CSE 340 Bütünleme

Olcay Taner YILDIZ

I. Question (15 points)

Given an even number N, write the parallel method

```
int goldbach(int N)
```

which checks if N can be expressed as the sum of two prime numbers. For example, 24 can be written as the sum of two prime numbers 5 and 19; 38 can be written as the sum of two prime numbers 7 and 31.

II. Question (20 points)

Solve Question I by dividing the jobs into the processors equally. Use manager and worker paradigm to accomplish this.

III. Question (15 points)

Consider the following algorithm to generate a sequence of numbers. Start with an integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when n = 1. For example, the following sequence of numbers will be generated for n = 12.

```
12 6 3 10 5 16 8 4 2 1
```

Write the method

```
int* create_sequence(int N)
```

which generates the sequence and returns it as an array for the number N.

IV. Question (15 points)

Write the parallel method

```
int second_largest(int* a)
```

which finds the second largest element of the array. Assume that array a is stored in an array and all processors have n/p elements of the array. Assume also that the array a is too large, it can not be stored in one processor. Processor 0 will print the result.

V. Question (20 points)

Write the parallel method

```
int is_super_increasing(int* a)
```

which checks if the array a is super increasing or not. An array is super increasing if each element $a[i]$ is larger than the sum of all the previous elements ($a[0] + a[1] + \ldots + a[i-1]$). For example the sequences

```
1 2 4 8 16 32 64
```

```
1 3 6 15 36 67
```

are super-increasing. Assume that array a is stored in an array and all processors have n/p elements of the array and the elements are stored in the increasing order of processor id's. Assume also that the array a is too large, it can not be stored in one processor. Processor 0 will print the result.

VI. Question (15 points)

Write the function *scatter_all* which scatters the elements of the array a stored in the processor 0 in the following way: Processor 0 will send 1 element to processor 0, 2 elements to processor 1, 1 element to processor 2, 2 elements to processor 3, 1 element to processor 4, 2 elements to processor 5, etc.