COMPSCI330 Design and Analysis of Algorithms
Assignment 5

Due Date: Friday, Dec 2, 2016

Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Typesetting and Submission** Please submit each problem as an *individual* pdf file for the correct problem on Sakai. LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you submit within one week of the deadline you will get half credit. Any submission after that will not receive any credit.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

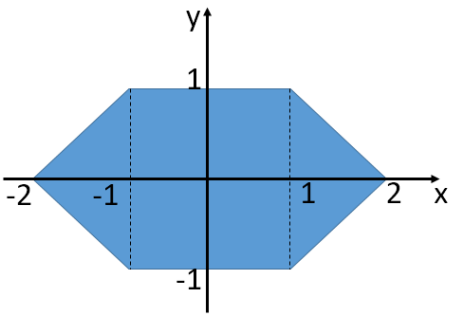

Figure 1: Feasible Region of a Linear Program

Problem 1 (Linear Programming Basics)**.** Consider a linear program with two variables x and y. The set of feasible solutions is the blue region in Figure 1.
(a) (10 points) Write out the constraints for x, y so that the feasible region is exactly equal to the blue region in Figure 1.
(b) (5 points) If the objective function is $\max x + 2y$, what is the optimal solution? If the objective function is $\max x + y$, what is the optimal solution?

Problem 2 (Linear Program Duality)**.** Consider the following linear program

$$\begin{aligned} \min \quad & x_1 - 3x_2 + 4x_3 \\ s.t. \; & -x_1 + 2x_2 - x_3 \geq 2 && (1) \\ & 2x_1 - x_2 + x_3 \geq 1 && (2) \\ & -x_2 + x_3 \geq -1 && (3) \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

(a) (5 points) Write the dual of this LP.
(b) (10 points) Show that the primal solution $x_1 = 1, x_2 = 2, x_3 = 1$ is the optimal solution by giving a dual solution with the same value.

Problem 3 (Hashing without Linked Lists)**.** (25 points) In class we talked about how to handle collisions in hashing by using a linked list for each location in your Hash table. However, in some cases you might want to make sure your hash table takes a *fixed* amount of memory. In those cases you can handle collisions in a different way. Suppose your hash table has n entries $0, 1, ..., n-1$, and your hash function f maps keys to $0, 1, ..., n-1$. In addition to f, you will also have a random "cycle" function g that maps $0, 1, ..., n-1$ to $0, 1, ..., n-1$. The property of g includes

1. For any $x \neq y \in \{0, 1, ..., n-1\}$, $g(x) \neq g(y)$.
2. For any x, let $x^1 = g(x)$, $x^2 = g(g(x))$, ..., $x^{t+1} = g(x^t)$, then $x^1, x^2, ..., x^{n-1}$ are all different from x and $x^n = x$.

Intuitively, you should think of g as defining a "cycle" between numbers 0,1,...,$n-1$. For example, if $n = 5$ one possible g is $g(0) = 3, g(3) = 2, g(2) = 4, g(4) = 1, g(1) = 0$.

The insert and find operations with a key key will first look at the location $x = f(key)$. If that location is not empty, the algorithm will continue to use function g to find the next location ($x = g(x)$) until it either finds the element or find an empty space for insertion. The implementations are given below:

```
A[0..n-1] is the hash table with (key,value) pairs

Insert(key, value)
    x = f(key)
    WHILE A[x] is not empty and A[x].key != key
       x = g(x)
    A[x].key = key
    A[x].value = value

Find(key)
    x = f(key)
    WHILE A[x] is not empty and A[x].key != key
       x = g(x)
    IF A[x] is empty THEN
        return "not found."
    ELSE
        return A[x].value
```

Suppose now the hash table contains αn elements (where α is a known fraction between $(0, 1)$), and we are inserting a random new element key such that $\Pr[f(key) = i] = 1/n$ for all $i = 0, 1, ..., n-1$. Suppose g is a uniformly random cycle. Show that the expected running time of $INSERT$ operation is $O(1/(1-\alpha))$.

(Hint: If a random variable X has value i with probability $p(1-p)^{i-1}$, then the expected value of this random variable is $1/p$.)

Problem 4 (Alternative Path)**.** (20 points) Alice is living in a big city where there is a lot of traffic. Everyday, one of the roads is highly congested. In order to avoid traffic, she decides to memorize two routes from home to work. These two routes will not share a single road, so no matter which road becomes congested, she can always get to the work using one of these routes. The map of the city is abstracted as a *directed* graph, with a source vertex s (Alice's home) and a sink vertex t (Alice's workplace). Each edge has a length (also given to Alice). You are going to use linear program to try to help Alice find two paths from s to t that do not use the same edge twice. The two paths you find should have minimum total length.

(In this problem you only need to write a LP to try to solve Alice's problem. If you use a variable that is between 0/1, you don't need to worry whether it will be an integer in the final solution.)

Problem 5 (Good Hash Function)**.** (25 points) We are going to design a hash function $f(x)$ that maps k-digit binary numbers to r-digit binary numbers (so the size of the hash table is $n = 2^r$). We would like the function to have the property we talked about in class: for any two k-digit binary numbers $x \neq y$, we want

$$\Pr[f(x) = f(y)] = 1/2^r.$$

In order to design the hash function, let $t = k/r$ (for simplicity assume t is an integer). Then we can write all k-digit binary numbers as a concatenation of t r-digit numbers $x_1, x_2, ..., x_t$ (e.g. if $k = 6$, $r = 2$, $t = 3$, then we can write 110110 as (11)(01)(10)).

Suppose we can define a "product operation" $p \otimes q$ between r-bit binary numbers. This product satisfies the usual properties of multiplication (associativity, commutativity and distributivity). Further, for any $p \neq 0$, there exist a unique p^{-1} such that $p \otimes p^{-1} = 1$.

When we are adding two r-bit binary numbers, if the result requires $r + 1$-bits we will simply drop the highest bit.

For function f, we will sample t uniformly random r-bit random numbers $u_1, u_2, ..., u_t$, and let $f(x) = \sum_{i=1}^{t} (u_i \otimes x_i)$.

Show that this random hash function satisfies the property we want: for any two k-digit binary numbers $x \neq y$, we want

$$\Pr[f(x) = f(y)] = 1/2^r.$$