Advanced Algorithms

2015PEAA6

Name: Anurag Nandi K
USN: 1BM17CS017
Section: 5A
Date: 2-12-2020

Part-A

1a) Naive string matching

Pseudocode:

```
Naive_String_Matcher (T, P)
        n ← length [T]
        m ← length [P]
        for s ← 0 to n-m
            do if P [1....m] = T [s+1. ....s+m]
                then print "Pattern occurs with shift" s
```

c++ code:

```
#include <bits/stdc++.h>
using namespace std;
void search (char* txt, char* pat)
{
    int m= strlen (pat);
    int n = strlen (txt);
    for (int i=0; i<=n-m; i++)
    {
```


Answer

Anusree Manoj K
1BM18CS019

```
        for(int j = 0 ; j<m ;j++)
        {
            if (txt[i+j] != pat[j])
                break;
        }
        if (j == m)
            cout << " Pattern occurs with shift "<< i << endl;

    }
}
int main ()
{
    char txt[ ] = " ABAA BACAAB "
    char pat[ ] = " BACA "
    search (pat, txt);
    return 0;
}
```

Output :

Pattern occurs with shift 4

Anusree

Anusree Manoj K
1134181CS017

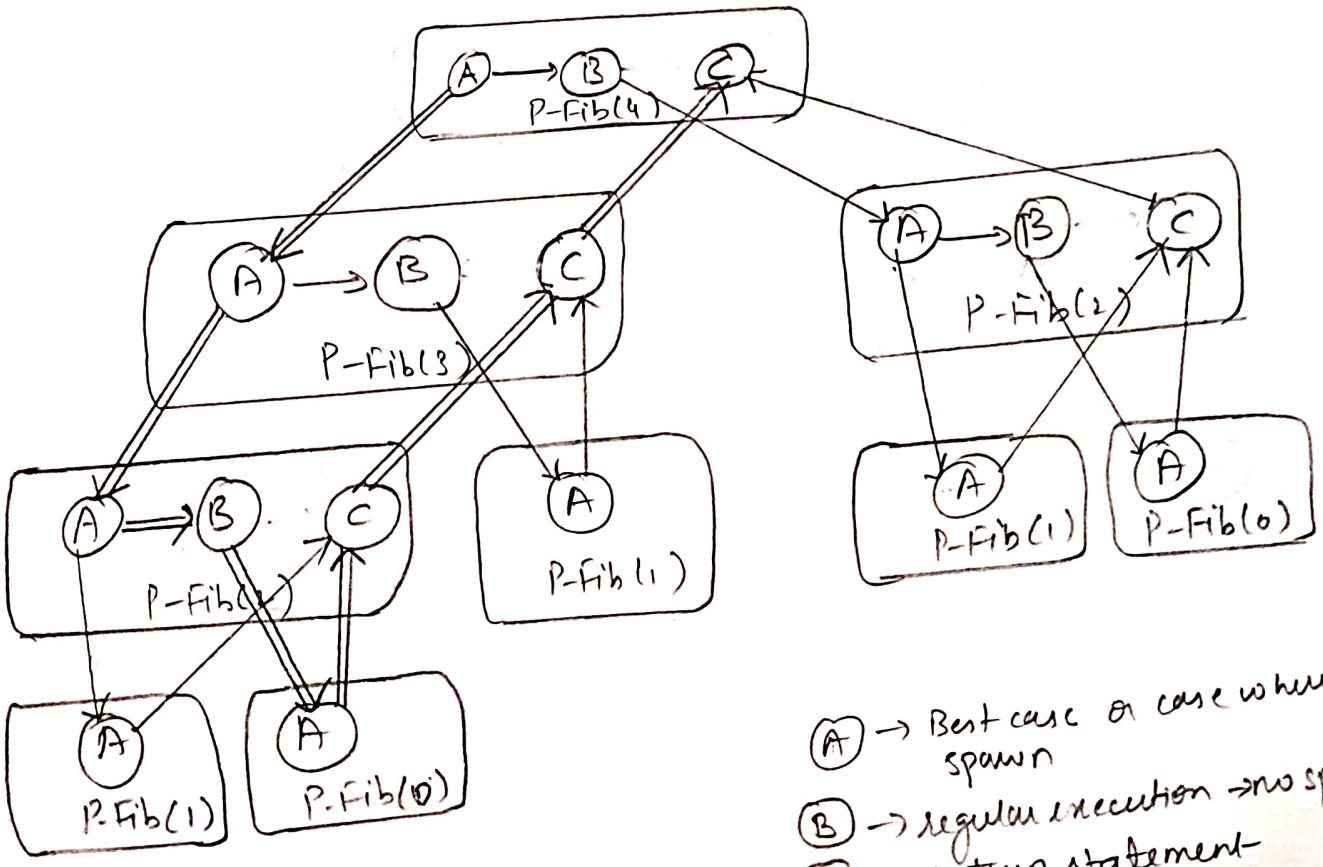
Part-B

2a) Multithreaded ~~fibo~~ Fibonacci number generation (n=4)

~~Fibo~~ Multithreaded algorithm:

```
P-Fib(n)
    if n ≤ 1
        return n
    else  x = spawn P-Fib(n-1)
          y = P-Fib(n-2)
          sync
          return x+y
```

Fibonacci (4) = P-Fib(4)



Anusree Manoj K
IBM18CS017

SPAN

The span is the ~~largest~~ longest time to execute the threads along any path of the computational DAG.

Span = The number of vertices on a longest or critical path (path shown as double-arrows in the figure)

span = 8 time units (Assuming unit time for each thread)

