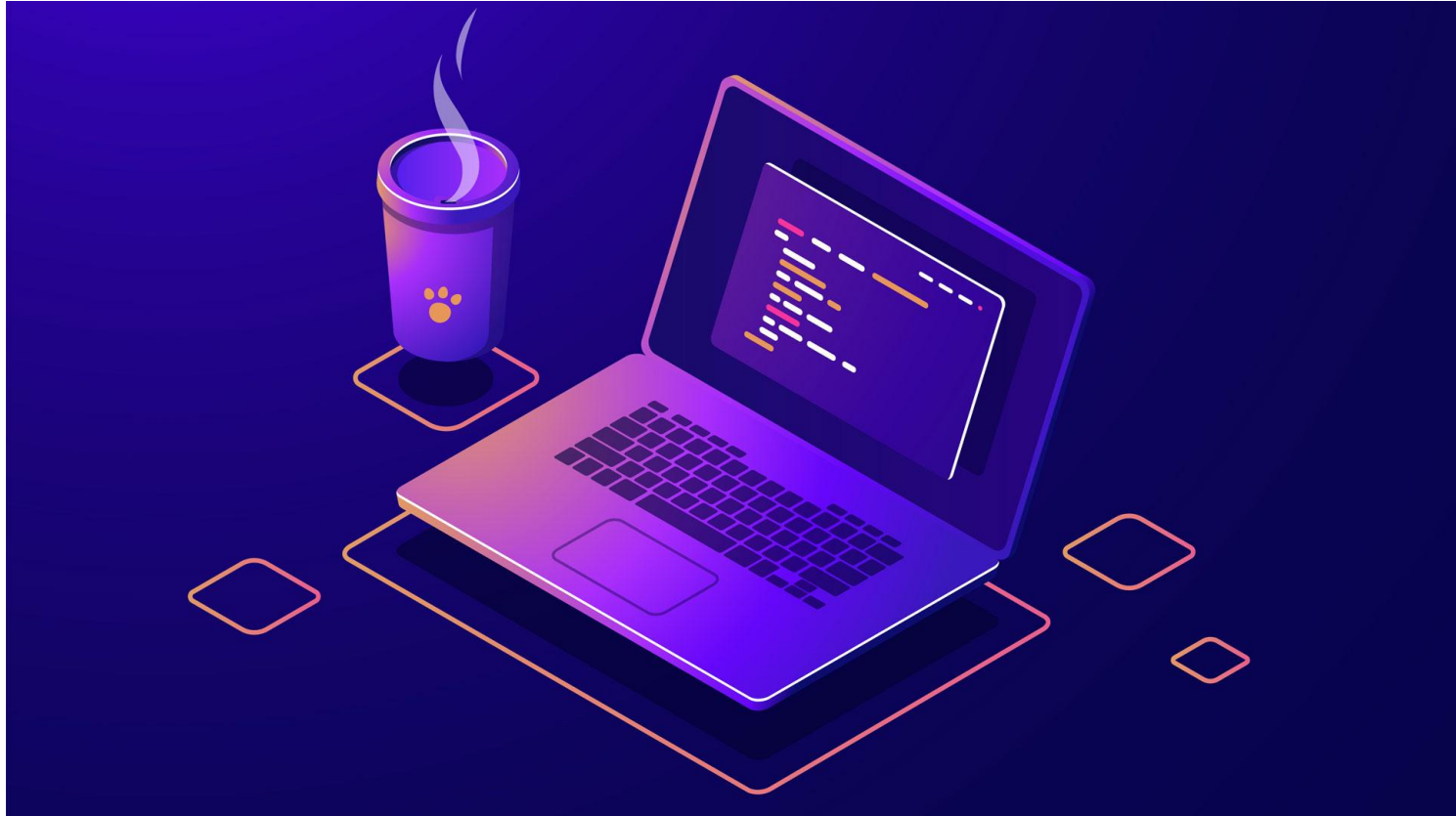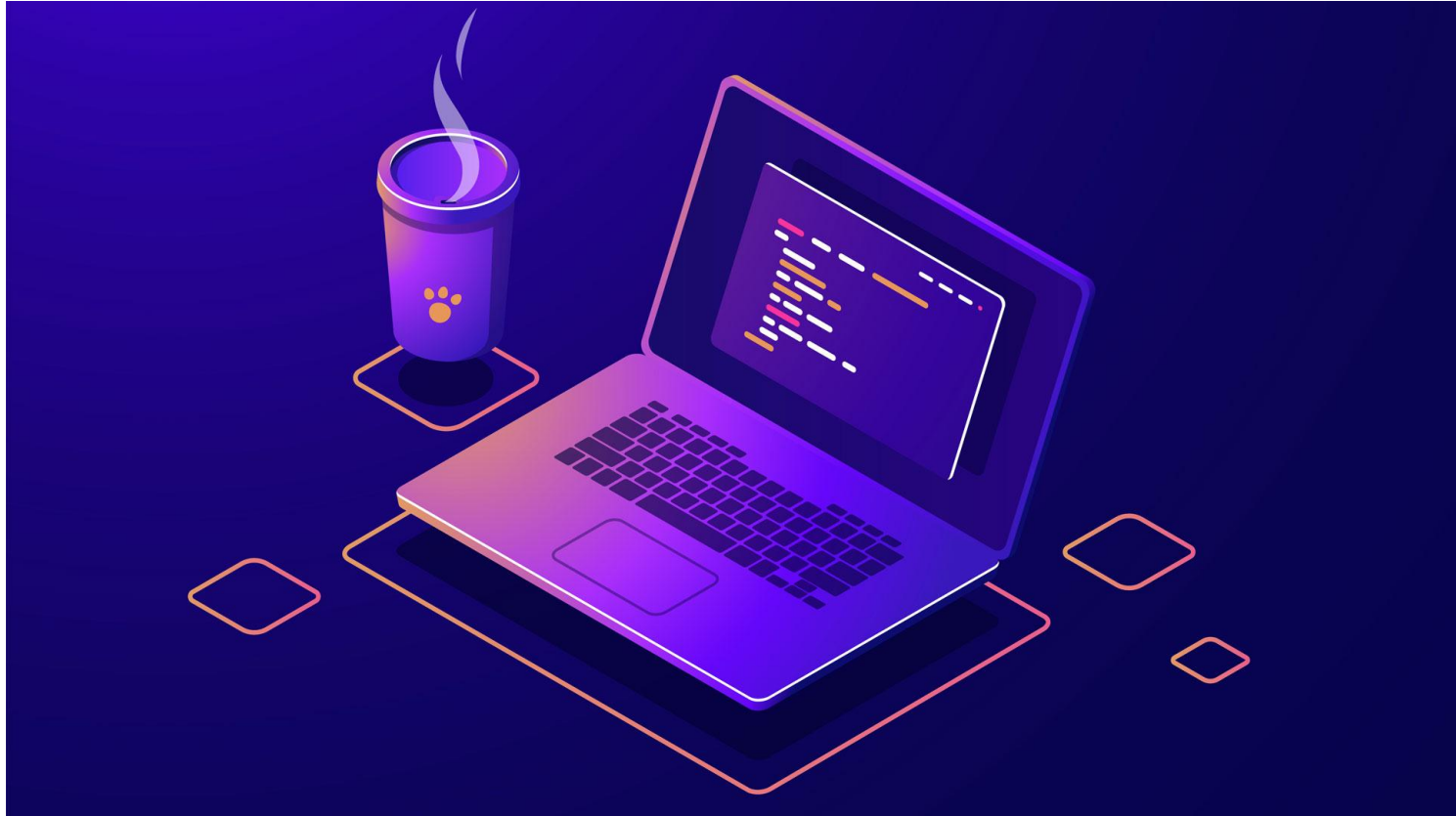# Parallelism Fundamentals Problems

