*Note*: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

# Section 1: Hashing, Streaming

## 1 Open Addressing

Recall that in hashing with chaining, each table cell stores a bucket of keys. In *open addressing*, all keys are stored directly inside the table. When a collision occurs, we try another location according to some probe sequence.

Suppose we have a table of size $m = 11$, hash function

$$h(x) = x \bmod 11,$$

and linear probing rule

$$h_t(x) = (h(x) + t) \bmod 11$$

for $t = 0, 1, 2, \ldots$.

This means that the key $x$ first tries slot $h_0(x) = h(x)$. If that slot does not answer the question, the algorithm tries $h_1(x)$, then $h_2(x)$, and so on. The sequence

$$h_0(x), h_1(x), h_2(x), \ldots$$

is called the *probe sequence* for $x$.

For this problem, assume there are no deletions. The operations work as follows:

- `put(x)` follows the probe sequence for $x$ until it finds an empty slot, then stores $x$ there. If it sees $x$ along the way, it can stop because $x$ is already present.
- `find(x)` follows the probe sequence for $x$ until either it finds $x$, in which case it returns `True`, or it reaches an empty slot, in which case it returns `False`. Reaching an empty slot proves $x$ is not present, because `put(x)` would have inserted $x$ there before probing any later slots.

The current table is:

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | | 12 | 23 | | 15 | 5 | | 18 | 29 | | 21 |

(a) Draw the probe sequence used to insert 34, and show the updated table.

(b) Starting from the updated table, draw the probe sequence used to search for 56. Does the search succeed or fail?

(c) The table above was built using linear probing, so nearby collisions can form clusters. To analyze the probabilistic behavior more cleanly, suppose instead that each new key has a uniformly random probe sequence over the table locations. If the load factor is $\alpha = n/m$, what is the probability that the first probe collides? Under the simplifying assumption that each probe is an independent uniformly random location, what is the expected number of probes needed to insert a new key?

(d) Explain why open addressing becomes slow as $\alpha$ approaches 1, and briefly compare this with chaining.

## 2 [Bonus] Perfect Hashing

You are given a static set $S$ of $n$ keys from a universe $[U]$. Your goal is to preprocess $S$ into a dictionary data structure that supports membership queries.

Design a randomized hashing-based data structure with the following guarantees:

- $O(n)$ expected preprocessing time,
- $O(n)$ space,
- $O(1)$ worst-case query time after preprocessing.

Your answer should describe the data structure, the preprocessing algorithm, the query algorithm, and why the stated time and space guarantees hold.

 

# 3 Streaming Algebra

You are given a continuous data stream of $N$ integers, $A = (a_1, a_2, \ldots, a_N)$. You do not know $N$ in advance, but you are guaranteed that $N \geq 4$. You may read the stream exactly once, and you may not store the array.

Your goal is to compute the maximum possible value of

$$a_i - 2a_j + 3a_k - 4a_l$$

over all choices of indices satisfying

$$1 \leq i < j < k < l \leq N.$$

Assume you are restricted to $O(\log S)$ bits of memory, where $S$ is the maximum possible absolute value of the final expression.

Design a deterministic streaming algorithm for this problem. Your answer should state the variables your algorithm maintains, how those variables are updated when a new stream element arrives, why the algorithm is correct, and why it uses only $O(\log S)$ bits of memory.

 

# Section 2: Graphs, Greedy Algorithms

## 4 Remove Covered Intervals

Given an array of intervals, remove all intervals that are covered by another interval in the list.

The interval $[a, b]$ is covered by the interval $[c, d]$ if and only if $c \leq a$ and $b \leq d$.

Find the number of remaining intervals.

Example: $[[1, 5], [5, 20], [3, 12], [4, 12]]$ In this example, the interval $[3, 12]$ fully covers the interval $[4, 12]$. Hence, the remaining intervals would be $[[1, 5], [5, 20], [3, 12]]$.

 

# 5 Money Madness

Anirudh is on vacation but forgot to convert his US dollars (USD) to Indian Rupees (INR)! He visits a currency exchange booth at the airport. The booth provides an exchange rate to convert between various different currencies. Unfortunately, the exchange rate provided is never better than the actual conversion rate (you will never make money by exchanging).

(a) Given a list of exchange rates, provide an algorithm to find the maximum amount of INR he can get for 1 USD.

(b) Now, assume the booth charges a percentage fee (varies depending on currency) for each exchange. For example, if 1 USD is worth 100 INR, the currency exchange might only give you 90 INR per dollar. We represent this as an exchange rate of 0.9, because you retain 90% of the value. Once again, all exchange rates are in the range $(0, 1)$ exclusive (so you cannot profit).

Design an algorithm that finds the best sequence of exchanges for converting USD to INR (using any number of exchanges), retaining as much value as possible.

 

# Section 3: Linear Programming and Zero-Sum Games

## 6 Review Planning

You have $T$ hours left to study for an oral exam. There are $m$ review activities you can choose from, such as redoing a homework problem, reading lecture notes, going through a discussion worksheet, or explaining a proof to a friend.

There are $k$ topics on the exam. Spending one unit of activity $j$ takes $t_j$ hours and gives $a_{ij}$ units of preparation for topic $i$. You may do a fractional amount of each activity. Let $x_j \geq 0$ be the amount of activity $j$ that you do.

You expect the examiner to look for areas where you are least prepared, so you want to make your study plan balanced: your goal is to maximize your weakest topic preparation, which is given by

$$\min_{i \in [k]} \sum_{j=1}^{m} a_{ij} x_j.$$

(a) Formulate this study planning problem as a linear program.

(b) Write down the dual of your LP from part (a). Briefly describe an interpretation of the dual from the examiner's perspective.

# 7 Penalty Kicks

A kicker and a goalie are playing a zero-sum game. The kicker chooses where to shoot, the goalie chooses where to dive, and the payoff is the probability that the kicker scores. The kicker wants to maximize this probability, while the goalie wants to minimize it.

The payoff matrix is

| | Dive Left | Stay Center | Dive Right |
|---|---|---|---|
| Shoot Left | 0.2 | 0.7 | 0.8 |
| Shoot Center | 0.4 | 0.5 | 0.4 |
| Shoot Right | 0.8 | 0.7 | 0.2 |

(a) Eliminate any dominated strategies in order to reduce this $3 \times 3$ game to a $2 \times 2$ game. Justify each strategy you eliminate.

*Hint: One strategy may be dominated by a mixture of other strategies.*

(b) Solve the resulting $2 \times 2$ zero-sum game. What is the kicker's optimal mixed strategy, what is the goalie's optimal mixed strategy, and what is the value of the game?

(c) Write a linear program for the kicker's optimal mixed strategy in the original $3 \times 3$ game, and give the optimal solution to this linear program.

(d) Suppose a different kicker is better at shooting left, and has the payoff matrix

| | Dive Left | Stay Center | Dive Right |
|---|---|---|---|
| Shoot Left | 0.3 | 0.8 | 0.9 |
| Shoot Center | 0.4 | 0.5 | 0.4 |
| Shoot Right | 0.8 | 0.7 | 0.2 |

Would the solution to the zero-sum game still involve a mixed strategy, or should the kicker now always shoot left? Explain in one or two sentences.

## Section 4: NP-Completeness

## Section 5: Divide & Conquer, FFT, Parallelism, Dynamic Programming

 