WORK

The work of a multithreaded computation is the total time to execute the entire computation on one processor

work = sum of the times taken by each thread
= No. of vertices in DAG [∵ No of vertices = No. of threads]

work = 17 time units (Assuming unit time for each thread and 17 vertices = 17 threads)

Anusree

Anusree Manoj K
IBM18CS017

25) Multithreaded Matrix Multiplication algorithm

```
Matrix_Multiply ( C, A, B, n):
// Multiplies matrices A and B, storing the result in C
// n is power of 2 (for simplicity)
    if n == 1
        C[1,1] = A[1,1] . B[1,1]
    else
        allocate a temporary matrix T[1...n, 1....n]
        partition A, B, C and T into (n/2)*(n/2) sub-matrices
        spawn Matrix_Multiply (C11, A11, B11, n/2)
        spawn Matrix_Multiply (C12, A11, B12, n/2)
        spawn Matrix_Multiply (C21, A21, B11, n/2)
        spawn Matrix_Multiply (C22, A21, B12, n/2)
        spawn Matrix_Multiply (T11, A12, B21, n/2)
        spawn Matrix_Multiply (T12, A12, B22, n/2)
        spawn Matrix_Multiply (T21, A22, B21, n/2)
        Matrix_Multiply (T22, A22, B22, n/2)
        sync
        Matrix_Add (C, T, n)
```


Anusree

Anusree Manoj K
IBM18CS017

```
Matrix_Add (C, T, n):
    // Adds matrices C and T in-place, producing C = C+T
    // n is power of 2 (for simplicity)
    if n == 1:
        C[1,1] = C[1,1] + T[1,1]
    else:
        partition C and T into (n/2) x (n/2) submatrices
        spawn Matrix_Add (C11, T11, n/2)
        spawn Matrix_Add (C12, T12, n/2)
        spawn Matrix_Add (C21, T21, n/2)
        spawn Matrix_Add (C22, T22, n/2)
        sync
```

