The Design and Analysis of Algorithms
National Taiwan University, Fall 2020

Homework 6

Yu-Chieh Kuo B07611039

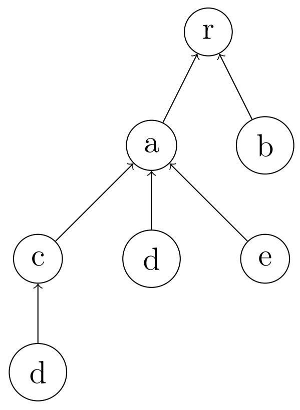
This homework answers the problem set sequentially.

1. We can use divide and conquer method to design the algorithm. For the divide part, we can divide the group into to sub-group, and make them compete with each other; for the conquer part, we make two sub-group play against each other. Let the variable $First$ be the number of the first player, $Last$ be the number of the last player, $Round$ be the total round to schedule, $Result$ be a $n \times (n-1)$ matrix, the algorithm can be written as below.

Algorithm 1 Round-Robin(First, Last, Round, Result)

```
if $Last := First + 1$ then
    Result$[First, Round] = Last$
    Result$[Last, Round] = First$
else
    $Middle := \frac{First+Last-1}{2}$
    $//Divide\ part$
    Round-Robin($First, Middle, \frac{Round+1}{2} - 1$, Result)
    Round-Robin($Middle + 1, Last, \frac{Round+1}{2} - 1$, Result)
    $//Conquer\ part$
    for $i = \frac{Round+1}{2} \dots Round$ do
        for $j = 0 \dots Middle - First$ do
            $Size := \frac{Round+1}{2}$
            $Player1 := First + j$
            $Player2 := (Middle + 1) + (i + j)\%size$
            Result$[Player1, i] = Player2$
            Result$[Player2, i] = Player1$
        end for
    end for
end if
```

2. The graph is as below.



3. The algorithm can be written as below.

Algorithm 2 Find(Array)

```
1: $n$ := the length of array
2: $total := \frac{n(n+1)}{2}$
3: $sum$ := summation of array
4: **return** $total - sum$
```

Since to count the summation of array needs n steps to traverse all array, the time complexity is $\Omega(n)$.

4. We have already known that the sum of the heights of all nodes in a full binary tree of height h is $2^{h+1} - h - 2$. Let $G(n)$ denote the sum of the heights of all nodes in a complete binary tree with n nodes. For a full binary tree (a special case of complete binary trees) with $n = 2^{h+1} - 1$ nodes where h is the height of the tree, we already know that $G(n) = 2^{h+1} - (h + 2) = n - (h + 1) \leq n - 1$. With this as a basis, we prove the general case of arbitrary complete binary trees by induction on the number $(n \geq 1)$ of nodes.
For base case $(n = 1 or n = 2)$: When $n = 1$, the tree is the smallest full binary tree with one single node whose height is 0. So,$G(n) = 0 \leq 1 - 1 = n - 1$. When $n = 2$,the tree has one additional node as the left child of the root. The height of the root is 1, while that of its left child is 0. So, $G(n) = 1 \leq 2 - 1 = n - 1$.
For inductive step $(n > 2)$: If n happens to be equal to $2^{h+1}1$ for some $h \geq 1, i.e.$ the tree is full, then we are done; note that this covers the case of $n = 3 = 2^{1+1} - 1$. Otherwise, suppose $2^{h+1} - 1 < n < 2^{h+2} - 1 (h \geq 1)$ *i.e.* the tree is a "proper" complete binary tree with height $h + 1 \geq 2$. We observe that at least one of the two subtrees of the root is full, while the other is complete (possibly full). There are three cases to consider:

   (a) The left subtree is full with n_1 nodes and the right one is complete but not full with n_2 nodes (such that $n_1 + n_2 + 1 = n$). In this case, both subtrees much be of height h and $n_1 = 2^{h+1} - 1$. From the special case of full binary trees and the induction hypothesis, $G(n_1) = 2^{h+1} - (h+2) = n_1 - (h+1)$ and $G(n_2) \leq n_2 - 1$. $G(n) = G(n_1) + G(n_2) + (h + 1) \leq (n_1 - (h + 1)) + (n_2 - 1) + (h + 1) = (n_1 + n_2 + 1) - 2 \leq n - 1$.

   (b) The left subtree is full with n_1 nodes and the right one is also full with n_2 nodes. In this case, the left subtree much be of height h and $n_1 = 2^{h+1} - 1$, while the right subtree much be of height $h-1$ and $n_2 = 2^h - 1$. From the special case of full binary trees, $G(n_1) = 2^{h+1} - (h+2) = n_1 - (h+1)$ and $G(n_2) = 2^h - (h+1) = n_2 - h$. $G(n) = G(n_1) + G(n_2) + (h+1) \leq (n_1 - (h+1)) + (n_2 - h) + (h+1) = (n_1 + n_2 + 1) - (h+1) \leq n - 1$.

(c) The left subtree is complete but not full with n_1 nodes and the right one is full with n_2 nodes. In this case, the left subtree much be of height h, while the right subtree much be of height $h-1$ and $n_2 = 2^h - 1$. From the induction hypothesis and the special case of full binary trees, $G(n_1) \leq n_1 - 1$ and $G(n_2) = 2^h - (h+1) = n_2 - h$. $G(n) = G(n_1) + G(n_2) + (h+1) \leq (n_1 - 1) + (n_2 - h) + (h+1) = (n_1 + n_2 + 1) - 1 = n - 1$.

5. The next function can be adapt as below.

Algorithm 3 NewNext(B, m, next)

```
for $i = 3 \ldots m$ do
    if B[next[$i$]+1] == B[$i$] then
        next[$i$] := next[next[$i$] + 1]
    end if
end for
```

The *next* table will be

1	2	3	4	5	6	7	8	9
a	b	a	a	b	a	b	a	a
-1	0	-1	1	0	-1	3	-1	1