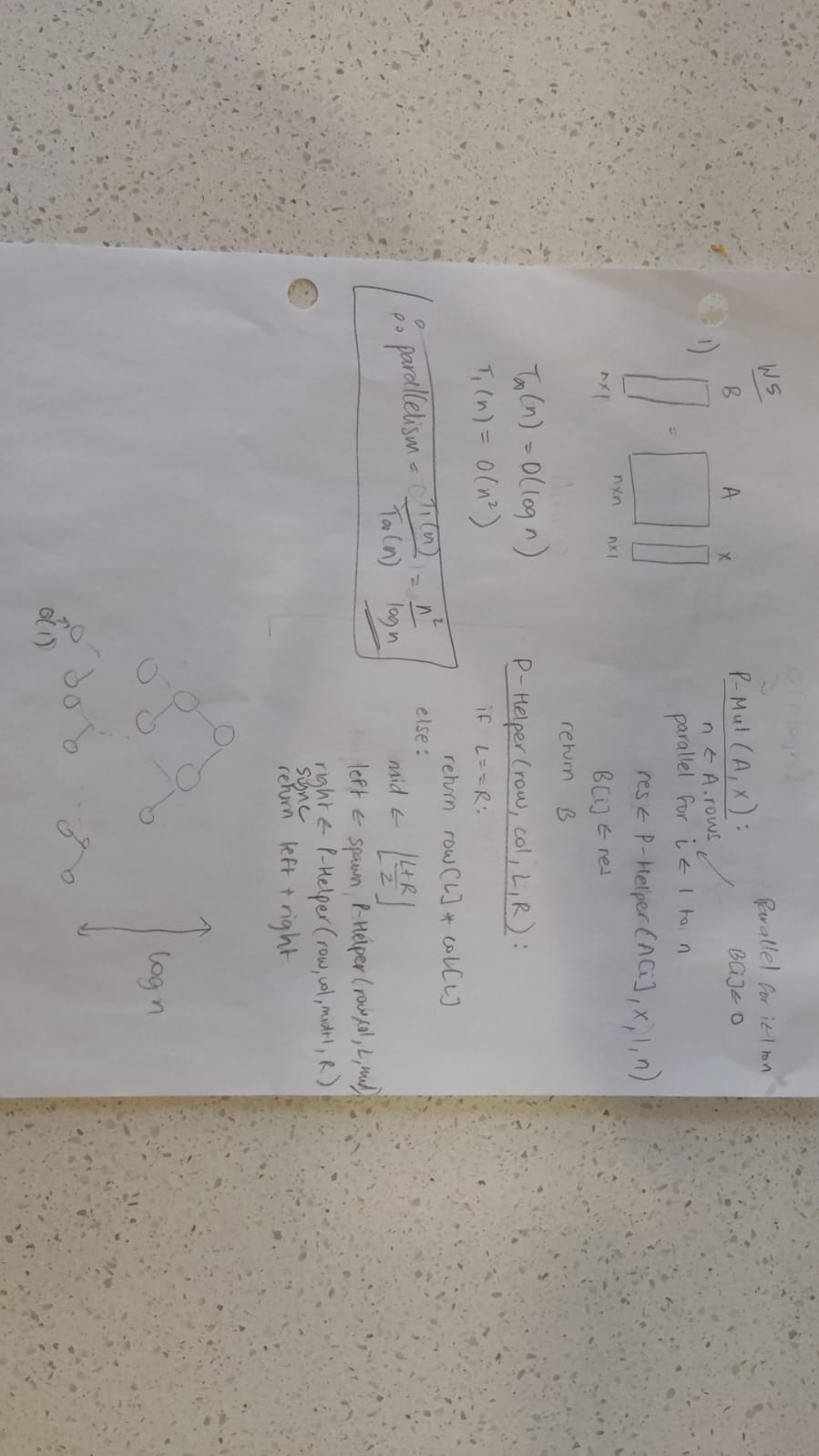
CS 255: Week 5 Exercises

1. Give a multithreaded algorithm to multiply an *n*×*n* matrix by an *n* dimensional vector that achieves Θ(*n*2*/*log*n*) parallelism while maintaining Θ(*n*2) work.