Representation:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

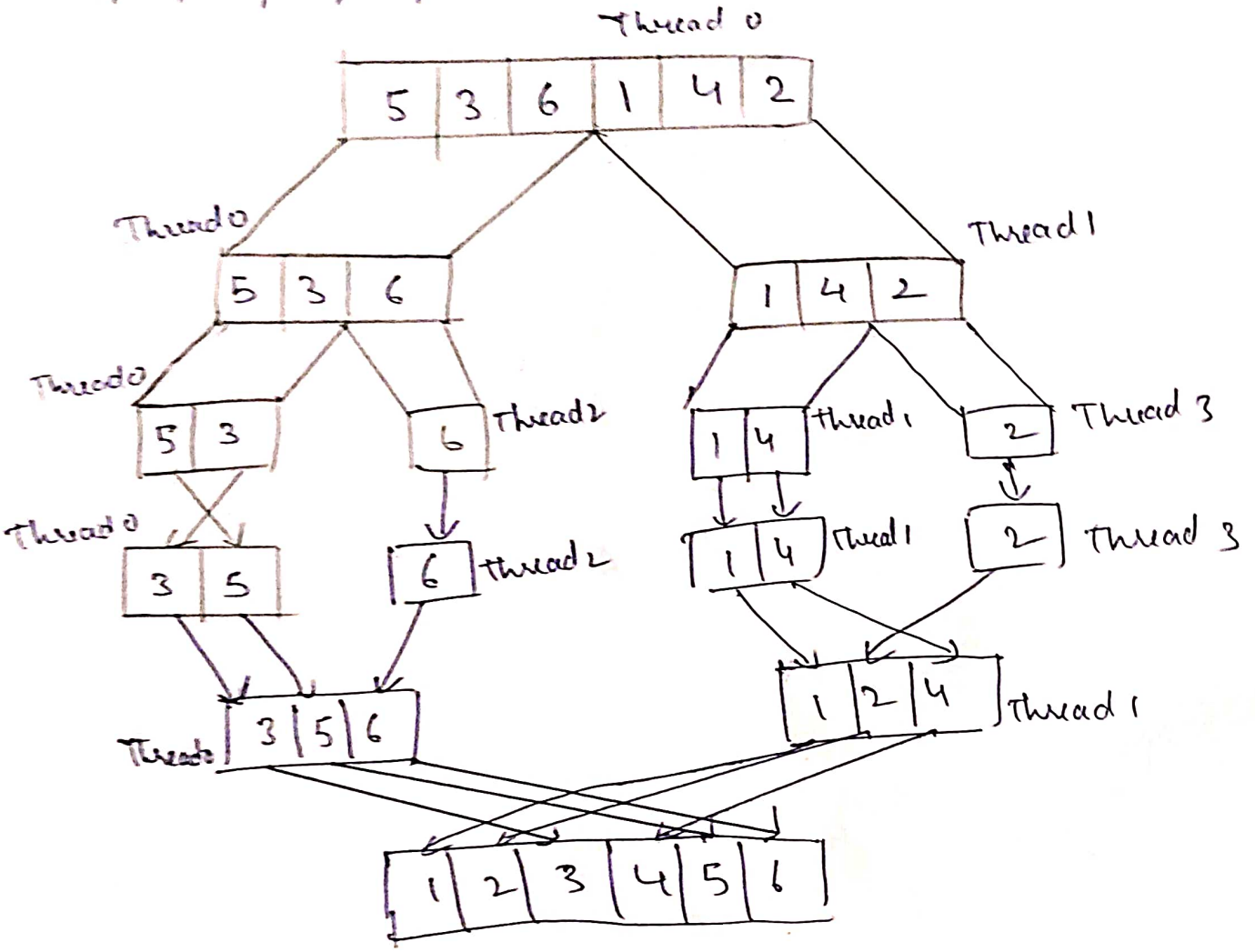
$$= \begin{bmatrix} \underset{C_{11}}{A_{11}B_{11}} & \underset{C_{12}}{A_{11}B_{12}} \\ \underset{C_{21}}{A_{21}B_{11}} & \underset{C_{22}}{A_{21}B_{12}} \end{bmatrix} + \begin{bmatrix} \underset{T_{11}}{A_{12}B_{21}} & \underset{T_{12}}{A_{12}B_{22}} \\ \underset{T_{21}}{A_{22}B_{21}} & \underset{T_{22}}{A_{22}B_{22}} \end{bmatrix}$$

