Exercise: Packing and Paging

1. **NextFit algorithm.** Recall that we introduced the FirstFit algorithm for the BinPacking problem during the lecture. Now consider another algorithm **NextFit**:

   > `NextFit` Algorithm
   >
   > When a new item arrives, put it into the *last* bin if the item fits into that bin.
   > Otherwise, close the last bin, open a new bin, and put this new item to this new bin.

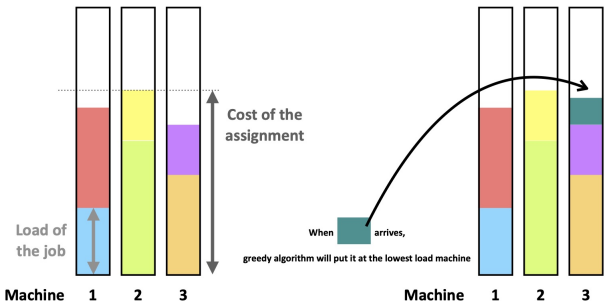
   (a) Claim that in the NextFit solution, the total size of jobs in any two consecutive bins is at least 1.

   (b) Prove that NextFit $\leq 2 \cdot \text{OPT}$.

2. **Online load balancing.** There is a finite set of m machines. A sequence of n jobs is arriving where each job J_i is specified by its processing load ℓ_i. Each job must be assigned to exactly one of the machines upon arrival. The total load of a machine is the sum of loads of the jobs assigned at it. The cost of an assignment is the maximum total load among the machines. The goal of the LoadBalancing problem is to find an assignment with the minimum cost.

   Consider the greedy algorithm (also see the illustration):

   `Assign each arriving job` r_i `to the machine with the lowest load (breaking ties arbitrarily).`



   (a) Consider any job J_k. Claim that the cost of the optimal assignment is at least ℓ_k.

   (b) Claim that the cost of the optimal assignment is at least $\frac{\sum_{j=1}^{n} \ell_j}{m}$
   (Hint: Use the similar argument for bin packing optimal cost lower bound.)

   (c) Show that the greedy algorithm is $2 - \frac{1}{m}$-competitive.

3. Consider the algorithm `LIFO` (Last-In-First-Out) for the PAGING problem:

> **`LIFO` (Last-In-First-Out) Algorithm**
>
> When there is a page fault and the cache is full, replace the page that has been copied into the cache the most recently.

Answer the following questions:

(a) Given a sequence of n page requests, show that `LIFO` is at most $\Omega(\frac{n}{k})$-competitive.

(b) Show that the analysis in (a) is tight.

4. Consider the algorithm `CLOCK` (CLOCK-replacement) for the PAGING problem:

> **`LIFO` (Last-In-First-Out) Algorithm**
>
> For every page in the cache, there is a binary variable *mark* attached to it. Initially, when a page is copied into the cache, its *mark* bit is set to 0. Once a page that in the cache is accessed, its *mark* bit is set to 1
>
> When there is a page fault and the cache is full, replace the first page that has its *mark* bit value equal to 0.

Given the phase partitioning, answer the following questions:

(a) Show that by the phase partitioning, the `CLOCK` algorithm incurs at most k page faults, where k is the cache size.

(b) Show that `CLOCK` is an optimal online algorithm for the PAGING Problem