Homework 4

Lei Zhang

March 2, 2016

1 Exercises 4.3 - 4

a.

For $n = 2, A[1, 2]$:
[1,2], [2,1]

For $n = 3, A[1, 2, 3]$:
[1,2,3], [2,1,3], [3,1,2], [1,3,2], [2,3,1], [3,2,1]

For $n = 4, A[1, 2, 3, 4]$:
[1,2,3,4], [2,1,3,4], [3,1,2,4],[1,3,2,4],[2,3,1,4],[3,2,1,4], [4,2,3,1], [2,4,3,1], [3,4,2,1], [4,3,2,1], [2,3,4,1],[3,2,4,1], [4,1,3,2], [1,4,3,2], [3,4,1,2], [1,3,4,2], [3,1,4,2], [4,1,2,3],[1,4,2,3], [2,4,1,3], [4,2,1,3],[1,2,4,3], [2,1,4,3]

b.

Basis: For n = 1, the output of Heap's algorithm is [1], which are all permutations of [1]. Thus the statement is true for n =1.

Inductive step: Assume HeapPermute(n) generates all permutations of A[1..n] holds for some unspecified value of k. There are k + 1 loops in HeapPermute(k+1). For each loop, the additional number k+1 is added to all the possible positions in the permutations generated from HeapPermute(k). At last, in total (k+1)! permutations are generated from HeapPermute(k+1).

Since both the basis and the inductive step have been performed, by mathematical induction, the statement holds for all natural numbers n. Q.E.D.

c.

For a positive integer n, there are $n!$ permutations of $A[1..n]$. In other words, the Heap's algorithm generates $n!$ permutations through about $n!$ swap processes. Thus, $C(n) \in \Theta(n!)$.

2 Exercises 4.3 - 7

//**Input:** A positive integer, n
//**Output:** all 2^n bit strings of length n, A[0..n]

function BITSTRINGS(n, A)
    if $n = 0$ **then**
        write A
    end if
    $A[n] \leftarrow 0$
    $BitStrings(n-1, A)$
    $A[n] \leftarrow 0$
    $BitStrings(n-1, A)$
end function

3 Exercises 4.2 - 1

a.

The order the vertices are popped off the stack: e, f, g, b, c, a, d
Thus the sorted order is: d,a,c,b,g,f,e

b.

This digraph is not a dag.

4 Exercises 4.2 - 9

a.

- Step 1: The order the vertices are popped off the stack: 1f, 2g, 3b, 4a, (now empty), 5d, 6c, (now empty), 7h, 8e
- Step 2:
- Step 3: The order vertices are pushed in the stack: e, h, c, (now empty), d, (now empty), h, g, f, a. Thus, the strongly connected components are {e, h, c}, {d}, {a, f, g, b}

b.

For adjacency matrix representation, the time efficiency class of this algorithm is $\Theta(|V|^2)$

For adjacency list representation, the time efficiency class of this algorithm is $\Theta(|V| + |E|)$

c.

Zero, as there's no strongly connected component can be contracted to a single vertex.

5 Exercises 4.4 - 11

a.

m	n	
26	47	
13	94	94
6	188	
3	376	376
1	752	752

The result is $94 + 376 + 752 = 1222$

b.

It does matter. If we multiply n by m, the number of basic operations is $\lfloor log_2 n \rfloor$. If we multiply m by n, the number of basic operations is $\lfloor log_2 m \rfloor$.

6 Exercises 4.4 - 12

a.

//**Input:** Two positive integers, n and m
//**Output:** The product of n and m

```
function RussianPeasant(n, m)
    if n = 1 then
        return m
    else if n mod 2 = 0 then
        return RussianPeasant(n/2, 2m)
    else
        return RussianPeasant((n − 1)/2, 2m) + m
    end if
end function
```

b.

$\Theta(logn)$, where n is the first factor of the product.