University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithms Theory

Sample Solution Exercise Sheet 13 - Bonus

Due: Tuesday, 16th of February 2021, 4 pm

Exercise 1: Randomized Paging *(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 the amortized cost (with respect to Φ) 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. *(1 Point)*

b) p is not in `rand`'s cache, but it is in `OPT`'s cache. *(2 Points)*

c) p is neither in `rand`'s cache, nor in `OPT`'s cache, and `OPT` evicts an unshared page. *(2 Points)*

d) p is neither in `rand`'s cache, nor in `OPT`'s cache, and `OPT` evicts a shared page. *(3 Points)*

Conclude that `rand` is k-competitive against an adaptive adversary. *(2 Points)*

Sample Solution

a) The actual cost of `rand` is 0. If `OPT` does not evict a page or if it evicts a shared page, the potential does not change. If `OPT` evicts an unshared page, d_i decreases by 1 and hence the potential decreases by k. In all cases we have $a_i \leq 0$ and hence it is $E[a_i] \leq 0$.

b) The actual cost of `rand` is 1. We have $\phi(i) - \phi(i-1) = 0$ if `rand` evicts a shared page and $\phi(i) - \phi(i-1) = -k$ otherwise. As the evicted page is chosen uniformly at random, we have

$$E[\phi(i) - \phi(i-1)] = \frac{d_i}{k}(-k) = -d_i \leq -1 \ .$$

The last inequality holds because p was in `OPT`'s cache, but not in `rand`'s cache. It follows $E[a_i] = 1 + E[\phi(i) - \phi(i-1)] = 0$.

c) The actual cost of both `rand` and `OPT` is 1. If `rand` evicts a shared page, the potential does not change. Otherwise, d_i decreases by 1 and hence the potential decreases by k. In both cases we have $a_i \leq 1$ and hence it is $E[a_i] \leq 1 \leq k = k \cdot c_i$.

d) The actual cost of both `rand` and `OPT` is 1. We have $\phi(i) - \phi(i-1) = 0$ if `rand` evicts the same page as `OPT` or an unshared page and $\phi(i) - \phi(i-1) = k$ otherwise. The probability of the last event is $\frac{k-d_i-1}{k}$. Thus, we obtain

$$E[\phi(i) - \phi(i-1)] = \frac{k - d_i - 1}{k} \cdot k = k - d_i - 1 \leq k - 1 = k \cdot c_i - 1 \ .$$

and hence $E[a_i] = 1 + E[\phi(i) - \phi(i-1)] \leq k \cdot c_i$.

We have shown that in any case, $E[a_i] \leq k \cdot c_i$. Let $c_i^{\texttt{rand}}$ be the actual cost of the i-th request for `rand`. For any sequence I of requests with $|I| = n$ we have

$$E[\texttt{rand}(I)] = E[\sum_{i=1}^{n} c_i^{\texttt{rand}}] \leq E[\sum_{i=1}^{n} a_i] = \sum_{i=1}^{n} E[a_i] \leq \sum_{i=1}^{n} k \cdot c_i = k \cdot \texttt{OPT}(I) \ .$$

Exercise 2: Maximum Cut *(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. *(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. *(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** ▷ $deg_X(v)$ is the number of v's neighbors in $X \subseteq V$.
    $B \leftarrow B \cup \{v\}$
  else
    $A \leftarrow A \cup \{v\}$
Output A and B

(c) 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 α if it guarantees that $\text{ALG} \geq \alpha \cdot \text{OPT}$*, where* ALG *and* 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)*

Sample Solution

(a) Initialize $S = \emptyset$. Each node joins S independently with probability $1/2$. For an edge $e = \{u, v\}$, the probability that e is between S and $V \setminus S$ is $\Pr(u \in S \wedge v \notin S) + \Pr(u \notin S \wedge v \in S) = 1/2$. So in expectation, half of the edges are between S and $V \setminus S$.

(b) We distinguish between *crossing edges* with one endpoint in A and one in B and *inner edges* with either both endpoints in A or both in B. We show that the following property is a loop-invariant: The number of crossing edges is at least the number of inner edges.

This is true before entering the loop the first time (there are no inner edges). In the following iterations, a node is added to B iff it has more neighbors in A than it has in B and it is added to A iff it has at least as many neighbors in B as in A. So in either case, the number of crossing edges that are added is at least the number of inner edges that are added.

When all nodes are processed, the number of crossing edges is at least the number of inner edges, i.e., at least half of the edges are crossing edges. So the resulting cut has size at least $|E|/2$ and $|E|$ is an upper bound for a maximum cut.

(c) Algorithm 1 actually describes an online algorithm. The first node that arrives is assigned to A, the second node to B and all other nodes are processed as described in the loop. As shown we obtain ALG$\geq |E|/2$ and OPT$\leq |E|$ and hence the competitive ration is at least $1/2$.

(d) Consider a graph with nodes $v_1, v_2, \ldots, v_n$. Nodes v_1 and v_2 are adjacent to each other node, more edges do not exist. The cut $(\{v_1, v_2\}, \{v_3, \ldots, v_n\})$ has size $2(n-2) \leq$OPT. Assume ALG is an online algorithm with competitive ratio $r > 1/2$ which computes a cut $(S, V \setminus S)$ for any graph (V, E). If ALG first receives v_1 and v_2, it must put one in S and the other in $V \setminus S$, because otherwise we had ALG$= 0$ and OPT$= 1$ in case the graph consists only of v_1 and v_2 (which an online algorithm can not know at this point). Any cut with $v_1 \in S$ and $v_2 \notin S$ (or vice versa) has size $n-1$. Therefore we have ALG/OPT$\leq \frac{n-1}{2(n-2)} \overset{n\to\infty}{\longrightarrow} 1/2$. So for n sufficiently large we have ALG/OPT$< r$, contradicting that ALG has competitive ratio r.