前缀和(位序)，排序数组写入B
for all Pi,j par-do
    if f[j] == i && RC[i] == j
       || LC[j] == i && RC[i] == n + 1
       || RC[j] == i && RC[i] == n + 1
        B[n + 1 - suffix_sum[i][j]] = A[i]
    if i == root && RC[i] == n + 1
        B[n] = A[i]
```

模型：需要共享存储，PRAM；LC、RC等需要CR；没有CW的需求，-> PRAM-CREW

时间复杂度：step 1,2,4为$O(1)$，step 3为$O(\log n)$

HW3

1

思路：判断0和1的交界

7.3

① 第二步使用对半搜索，$O(lgn)$。其他忽略。

② 略

7.6

① 注意是求运算量。用树的节点数来分析，n个叶子节点的满二叉树总节点数为$2n-1$

a. 如果赋值不算入运算

正向遍历：$n-1$

反向遍历：$\lceil\frac{2n-1}{2}\rceil-(lgn+1)=n-lgn-1$

b. 如果赋值算入，在a之外，加上赋值的数量：

初始化：n

正向遍历：$n-1$

反向遍历：$2n-1$

② 略

HW4

2.6

这里说明一下定义：有向图双向的边贡献为2，出1入1。

网络直径是$\min\max d(u,v)$ 图的直径是指任意两个顶点间距离的最大值（距离是两个点之间的所有路的长度的最小值）

节点度：4（由于定义不清晰，这个题答2，3，4的都算对，后续按定义来）

网络直径：$2n-1=5$

对剖宽度：$2^{n-1}=4$

2.7

节点度：2、4

网络直径：$2k=6$

对剖宽度：$2^k=8$

2.15

$proof$ 假设第i轮传播的所有信包大小为M_i，穿越l条链路，并且忽略节点延迟时间t_h（显然待证明结论中没有该项）根据教材上的公式：

$t_{comm-i}SF=t_s+(M_it_w+t_h)l=t_s+M_it_wl$

$t_{comm-i}CT=t_s+M_it_w+lt_h=t_s+M_itw$

每轮播送只经过一条链路，$l=1$，并且SF和CT方式对应的每次播送M_i是相等的，此时

$t_{comm-i}SF=t_{comm-i}CT$，记为$t_i$。开始时信包数为$p$个，每次播送一半的信包，因此$M_i=\frac{pm}{2^i}$

播送一共$\log p$轮，总时间为：

$$t_{one-to-all-pers}=\sum_{i=1}^{\log p}t_i=t_s\log p+\sum_{i=1}^{\log p}\frac{pm}{2^i}t_w=t_s\log p+mt_w(p-1)\quad Q.E.D$$

HW5

9.S1

参考

```
for all Pi,j par-do Ci,j = 0 endfor
for k = 0 to sqrt(p) - 1 do
    for all Pi,j par-do
        Ci,j = Ci,j + Ai,(i+j+k)modsqrt(p) * B(i+j+k)modsqrt(p),j
    endfor
endfor
```

时间复杂度：每个块做乘法的时间复杂度为$O((\frac{n}{\sqrt{p}})^3)$，一共做$\sqrt{p}$次，所以为$O(\frac{n^3}{p})$

9.9

（2.1）t_a

（2.2）n次：

乘-加：t_c

读a, b, c写c：$4t_a$

一共$t_a + n(t_c + 4t_a)$

（写明忽略某些读写情况的也可以）