CS 406
HOMEWORK 1
1.04.2023

Name: *Tacettin Emre Bök*
ID: *28091*

1. If the **r** of runtime of a serial program is *"perfectly parallelized"*, and **1-r** is *"inherently serial"*, this means that **r** of the serial runtime can bu executed in parallel. So, according to Amdahl's Law maximum speedup one can have is

$$\lim_{x \to \infty} 1/(1-r) + r/x$$

   Where x is the number of processors. This limit equals to $1/1-r$. So, the upper bound on the speedup one can have is $1/1-r$

2. **a)** Memory latency is the time it takes from making a request for a byte or word in memory until it is retrieved by the processor. On the other hand, Memory bandwidth is the rate at which data can be read from or store in the memory by a processor

   b) Spatial locality refers to the use of the items within close storage locations in the memory, whereas temporal locality refers to the repetitive use of a specific data or resources in a small amount of time.

   c) In Shared Address Space memory model, a part or all of the memory in the system is accessible to all processors directly. On the other hand, Distributed Address Space is the model in which every processor in the system have their own address spaces, and processors can access directly only to their own address space.

3. **a)**

   Fig 3.1: in Figure 3.1 the total number of tasks in the family of graphs is $N = n^2$ where n is equal to the size of each dimension of the square grid. The longest path T is the path that starts from to top-left corner node of the graph and ends in the bottom-right node of the graph. The length of this path is equal to $(n-1)(n-1) = 2n-2$ in terms of n and it is equal to $2\sqrt{N}-2$ in terms of N

   Fig 3.2: In Figure 3.2 total number of tasks in the family of graphs is equal to $N = 2^n - 1$ where n is equal to the height of the tree. The longest path T is the path that starts form the root node of the tree and ends in a leaf node. The length of this path is height of the tree which is equal to n. In terms of N, the length of the longest path is equal to $\log_2(N+1)$.

   b)

   Fig 3.1: In Figure 3.1 the degree of concurrency is equal to n which is the width of the square. In terms of N, this equals to $\sqrt{N}$

   Fig 3.2: In Figure 3.2 the degree of concurrency is equal to number of leaf nodes which is equal to 2^{n-1}. In terms of N, this equals to $2^{\log_2 N+1-1}$

c)

Fig 3.1: The **speedup** is equal to T_1/T_p where T_p is equal to the time with p number of processor(s). In Figure 3.1, T_1 is equal to n^2 because we have n^2 tasks to execute and one processor can execute one at a time, and T_n is equal to $2*(n-1)+1$ because the maximum degree of concurrency is n and if there are n processing elements they can execute at most n tasks in parallel (the most tasks executed concurrently happens in the diagonal of the square). Up to that point there are $n-1$ steps where system executes $1, 2, 3...n-1$ tasks in parallel. The time spent in there is $n-1$ as n processing elements can execute each of them at once (as the maximum concurrent tasks executed is n-1). After here we execute the tasks in the diagonal, this takes 1 time. After executing the tasks at the diagonal we have another $n-1$ since its symmetric. So in total, $T_p = T_n = n-1+1+n-1 = 2n-1)$. Finally **speedup** is equal to

$$\frac{n^2}{(2n-1)}$$

In terms of N, its equal to

$$\frac{N}{(2\sqrt{N}-1)}$$

Additionally, **efficiency** is calculated as $T_1/(p*T_p)$. So, it is equal to

$$\frac{n^2}{n*(2n-1)}$$

In terms of N, its equal to

$$\frac{N}{\sqrt{N}*(2\sqrt{N}-1)}$$

Fig 3.2: In Figure 3.2 the **speedup** is equal to

$$\frac{2^n-1}{n}$$

In terms of N, its equal to

$$\frac{2^{\log_2 N+1}-1}{\log_2 N+1}$$

because there are 2^n-1 tasks and one processor can execute one task at a time, and if we have 2^{n-1} processing elements, these processors can execute the tasks that are horizontally in the same layer at once, in parallel. This takes n because we have n such layer. In a similar way, **efficiency** is equal to

$$\frac{2^n-1}{2^{n-1}*n}$$

In terms of N, its equal to

$$\frac{2^{\log_2 N+1} - 1}{2^{\log_2 N+1-1} * \log_2 N + 1}$$

4. The code is:

```cpp
#include <iostream>
#include <memory>

using namespace std;

void reverse(int *arr, int length) {
    #pragma omp parallel for
    for (int i = 0; i < length/2; i++) {
        int temp = arr[i];
        arr[i] = arr[length-i-1];
        arr[length-i-1] = temp;
    }
}

int main(){
    int N = 1000000;
    int *arr = new int[N];

    for(int i = 0; i < N; i++){
        arr[i] = rand();
    }
    reverse(arr, N);
}
```

5.

6. **a)** If we do not include **-fopenmp**, the code is not executed in parallel. So, as it is executed sequentially, output will always be 5. So, 5 is the both minimum and maximum value the function party can return.

   b) If we include **-fopenmp** the for loop will be executed in parallel by 3 threads. This creates a race condition where each thread wanting to increment the value of a. This will result in unpredictable values of a. Maximum value that party function can return is 15 which occurs when no increment disappears due to write-read conflict and each thread adds their amount (5) successfully. On the other hand, minimum value that party function can return is 5 which occurs when missing increments due to write-read conflicts happens and some of the increments are missed (at most 10 of them.)