CS 256: Algorithm Design and Analysis

Assignment 8 (due 11/19/2020)

Instructor: Sam McCauley

NP-Hardness

Problem 1. Define the $\mathsf{ODD}-\mathsf{SUBSET}-\mathsf{SUM}-\mathsf{RANGE}$ problem as follows. Given n odd numbers $s_1, \ldots s_n$, and two target integers T_1 and T_2, determine if there exists a subset of numbers that adds up to a value between T_1 and T_2.[1]

Prove that $\mathsf{ODD}-\mathsf{SUBSET}-\mathsf{SUM}-\mathsf{RANGE}$ is NP-hard.

Hint: Let's say that $s_1, \ldots s_n$ were required to be even numbers. Then how could we solve this problem? Can we extend that idea to the case of odd numbers? You'll get partial credit for solving the even case.

Solution. □

[1] "Between" is inclusive, so a subset adding up to exactly T_1 or T_2 indicates a yes instance.

Problem 2 (**Extra Credit (10 points)**)**.** Define the $\mathsf{PRIME} - \mathsf{SUBSET} - \mathsf{SUM}$ problem as follows. Given n prime numbers $s_1, \ldots s_n$, and a target integer T, determine if there exists a subset of numbers that adds up to exactly T. Prove that $\mathsf{PRIME} - \mathsf{SUBSET} - \mathsf{SUM}$ is NP-hard.

A result that may be useful is from Baker, Harman, and Pintz (2001): for some constant C_0, for any $x \geq C_0$, there is a prime in the interval $[x - x^{.525}, x]$.

Solution. □

Probability

Problem 3. Let's say we have a hash table of size n that uses hashing with chaining. Let's say the table contains n elements, each of which is stored in a random slot in the hash table (independent of all other elements).

The probability that a given chain has length k is given by

$$\binom{n}{k}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{n-k}$$

This question asks you to give two upper bounds on this formula.

(a) Show that the probability that a given chain has length 0 is at most $1/e$.

(b) Show that for $k \geq 1$, the probability that a given chain has length k is at most[2] $1/k!$.

Solution. □

[2]This bound is a little loose—in fact, this upper bound is just 1 for $k = 1$. That's OK; we're just trying to get an idea of the probability. For small k, the correct probability is closer to $1/(ek!)$.

Problem 4. Let's say that we have a hash table of size $2n$ (with random hash functions) that uses linear probing. In particular, assume that we have a hash function h that takes in an element x and maps it to a number from 0 to $2n-1$. For any hash slot $s \in \{0, \ldots, 2n-1\}$, $\Pr[h(x) = s] = 1/2n$. Assume that the output of h is fully independent: $\Pr[h(x) = s] = 1/2n$ regardless of the hash of any other elements.

We store the hash table using an array A of size $2n$. To insert an element x, we check if $A[h(x)]$ is empty; if so we store x in $A[h(x)]$. Otherwise, we move to $A[h(x)+1]$ and repeat the same process, continuing until an empty slot is found. This hash table is "circular": it loops back to the beginning if this process goes off the end of the table. So if slot $2n-1$ of the table is full, we then examine slot 0, then slot 1, etc.

Let's say we insert n items into the hash table. Afterwards, we insert a new item i. Answer these two questions about the probability of which slot i is stored in. Your answer can be an equation (it does not need to be in a simple form); however, you should explain your answer.

(a) What is the probability that i is stored in $A[h(i)]$?

(b) What is the probability that i is stored in $A[h(i)+1]$?[3]

Let's break this question down further. First, for $k \in \{1, 2, \ldots, n\}$, let P_k be the probability that $A[h(i)-k]$ and $A[h(i)+1]$ are empty, but the k consecutive slots before $h(i)+1$[4] contain an element. If we knew $P_1, P_2, \ldots P_n$, then we could return the probability that i is stored in $A[h(i)+1]$ as $P_1 + P_2 + \ldots + P_n$.

Now let's calculate P_k for each k.

Main question (full points for answering this): What is the probability that of the n items inserted into the hash table, exactly k of the items hash to the $k+1$ slots before $h(i)+1$; that is, there are k items $j_1, j_2, \ldots j_k$ such that $h(j_\ell) \in \{h(i)-k, h(i)-k+1, \ldots, h(i)\}$ for all $1 \le \ell \le k$?

Multiplying your response to the main question by[5] $\frac{1}{k+1}\left(1 - \frac{n-k}{2n-k-1}\right)$ obtains P_k. Summing over all P_k obtains the answer.

Solution. □

[3] Since A is circular, if $h(i) = 2n-1$, this is asking for the probability that i is stored in $A[0]$.

[4] Namely, $A[h(i)-k+1], A[h(i)-k+2], \ldots A[h(i)]$; all of these indices are modulo $2n$.

[5] Where did this number come from!? The answer is actually part (a)! We know that there are k items in $h(i)-k, h(i)-k+2, \ldots, h(i)$ and we want the probability that the first slot is empty. Similarity, there are $n-k$ items in the remaining slots, and we (again) want the probability that the first slot is empty. Multiplying obtains this answer.

Problem 5. We roll a standard die over and over. What is the expected number of rolls until the first pair of consecutive sixes appear? (*Hint.* The answer is not 36.)

Solution. □

Problem 6. Consider the following algorithm for generating a biased random coin. The subroutine FairCoin returns either 0 (heads) or 1 (tails) with equal probability; the random bits returned by two different calls to FairCoin are mutually independent.

```
BiasedCoin:
  if FairCoin = 0:
      return 0
  else:
      return 1 − BiasedCoin
```

(a) Prove that BiasedCoin returns 1 with probability 1/3. (*Hint.* Easier to show that it returns 0 with probability 2/3.)

(b) What is the expected number of times that BiasedCoin calls FairCoin?

Solution. □