Exam 1

Henry Z. Lo

June 23, 2014

In this exam, there is one problem involving hash tables, one involving dynamic programming, one divide and conquer, and one solvable by a greedy algorithm. For the dynamic programming problem, you can use either memoization or bottom-up, but be sure to mention which you are using.

For each problem, do the following in class:

1. Describe the technique / data structure used to solve the problem (10%).
2. Describe why your solution works (30%):
   - For greedy algorithms, specify what your greedy choice is, and show that it leads to an optimal solution (e.g. use an exchange argument).
   - For dynamic programming, show what the computation tree looks like, and describe how your algorithm relates to it.
   - For divide and conquer, describe how you divide, until when, what the conquer step involves, and why it is faster than other approaches.
3. Provide pseudocode (50%).
4. Give the runtime for your algorithm, and explain (10%).

The most important things are that your algorithm is correct and that you have an optimal runtime. You should give convincing arguments about how and why your algorithms work, so that I can understand your train of thought.

For the homework, write a class called `ExamSolutions`. For each problem, you will write a static function to solve it. You can write helper functions as needed. See the `ExamSolutionsTest` test case on the course website.

Problem 1. *You have been hired to engineer a pop song. You are given a set of song segments, each of which has a length s_i and a popularity p_i. Find the highest possible popularity for a song.*

- *You can construct a song by selecting song segments.*
- *Each song segment can only be used once.*
- *The song cannot be longer than s seconds.*

For example, assume a song segment has the structure (s_i, p_i). Then given $s = 200$, and the following song segments:

$$(120, 6), (80, 4), (120, 8), (60, 3)$$

The optimal solution is $(80, 4), (120, 8)$, which gives a total time of 200 seconds, and a total popularity of 12.

Write a static `solveSong` function which takes in three arguments:

- Integer array, representing song lengths.
- Integer array, representing song popularities.
- Integer, representing maximum song length.

You can assume the two arrays have the same length. The function should return the maximum possible popularity as an `int`.

Problem 2. *Given a string, count the number of times a appears before b.*

For example:

```
aThenB("aabb") == 4
aThenB("abcab") == 3
```

Write a static `aThenB` function which takes in a string argument, and returns the number of times an a occurs before a b as an `int`.

Problem 3. *John Henry only dates people whose last names are also first names. Given a list of people's names, pick out a list of possible dates for John Henry. Assume that the only first names are those in this list.*

For example, if your list consists of `["Agatha Christie","Christie Ann"]`, then the only possible candidate is `"Agatha Christie"`. Even though Ann is a first name, it is not a first name that appears in this list, so we can disregard it.

Write a static `potentialDates` function which takes in an array of strings. Each string consists of a first and last name, separated by a space. Return a subset of this array which contains names whose last names are also first names.

Problem 4. *You are managing a sales force of n people. You need to assign them to m cities, each of which has an expected profit of c_i. Each person assigned to a city reduces the expected profit by half (rounded down) for the next person. Find the maximum possible profit.*

As an example, suppose the expected profit for Montreal is $1000:

- If you assign one person to Montreal, then the total profit is $1000.
- The profit for a second person in Montreal is $500, because someone is already there. This brings the total profit to $1500.
- Likewise, a third person in Montreal will only net $250, bringing the total profit to $1750.

The complete problem consists of an integer array $c_1, \ldots, c_m$, and a salesforce size n. As an example, let the cities be `[6000,2400,3600]`, and $n = 5$. Then the answer is $\$16,800$.

Write a static `salesForce` function which takes in two arguments:

- Integer array, representing expected profit for various cities.
- Integer, representing the number people in your sales force.

The function should return the maximum possible profit as an `int`.