CS302 - Problem Set 7

If you'd like a refresher on sample spaces, and calculating the probability of a sequence of random events, see this text, pages 766-768, Steps 1-3.

Hints and solutions are on the last page.

This looks like a lot of questions, but it is because I am giving you a step-by-step guide (at least for the small arrays).

1. Suppose you are searching an array A of length n for an element with value t. You may assume A has no repeated elements. The strategy *sampling without replacement* works as follows: Let $T = \{1, 2, \ldots, n\}$. Choose ("sample") an element $i \in T$ at random, and check if $A[i] = t$. If it is, return i. If it is not, remove i from T (i.e. set T to equal $T - \{i\}$). Repeat by choosing an element of the updated set T at random. Repeat until you sample an index i such that $A[i] = t$. The strategy *sampling with replacement* is similar except there is no update to T, so throughout the algorithm, T is always equal to $\{1, 2, \ldots, n\}$. (Replacement refers to whether the guessed index is placed back ("replaced") into the set T or not after it is guessed.)

   For problems (a)-(l) you should assume that $n = 3$ and $A[1] = t$ (the index of the element of interest is 1.) Problems (a)-(h) deal with sampling without replacement, and problems (i)-(l) deal with sampling with replacement.

   (a) What is the sample space S for sampling without replacement? (Please list all elements of the sample space.)

   (b) Let $p : S \to \mathbb{R}$ be the function that gives the probability of each element of S occurring. List $p(s)$ for each $s \in S$. (See reference above if you are having difficulty.)

   (c) Let $R : S \to \mathbb{R}$ be the function that gives the number of rounds that occur for each element of the sample space. (If you make g guesses before finding the item you are looking for, the number of rounds is g.) What is R for each element of S?

   (d) Using your answers to the previous parts, calculate (directly, without using indicator random variables) the average number of rounds for the sampling without replacement strategy for an array of length 3 if the element you are looking for is in the 1st position.

   (e) How would your answer to (d) change if the element you were looking for were not in the 1st position?

   (f) Challenge (Wait to turn page until attempted for challenge): Write R as a weighted sum of indicator random variables.

   Or: turn to next page.

Consider the indicator random variables $X_r : S \to \{0, 1\}$ where

$$X_r(s) = \begin{cases} 1 & \text{if } s \text{ has at least } r \text{ rounds} \\ 0 & \text{if } s \text{ less than } r \text{ rounds.} \end{cases} \tag{1}$$

We can write R as a weighted sum of these indicator random variables:

$$R(s) = \sum_{r=1}^{3} \alpha_r X_r(s) \tag{2}$$

where $\alpha_r \in \mathbb{R}$. What values of $\alpha_1, \alpha_2, \alpha_3$ make Eq. (2) true?

(g) What is the probability that at least r rounds occur?

(h) Using linearity of expectation, properties of indicator random variables, and your answers to the previous two questions, recalculate $\mathbb{E}[R]$ and check that your answer is the same as before.

(i) What is the sample space S' if we use sampling with replacement to search the same length-3 array? Please describe the set in words or using mathematical notation.

(j) Let $R : S' \to \mathbb{R}$ be the function that gives the number of rounds that occur for each element of the sample space S' (sampling with replacement). Write R as a weighted sum of indicator random variables X_r (where X_r is defined above).

(k) With sampling with replacement, what is the probability that at least r rounds occur?

(l) Using linearity of expectation and properties of indicator random variables, what is the average number of rounds in sampling with replacement in an array of size 3?

(m) Using the intuition and techniques you have developed from the size 3 example, please now analyze the case of a length-n array. Determine the average number of rounds to find an item in an array of size n, with no repeated elements, where the item you are looking for is in the array, if we use

- sampling without replacement
- sampling with replacement

(n) Please comment on the advantages or disadvantages of using sampling with or without replacement.

2. Suppose you run a company with two offices, one in Washington, D.C. and the other in San Francisco, California. You always spend the whole week in one location, but each weekend, you decide whether to fly to the other office. In week i, you can make W_i dollars if you are in Washington, and C_i dollars if you are in San Francisco. Each flight from one office to the other costs \$1000. Suppose you are given the lists $W_1, W_2, \ldots, W_n$

and $C_1, C_2, \ldots, C_n$. Also suppose you are in Washington initially and you must end in Washington at the end of the nth week. What schedule will maximize your profits? (Note your solution should not just give the maximum profit, but should return a schedule.) (I realize this is an extremely outdated question, as the obvious cheapest, safest, and most environmentally friendly solution is to video conference rather than fly across the country! However, for the purposes of practice, please humor me.)

(a) A greedy algorithm would look at how much money you would make in each week, and work in the place that will earn the most (after travel costs). Give a counter example to show why this greedy algorithm fails.

(b) Create a dynamic programming algorithm (follow our usual steps:)

i. Determine options for the solution.

ii. Write a recurrence relation for the solution of one size problem in terms of the solutions to smaller sized problems for each option. (You should come up with notation for the optimal solutions for subproblems. For example, we have used S_i for MWIS, and $S_{i,R}$ for knapsack. You'll need to think about what parameters you need to define optimal subproblems.) Briefly explain why your recurrence is correct.

iii. Write pseudocode to fill in an array (might be multidimensional) with the value of the objective function. (It should take as input arrays/lists W and C that contain the amount you can make .)

iv. Write pseudocode to work backwards through the array you created to determine the optimal schedule.

(c) What is the runtime of your algorithm?

3. Suppose I would like to give you more flexibility on your exams, so I give you some large number (let's call it M) of problems, where the ith problem is worth P_i points. The time I give you to take the exam is not sufficient to solve all of the problems, so each student might solve a different subset of problems and might get different grades on each problem. I would like to give a good grade to a student who does sufficiently well on a sufficient number of questions, and give a worse grade to a student who just gets a few points correct on a lot of problems. How could I reduce this problem of determining grades to the knapsack problem?

4. Approximately how long did you spend on this assignment (round to the nearest hour)?

Hints/Answers:

1d/1h. 2
1f. $\alpha_r = 1$ for all r.
1g. The probability of at least r rounds is the probability that we don't find it in the first round and we don't find it in the second round and we don't find it in the third round...(when do we stop?) Since we have a sequence of events, we can use the same techniques (see the reference) as before for calculating the probability that the entire sequence occurs. 1l. 3
2b: Create recurrence relations for the following 2 functions: $S_{i,DC}$ be the optimal sequence of weeks up to and including week i, where in week i, we work in DC. Let $S_{i,SF}$ be the optimal sequence of weeks up to and including week i, where in week i, we work in SF.