Exercise 2

Time tip: Roughly 45sec to 1min per 1pt
Q1) [3 x 2pts] Suppose an SQL query takes 20min to run on a single worker node and x min when run on 5 worker nodes. What is the speedup for the given value of x? Is the speedup linear, sublinear, or superlinear?
A. x = 7min
B. x = 4min
C. x = 3min

A. 20/7 ~ 2.9x; sublinear
B. 20/4 = 5x; linear
C. 20/3 ~ 6.7x; superlinear
Exercise

Q2) [3 x 2pts] Suppose an ML training procedure takes 40min to run on a single worker node. We then triple the dataset size, say, to help improve accuracy and use 3 worker nodes. It now takes $x$ min. What is the scaleup for the given value of $x$? Is the scaleup linear, sublinear, or superlinear?

A. $x = 50$min \hspace{1cm} A. $40/50 = 0.8x$; sublinear
B. $x = 40$min \hspace{1cm} B. $40/40 = 1x$; linear
C. $x = 35$min \hspace{1cm} C. $40/35 \approx 1.1x$; superlinear
Exercise

Q3) Consider the following task graph with the task lengths shown. You are given 3 workers to execute this graph in a task-parallel manner like discussed in class

A. [2pts] What is the lowest possible completion time of this workload?
B. [4pts] What is the highest possible speedup of this workload on 3 workers vs 1 worker?
C. [4pts] What is the total idle time across all workers in a schedule that yields the highest speedup?

A. Longest path from input(s) to end; in this case, it is T2 -> T5 -> T6; so, 15+5+5 = 25 units
B. Time on 1 worker is just sum of all task lengths: 5+15+20+10+5+5 = 60 units; lowest possible completion time on 3 workers is 25; so, highest possible speedup is 60/25 = 2.4x
C. Total “on time” across 3 workers is 25*3=75; of these, 60 units are for work; so, total idle time across all workers is 75-60 = 15 units
Q4) [4pts] Suppose you are given a workload with \( n \) tasks, each of length \( k \) units. You are allowed to use task parallelism as discussed in class. What is the lowest possible completion time of this workload? What should the task graph look like and what is the number of workers needed to achieve that completion time?

Lowest possible completion time is just \( k \) units, which is the length of any one given task. Note that tasks are indivisible in task parallelism as discussed in class.

The task graph should have no edges among the \( n \) vertices, i.e., no dependencies among any tasks to get the above completion time.

We need \( n \) workers to get the above completion time. Basically, each task runs on its own worker independently from the other tasks.