Exercise 1: Randomized Paging  \hspace{1cm} (10 Points)

Consider the following simple randomized algorithm `rand` for the online paging problem:

If a page fault occurs, choose the page to be evicted uniformly at random.

We want to show that `rand` is \( k \)-competitive against an adaptive adversary.

Let `OPT` be an optimal offline algorithm. Let \( d_i \) be the number of pages in `rand`'s cache (fast memory) that are not in `OPT`'s cache, at step \( i \). Define a potential function \( \Phi(i) = k \cdot d_i \). Let \( a_i \) be amortized cost (with respect to \( \Phi \)) of the \( i \)-th request for `rand` and let \( c_i \) be the cost of the \( i \)-th request for `OPT`. Let \( p \) be the page requested in the \( i \)-th step.

Show that \( E[a_i] \leq k \cdot c_i \) if

a) \( p \) is in `rand`'s cache. \hspace{1cm} (1 Point)

b) \( p \) is not in `rand`'s cache, but it is in `OPT`'s cache. \hspace{1cm} (2 Points)

c) \( p \) is neither in `rand`'s cache, nor in `OPT`'s cache, and `OPT` evicts an unshared page. \hspace{1cm} (2 Points)

d) \( p \) is neither in `rand`'s cache, nor in `OPT`'s cache, and `OPT` evicts a shared page. \hspace{1cm} (3 Points)

Conclude that `rand` is \( k \)-competitive against an adaptive adversary. \hspace{1cm} (2 Points)

Exercise 2: Maximum Cut \hspace{1cm} (10 Points)

Let \( G = (V,E) \) be an unweighted undirected graph. A maximum cut of \( G \) is a cut whose size is at least the size of any other cut in \( G \).

(a) Give a simple randomized algorithm that returns a cut of size at least \( 1/2 \) times the size of a maximum cut in expectation and prove this property. \hspace{1cm} (2 Points)

(b) Prove that the following deterministic algorithm (Algorithm 1) returns a cut of size at least \( 1/2 \) times the size of a maximum cut. \hspace{1cm} (3 Points)

Algorithm 1 Deterministic Approximate Maximum Cut

Pick arbitrary nodes \( v_1, v_2 \in V \)
\[
A \leftarrow \{v_1\}
B \leftarrow \{v_2\}
\]

for \( v \in V \setminus \{v_1, v_2\} \) do

if \( \deg_A(v) > \deg_B(v) \) then

\( B \leftarrow B \cup \{v\} \)

else

\( A \leftarrow A \cup \{v\} \)

Output \( A \) and \( B \)

\( \deg_X(v) \) is the number of \( v \)'s neighbors in \( X \subseteq V \).
Let us now consider an online version of the maximum cut problem, where the nodes $V$ of a graph $G = (V, E)$ arrive in an online fashion. The algorithm should partition the nodes $V$ into two sets $A$ and $B$ such that the cut induced by this partition is as large as possible. Whenever a new node $v \in V$ arrives together with the edges to the already present nodes, an online algorithm has to assign $v$ to either $A$ or $B$. Based on the above deterministic algorithm (Alg. 1), describe a deterministic online maximum cut algorithm with strict competitive ratio at least $1/2$. You can use the fact that Algorithm 1 computes a cut of size at least half the size of a maximum cut.

**Hint:** An online algorithm for a maximization problem is said to have strict competitive ratio $\alpha$ if it guarantees that $\text{ALG} \geq \alpha \cdot \text{OPT}$, where $\text{ALG}$ and $\text{OPT}$ are the solutions of the online algorithm and of an optimal offline algorithm, respectively. (2 Points)

(d) Show that no deterministic online algorithm for the online maximum cut problem can have a strict competitive ratio that is better than $1/2$. (3 Points)