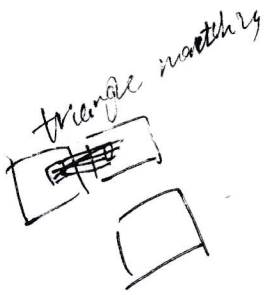


SHANGHAITECH UNIVERSITY

2019-2020 SEMESTER 2 FINAL EXAMINATION

CS121 PARALLEL COMPUTING

July 2020

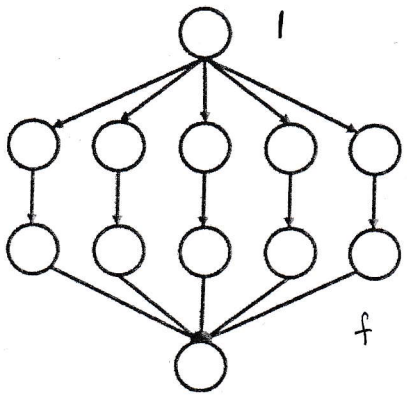
Time Allowed: 2 hours

Write your answers on additional sheets of paper. Staple all your answer sheets together and clearly label them with your name and student ID.

In all problems in which you are asked to design algorithms, you should clearly describe how your algorithm works, provide code or pseudocode when asked to, and argue why your algorithm is correct.

All answers must be written neatly and legibly in English.

1a. Consider the task graph shown below. Each node represents a subtask which takes unit time to perform. An arrow from one subtask to another means the first subtask must finish before the second subtask can start. What is the maximum speedup for performing this task using two identical processors?



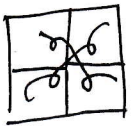
(A) 3/4 (B) 2 (C) 4/3 (D) 12/7

(5 points)

1b. For the same task as in question 1a, what is the least number of processors needed to achieve the maximum possible speedup?

(A) 12 (B) 8 (C) 5 (D) 4

(5 points)



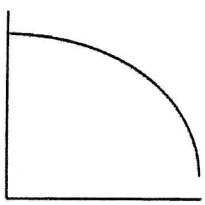
1c. Suppose you want to write an MPI program to transpose a matrix. That is, given a matrix M, you want to output a matrix M' such that $M'[i][j] = M[j][i]$ for all i, j. Assume the matrix is initially partitioned row-wise across the processors. Which of the functions below lets you perform the operation in the simplest, and likely most efficient way?

(A) MPI_Send (B) MPI_Alltoall (C) MPI_Scatter (D) MPI_Bcast

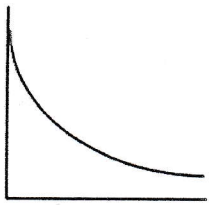
(5 points)

1d. Suppose you have a parallel program which performs matrix multiplication. For a fixed input, which of the following graphs can represent the running time of the program as a function of the number of processors? Assume there is no anomalous behavior such as superlinear speedup.

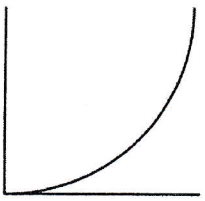
(A)



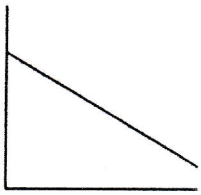
(B)



(C)



(D)



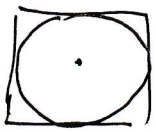
(5 points)

2. Write a parallel OpenMP program to compute the prime factorization of an input value, outputting the factors in an array. For example, if the input is 60, then your program can output [2, 2, 3, 5]. Your program must output a complete factorization of the input, though the factors do not need to be in sorted order. Optimize your program for performance.

Note: For efficiency, your program may need to break out of a `parallel for` loop before finishing all the loop iterations. In OpenMP, this can be done by placing a `#pragma omp cancel for` statement inside the for loop after an `if` statement checking the break condition.

(20 points)

$(x-\frac{1}{2})^2+(y-\frac{1}{2})^2 <= \frac{1}{4}$ section



i = argv[1]

3. Consider the following Monte Carlo algorithm for computing π. Form a square region bounded by the points (0,0), (0,1), (1,1), and (1,0) and consider the largest circle inscribed within the square. Generate a number of random points, and compute the fraction p of points that fall within the circle. As the number of points increases, the value of p approaches $\pi/4$.

Write a parallel OpenMP program to implement this algorithm. Use the program template shown in the figure below. The number of samples to use is given in argv[1]. To generate random numbers, the template includes a (fictitious) header file random.h, which defines a function random() returning a random **double** value between 0 and 1.

```
#include <stdio.h>
#include <omp.h>
#include <random.h>

main(int argc, char *argv[]) {
   int i;
   int samples;
   double x,y;
   double pi;

   /* complete this code */


   printf("Estimate of pi: %7.5f\n", pi);
}
```

(random(), random())

(20 points)

4. Given a sequence of numbers $x_1, \ldots, x_n$ and an integer $0 < k \leq n$, we want to compute the maximum values in all windows of size k in parallel. That is, for all $i \in [1, n-k+1]$, we want to compute the quantity $\max\{x_i, \ldots, x_{i+k-1}\}$. Note this produces a total of $n-k+1$ output values.

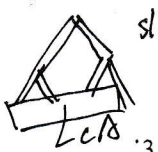
Using $O(n)$ processors, there is a trivial parallel algorithm for this problem which takes $O(k)$ parallel time and performs $O(nk)$ work. However, there exist faster parallel algorithms which also do asymptotically less work.

Write an efficient CUDA program for solving this problem. Clearly describe the algorithm you use and also briefly describe your CUDA implementation. You will not be marked down for minor CUDA syntax errors.

(20 point)

[ ] sliding window

[ ]



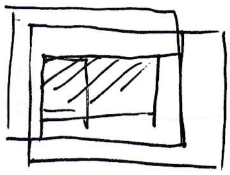
prefix sum

LCA

least close ancester.

5. Consider an $n \times n$ matrix A of numbers and an integer $0 < k \leq n$. We want to compute the sums of the values in all $k \times k$ windows in A. That is, for all $1 \leq p, q \leq n - k + 1$, we want to compute the sum $\sum_{p \leq i \leq p+k-1, q \leq j \leq q+k-1} A_{i,j}$. Describe a PRAM algorithm for doing this which uses n^2 processors, and runs in $O(\log n)$ parallel time and does $O(n^2)$ total work.

(20 points)



dept · n^2 depth=1

P×P [illegible]

END OF EXAM