Advanced Algorithms: Exam 1 Review

Dana Randall Spring 2024

Sarthak Mohanty

Exercise 1

A three-way cut is a set of edges such that removing it from the graph yields three connected components. Modify the min-cut algorithm discussed in class to get a three-way min cut. Compute the probability of success of your design and discuss how to boost it to be 1 - $o(1)$.

Hint: Follow the process from class, taking special care near the end.

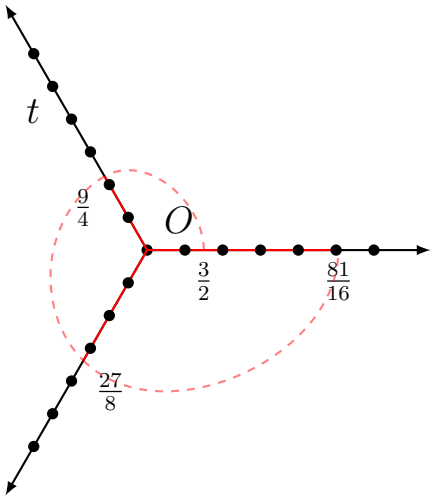
Exercise 2

(Navigating The Labyrinth) The young maiden Ariadne is at the entrance of a simple labyrinth which branches in $m \geq 2$ corridors, as shown below. Unfortunately, she does not have her magic string with her! She knows the architect of the labyrinth, Daedalus, is somewhere in the labyrinth. She devises the following strategy to find him[1]:

Continue until we reach t: execute cycles of steps, where the function determining the number of steps to walk before the ith turn starting from the origin is

$$f(i) = \left(\frac{m}{m-1}\right)^i \quad \forall i \geq 1$$

What is the competitive ratio of this algorithm?



[1]This strategy is known as *spiral search* and has been proven to be optimal. Looks like Ariadne was a Theory thread!

Exercise 3

(Potential Functions) We discussed in class how potential functions were common motifs to help analyze online algorithms.

(a) For an online algorithm A, let $c_A(\sigma)$ be the cost of the algorithm on the entire sequence and $c_A(i)$ be the cost on request i (and similarly for OPT, the optimal offline algorithm). Prove the following. If for every request sequence σ there exists a potential function Φ such that for all $1 \leq i \leq n$ we have

$$c_A(i) + \Phi(i) - \Phi(i-1) \leq c \cdot c_{\text{OPT}}(i),$$

then A is c-competitive.

(b) Consider the online problem Read Into Buffer: The task is to read an unknown-length non-empty string $s \in \Sigma^+$ of symbols into a buffer stored as contiguous array, one symbol at a time. The buffer must never be larger than $2|s|$ symbols. Reading a symbol has a basic cost of 1. When the currently allocated buffer is full, a new, larger array has to be allocated and the old contents must be copied, incurring a cost of 1 for each symbol already read. In other words, the cost for the kth symbol is either 1 (not full) or k (full).

Devise an online algorithm for this problem and use a potential function to prove that its competitive ratio is at most 3.

Exercise 4

In the context of the stable marriage problem, a loop is a sequence of distinct participants $(X_1, Y_1, X_2, Y_2, \ldots, X_k, Y_k, X_1)$ with the following properties:

1. Each participant X_i prefers Y_i over Y_{i+1}, with the exception that X_k prefers Y_k over Y_1.
2. Each participant Y_i prefers X_{i+1} over X_i, with the exception that Y_1 prefers X_1 over X_k.

Assess the accuracy of the following statement: A preference list containing a loop leads to the existence of at least one stable marriage.

Exercise 5

Consider a scenario in the stable marriage problem involving n men and n women. Determine the *exact* maximum number of proposals that could occur when running the Gale-Shapely algorithm.

Exercise 6

Consider FlushWhenFull algorithm for paging: when a cache miss occurs and the entire cache is full, evict all pages from the cache. What is the competitive ratio of FlushWhenFull?

Exercise 7

Consider adding the power of limited lookahead to an algorithm for paging. Namely, fix a constant $f \in \mathbb{N}$. Upon receiving the current page p_i, the algorithm also learns $p_{i+1}, p_{i+2}, \ldots, p_{i+f}$. Recall that the best achievable competitive ratio for deterministic algorithms without lookahead is k. Can you improve on this bound with lookahead?

Exercise 8

(Online Coloring) We briefly covered coloring in our homework; now we will cover it in an online context. Our input sequence consists of the vertices $v_1, \ldots, v_n$ of an undirected graph $G = (\{v_1, \ldots, v_n\}, E)$. Together with each vertex v_i, we are given the list E_i of all edges connecting v_i to previously given vertices $v_1, \ldots, v_{i-1}$. Edges from v_i to vertices v_j with $j > i$ are only revealed once vertex v_j is given. The number of edges and vertices in the graph is not known in advance.