Problem 4

Problem 4

Given the following code:

Problem 4

```
# define N 4
int m[N][N]

// initialization
for (int i = 0; i < N; i++) {
    start_task("for_initialize");
    for (k = i; k < N; k++)
        if(k==i) modify_d(&m[i][i],i,i);
        else {
            modify_nd(&m[i][k],i,k);
            modify_nd(&m[k][i],k,i);
        }
    }
    end_task("for_initialize");
}
```

Problem 4

```
// computation

for (int i = 0; i < N; i++) {
    start_task("for_compute");
    for (k = i+1; k < N; k++)
        int temp = m[i][k];
        m[i][k] = m[k][i];
        m[k][i] = temp;
    }
    end_task("for_compute");
}
```

Problem 4

```
// print results

start_task("output");
print_results(m);
end_task("output");
```

Assuming that: 1) the execution of the `modify_d` routine takes 10 time units and the execution of the `modify_nd` routines takes 5 time units; 2) each internal iteration of the computation loop (i.e. each internal iteration of the *for_compute* task) takes 5 time units; and 3) the execution of the *output* task takes 100 time units, **we ask:**

(a) Draw the Task Dependence Graph (TDG), indicating for each node its cost in terms of execution time (in time units).

(b) Compute the values for T_1, T_∞ and the potential *Parallelism*, as well as the parallel fraction (*phi*).

(c) Indicate which would be the most appropriate task assignment on two processors in order to obtain the best possible "speed up". For that assignment, calculate T_2 and S_2.

Solutions

a)

Task Dependence Graph:
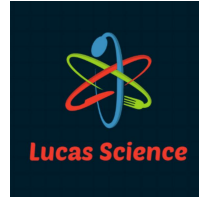
Solutions

a)
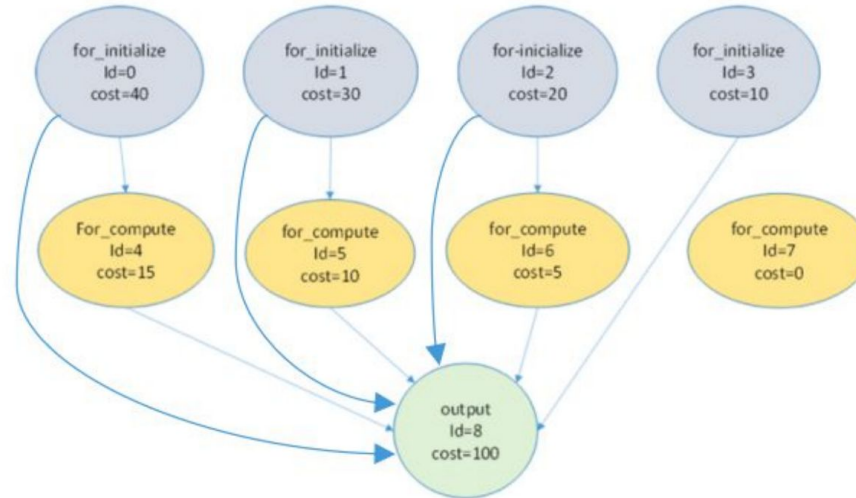
task for_initialize (i=1, cost=30)

task for_compute (i=1, cost=10)

task print_results (cost=100)

Instructor Social Media

Youtube: Lucas Science

Instagram: lucaasbazilio

Twitter: lucasebazilio