Sol:



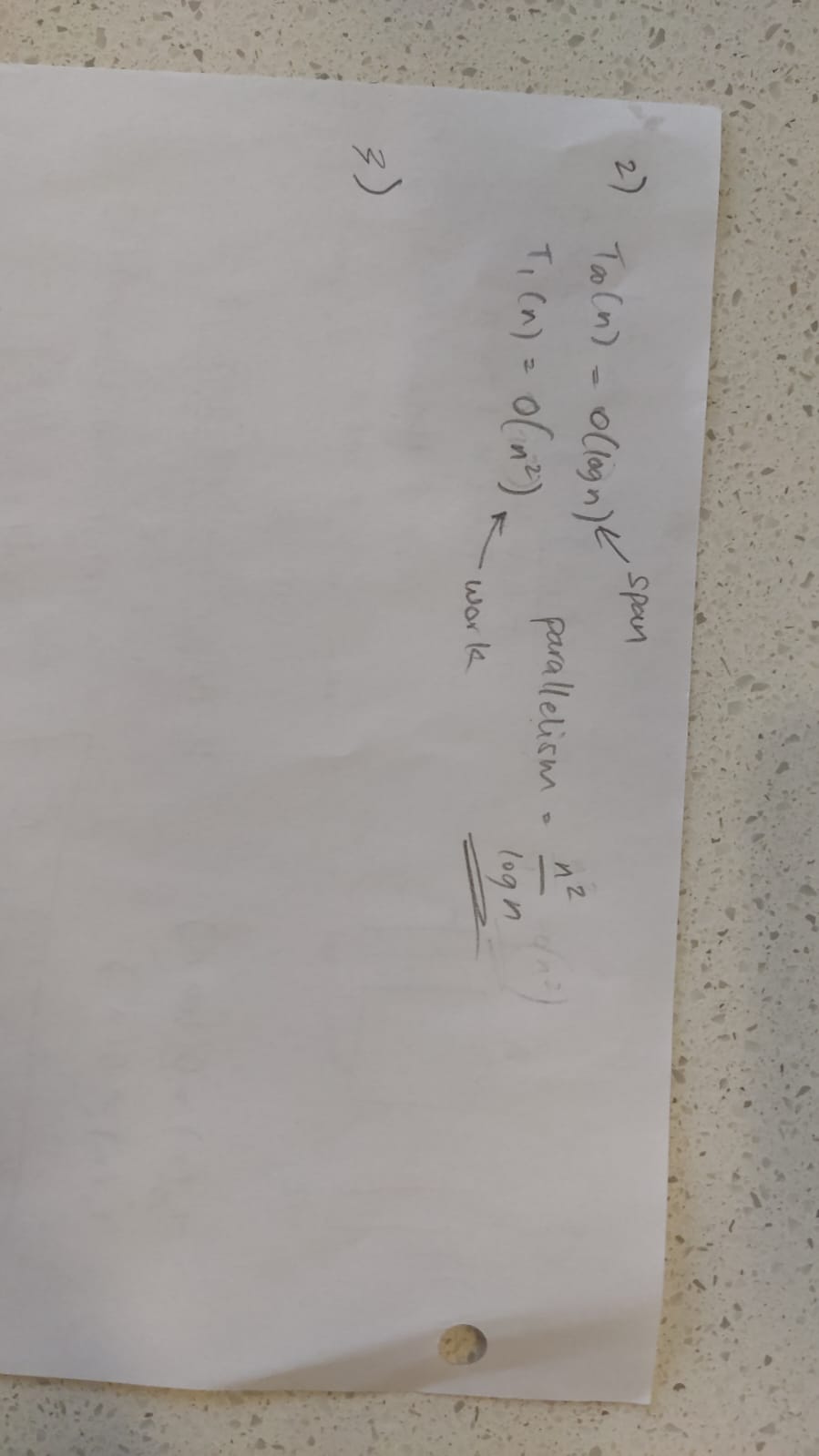
2. Consider the following multithreaded pseudocode for transposing an *n* × *n* matrix *A* in place:

**P-Transpose**(*A*): *n* ← *A*.rows parallel for *j* ← 2 to *n*

parallel for *i* ← 1 to *j* − 1 exchange *aij* with *aji*

Analyze the work, span, and parallelism of this algorithm.

Sol:



3. This problem explores how to adapt the various performance measures in order to handle randomized multithreaded algorithms.

(a) Consider a randomized multithreaded algorithm for which 1% of the time we have

*T*1 = 104 and *T*10*,*000 = 1, but for 99% of the time we have *T*1 = *T*10*,*000 = 109.

Argue that the speedup of a randomized multithreaded algorithm should be defined as *E*[*T*1]*/E*[*TP* ] rather than *E*[*T*1*/TP* ].

Sol: if we use E[T1]/E[Tp] for 1% time we get 10^4/1 = 10^4 as speedup for 1%

And 10^9/10^9 = 1 speedup for 99%

If we use E[T1/Tp] then using linearity of expectation we will have 0.01\* 10^4 + 0.99\* 1 = 100.99 which gives a wrong estimate of actual speedup. Proves E[T1]/E[Tp] givs better understanding of speedup rather than E[T1/Tp]

(b) Argue that the parallelism of a randomized multithreaded algorithm should be defined as the ratio *E*[*T*1]*/E*[*T*∞]

Sol: same like above E[T1]/E[Tinf] shows in a ideal paralleized env how will increasing number of processors increase the parallelism and it depends on the code of how the level of parallelism is present.