Anusree

Anusree Manoj K
IBM18CS017

2c) Multithreaded merge sort

5, 3, 6, 1, 4, 2



Anusree

Anusree Manoj K
IBM18CS017

Part C

3a) Algorithm for string matching using Rabin Karp approach

```
Rabin_Karp_Matcher (T, P, d, q)
// T is text string, P is pattern, d is radix, q is modulus (prime)

n ← length [T]
m ← length [P]
h ← d^(m-1) mod q  // higher order digit position for m-digit
                   // window

p ← 0
t0 ← 0
for i ← 1 to m                    // Preprocessing
    do p ← (dp + P[i]) mod q
       t0 ← (dt0 + T[i]) mod q

for s ← 0 to n-m
    do if p = ts
       then if P[1..m] = T[s+1... s+m]
            then " Pattern occurs with shift" s

       if s < n-m
       then ts+1 ← (d(ts - T[s+1] h) +
                    T[s+m+1]) mod q
```


Anusree

Anusree Manoj K
IBHI8CS017

Given T = "ababaabbab"

P = "abba"

d = 26, q = 13

$h(abba) = h(a, b, b, a)$

$= (1\times10^3 + 2\times10^2 + 2\times10^1 + 1\times10^0) \bmod 13$

$= (1000 + 200 + 20 + 1) \% 13$

$= (1221) \% 13 = 12$

a - 1
b - 2
c - 3
d - 4
e - 5

~~first substring from T = abab~~

$h(abab) = (1\times10^3 + 2\times10^2 + 1\times10 + 2\times10^0) \% 13$

$= (1000 + 200 + 10 + 2) \% 13$

$= 1212 \% 13 = 3 \neq 12$

$h(baba) = (2\times10^3 + 1\times10^2 + 2\times10^1 + 1\times10^0)$

$= (2121) \% 13 = 2 \neq 12$

$h(abaa) = (1\times10^3 + 2\times10^2 + 1\times10^1 + 1\times10^0)$

$= 1211 \% 13 = 2 \neq 12$

$h(baab) = (2\times10^3 + 1\times10^2 + 1\times10^1 + 2\times10^0)$

$= 2112 \% 13 = 6 \neq 12$

$h(aabb) = (1\times10^3 + 1\times10^2 + 2\times10^1 + 2\times10^0)$

$= 1122 \% 13 = 4 \neq 12$

$h(abba) = (1\times10^3 + 2\times10^2 + 2\times10^1 + 1\times10^0)$

$= (1221) \% 13 = 12 = 12$ //

∴ Pattern occurs at shift 5

Answer

Anusree Manoj K
IBM18CS017

4a) String matching using Horspool's technique

```
Horspool_Matching (P[0...n-1], T[0...n-1])
    // Input: Pattern P[0..m-1] and text T[0...n-1]
    // Output: index of first matching substring, -1 if no match
    ShiftTable (P[0...m-1]) // generate Table of shifts
    i ← m-1
    while i ≤ n-1 do
        k ← 0
        while k ≤ m-1 and P[m-1-k] = T[i-k] do
            k ← k+1
        if k = m
            return i-m+1
        else i ← i + Table[T[i]]
    return -1
```


Anusree

Anusree Manoj K
1TBMI19CS017

given T = " praveen prashanathpradan"

P = " prada "

P = p r a d a
0 1 2 3 4

Shift table

p	r	a	d	*
4	3	2	1	5

e=shifts 5 h=shifts 5 h=shifts 5

p	r	a	v	e	e	n	p	r	a	s	h	a	n	a	t	h	p	r	a	d	a	n
p	r	a	d	a																		
					p	r	a	d	a													
							p	r	a	d	a											
												p	r	a	d	a						
																	p	r	a	d	a	

pattern matched at index 17

Anusree