When we are given v_i, we have to choose a color $c(v_i) \in \mathbb{N}$ for v_i in such a way that no vertex adjacent to v_i has color $c(v_i)$. As usual, this choice is final; we cannot change the color later on. We want to minimize the number of colors used. For an example, see Figure 1.

We want to show that there is no deterministic c-competitive algorithm for this

problem for any constant c. For any constant number $2 \leq k \in \mathbb{N}$ of colors and any deterministic online algorithm A, devise a strategy for an adversary that satisfies the following requirements.

- The strategy always produces a forest $T_{A,k}$.
- The online algorithm A uses at least k colors on $T_{A,k}$.

Offline, every forest can be colored with 2 colors; therefore, this implies that A's competitive ratio is at least $k/2$.

Exercise 9[2]

(Online Load Balancing) We discussed load balancing quite a bit in the beginning of the semester. Here we consider a weighted variant. Suppose we receive a sequence of items, where $a_i \in (0, \infty)$ Moreover, we have a fixed number of m bins with unlimited capacity. We have to pack each incoming item into a bin that we have to fix before the next item is given to us. As in load balancing, we want to minimize the total weight W packed into the fullest bin.

We consider the following algorithm GREEDY for this problem. Initially, all bins are empty. On receiving the ith item a_i, GREEDY packs a_i into the emptiest bin; ties are broken arbitrarily. For an example of the algorithm, refer to Figure 1.

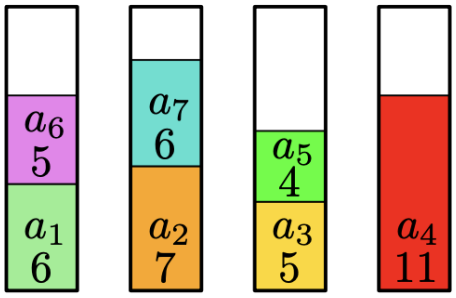


Figure 1: The packing that results from applying GREEDY to $m = 4$ bins and the input sequence $(6, 7, 5, 11, 4, 5, 6)$. The quality of the solution produced by GREEDY is $W_G = 13$ (the fullest bin contains $7 + 6 = 13$ units of weight). The optimal solution has quality $W_O = 11$.

[2]From Technische Universität Braunschweig.

(1) Prove that GREEDY cannot be better than $(2 - \frac{1}{m})$-competitive. *Hint: Consider a sequence that begins with small items and ends with a big item.*

(2) Prove that GREEDY is $(2 - \frac{1}{m})$-competitive. *Hint: Consider the last item a_T placed into the fullest bin and the value $W_G - a_T$.*

Exercise 10[3]

(*k-wise Independent Hashing*) We can extend our definition of k-wise independence to hashing as well. A hash family $\mathcal{H}$ is *k-wise independent* if for all k distinct keys $x_1, x_2, \ldots, x_k \in \mathcal{U}$ and every set of k values $v_1, v_2, \ldots, v_k \in \{0, 1, \ldots, m-1\}$, we have that

$$\mathrm{P}_{h \in \mathcal{H}} \left(h(x_1) = v_1 \wedge h(x_2) = v_2 \wedge \ldots \wedge h(x_k) = v_k \right) = \frac{1}{m^k}$$

Intuitively, this means that if you look at only up to k keys, the hash family appears to hash them truly randomly.

(a) Is this hash family from $U = \{a, b\}$ to $\{0, 1\}$ (i.e., $m = 2$) one-wise independent (i.e., uniform)? How about pairwise independent?

	a	b
h_1	0	0
h_2	1	0

(b) Can you fill in the blanks in this hash family with values in $\{0, 1\}$ to make it pairwise independent? 3-wise independent?

	a	b	c
h_1	0	0	
h_2			1
h_3	0		
h_4	1	1	0

[3]Taken from CMU's 15-451/651 recitation.

Exercise 11

Here we consider a classic 2-universal hash function. Consider a universe where the keys are k bits long, so $U = 0, \ldots, 2^k - 1$. For a matrix $\boldsymbol{A} \in \{0,1\}^{k \times n}$, define the function $h_A(x) = \boldsymbol{A}\boldsymbol{x}$, where we do addition mod 2. Now consider the family of functions $\mathcal{H} = \{h_{\boldsymbol{A}} : \boldsymbol{A} \in \{0,1\}^{k \times n}\}$

(a) Prove this function is 2-universal.

(b) How many random bits does it take to sample a random hash function from this family?

(c) *(Optional)* What is the running time of h? How does it compare to, say, a binary search tree?

Exercise 12

(Randall) Construct an instance of the stable matching problem with 4 men and 4 women and a stable matching in which every man is matched to his least preferred woman.

Exercise 13

(Randall) Prove that in the output of the Traditional Dating Algorithm (when men propose) at most one man gets his last choice.

Exercise 14

(Randall) Recall that the woman optimal matching is the output of the TDA algorithm when women propose to men. Prove that an instance of the stable matching problem has exactly one stable matching if and only if the man optimal matching is equal to the woman optimal matching.

Extra Practice

These are a collection of problems online I found interesting. I haven't drafted up solutions for these, and their relevance to the actual exam is undetermined.

Exercise 15

Instead of searching for treasure on a line, consider the problem of searching for treasure on a 2-dimensional grid. The robot begins at the origin of $\mathbb{Z}^2$ and the treasure is located at some coordinate $(x, y) \in \mathbb{Z}^2$. The measure of distance is given by the Manhattan metric $\|(x, y)\| = |x| + |y|$. The robot has a compass and at each step can move north, south, east, or west one block. Design an algorithm for the robot to find the treasure on the grid. What is the competitive ratio of your algorithm? Can you improve the competitive ratio with another algorithm?

Exercise 16

Consider the setting of the ski rental problem with rental cost 1 and buying cost b, $b \in \mathbb{N}$. Instead of an adversary choosing a day $n \in \mathbb{N}$ when the weather is spoiled, this day is generated at random from distribution p. Design an optimal deterministic online algorithm for the following distributions p:

(a) uniform distribution on $[n]$.

(b) geometric distribution on $\mathbb{N}$ with parameter $1/2$.

Exercise 17

We are given a sequence of items of unknown weights $a_1, \ldots, a_n \in [0, 1]$ and want to assign these items to bins in an online fashion; however, the bins do not have limited capacity. In the Bin Covering problem, we want to *maximize* the number of *covered bins*, i.e., the number of bins that receive items of total weight at least 1.

Find an online algorithm for BIN COVERING with an absolute[4] competitive ratio of 2 and prove its competitive ratio. Prove that no deterministic online algorithm can have an absolute competitive ratio $c < 2$.

Exercise 18[5]

In this exercise, we consider the LIST UPDATE Problem. Suppose that we are storing a set $S = s_1, ..., s_n$ of values as unsorted linked list. Our input sequence consists of queries $s_i \in S$ that ask us to find a certain element in our list. In order to find an element, we iterate through the list starting from the front and return the element once we find it. Because iterating through our list takes time, we incur a cost of 1 for each element that we touch during the search. For example, searching for 2 in the list [1, 2, 3] costs two units, and searching for 2 in [2, 1, 3] costs just one unit.

After each query for an item s_i, we are allowed to move the item s_i to any point closer to the front of the list; this exchange does not cost us anything. For example, if we search for 2 in the list [1, 4, 2, 3], we can change the list to [2, 1, 4, 3], [1, 2, 4, 3] or [1, 4, 2, 3] after finding 2 for a total cost of three units.

Our goal is to devise an online algorithm for this problem; after each request, the algorithm has to decide where to move the requested item. Intuitively speaking, we want our algorithm to maintain its list such that frequently requested elements are closer to the front than less frequently requested elements.

(a) The algorithm FREQUENCYCOUNT maintains a frequency count for each element that is incremented each time the element is requested. After each request, it moves the requested item such that the list is in nonincreasing order of frequency count. Prove that FREQUENCYCOUNT is not c-competitive for any constant c.

(b) Now consider the MOVETOFRONT algorithm for the List Update Problem. After each request s_i, the algorithm moves the requested item s_i to the front of the list. For example, after searching for 2 in [1, 4, 2, 3], the

[4]We may have briefly mentioned this in class, but here “absolute" just means replace the max in the definition with a sup

[5]Taken from here.

list becomes [2, 1, 4, 3].

Hint: Use the number of inversions in MoveToFront's list with respect to OPT's list after request i as a potential function $\Phi(i)$. An inversion is a pair (x, y) of elements such that x comes before y in MoveToFront's list, but after y in OPT's list.

In order to prove $c_{\text{MTF}}(i) + \Phi(i) - \Phi(i-1) \leq 2c_{\text{OPT}}(i)$ for a request s_i, consider the number of items k that come before s_i in both OPT's and MoveToFront's list and the number of items ℓ that come before s_i in MoveToFront's list but after s_i in OPT's list.