Due: ~~Dec 28 10:00~~ Dec 30 17:00

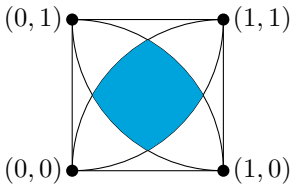
Cmpe 300: Homework 3 — Due: December 30th 17:00

Emilcan Arıcan 2016400231

The purpose of this homework is to familiarize you with algorithm analysis. Solve the following questions in LaTeX or using a word processor. Submit your solutions to Moodle as a PDF. Due date is strict. You do not have to print the questions. Your answers need not be on separate pages.

- This is an individual assignment, so work on your own.
- Please do not submit just an answer, and rather show all your reasoning.
- For any further questions, contact the assistant at utkan.gezer@boun.edu.tr.

1. **(50 pts)** Solve the following parts concerning the figure to the right.



Description of the figure The side length of the square is 1 unit. There are 4 quarter-circles at each corner. They each have a radius of 1 unit. The blue area is the intersection of those quarter-circles.

(a) Write a numerical probabilistic algorithm approximating the blue area in the figure.

Hint Write an inequality for the circular areas centered at each corner. A point satisfying all of those inequalities is a point in the blue area, and vice versa.

- First create a random point in the region of $x \geq 0, y \geq 0, x \leq 1, y \leq 1$.
- Then check its distances to corners.
- If distance to each corner is less than 1, it's in the blue region.
- Repeat this process 100 times.
- Ratio of number of hits to number of all generated points is an approximation for the area of the blue part.

Algorithm 1: Area approximation

Result: Approximate area of the blue area
Function *ApproximateArea()*
  $hit \leftarrow 0$;
  for $i \leftarrow 0$ *to* 100 **do**
    call Random({0,...,1000},x);
    call Random({0,...,1000},y);
    $x \leftarrow x/1000$;
    $y \leftarrow y/1000$;
    if $(x-0)^2 + (y-0)^2 < 1$ *and* $(x-1)^2 + (y-0)^2 < 1$ *and* $(x-0)^2 + (y-1)^2 < 1$ *and* $(x-1)^2 + (y-1)^2 < 1$ **then**
      $hit \leftarrow hit + 1$;
    end
  end
  return $hit/100$;
end

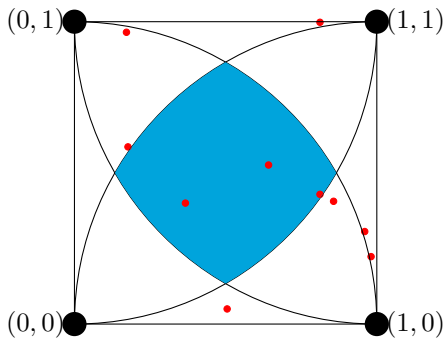
(b) Generate 10 random points within the square (x, y pairs between 0 and 1) using your favorite programming language or the Random Decimal Fraction Generator of Random.org.

Random Decimal Fraction Generator of Random.org is used.

$$\begin{bmatrix} (0.172, 0.965) & (0.812, 0.429) & (0.981, 0.223) & (0.177, 0.586) & (0.812, 0.998) \\ (0.367, 0.400) & (0.960, 0.306) & (0.505, 0.050) & (0.642, 0.526) & (0.857, 0.406) \end{bmatrix}$$

(c) Mark the points that lie within the blue area, and give an approximation for the blue area.

There are 10 selected points and 3 of them in the blue section. Ratio of hit points to all is an approximation for area of the blue area. Therefore area of the blue area is approximately 3/10.



2. **(50 pts)** Suppose that there are n-dimensional m lists, and that the first k of the lists are sorted in the ascending order. What is the smallest number of comparisons needed (i.e. the lower bound) in order to find the maximum element among the $m \cdot n$ elements, given the following conditions:

   Claim: Each of the n-1 elements in the list that is not the maximum must loose at least one comparison.
   Suppose the contrary: There are two elements (L[i],L[j]) that are not losing any comparisons. Suppose the algorithm outputs L[i] as the maximum. Then change L[j] with a value that greater than L[i]. The algorithm will do exactly the same comparisons. In the first execution, L[j] won every comparison; in the second execution also, L[j] will win every comparison. Algorithm executes with respect to results of comparisons, so both executions will be the same. Thus, algorithm will again output L[i] as the maximum, which is a contradiction.
   So, lower bound of finding maximum element of a list is n-1.

   (a) $k = 0$
   Finding maximum of a list takes $n - 1$ comparisons. Since it will be done for m lists, there are $m.(n - 1)$ operations until now. Finally we have to find the maximum value among the maximums of the lists. So, it takes $m - 1$ comparisons. In total there are $m.(n - 1) + m - 1$ comparisons.
   So lower bound is: **m.(n-1)+m-1**

   (b) $k = m$
   Since all of the lists are sorted in ascending order, last elements of the lists are the maximums of them. Since we already know the maximums of the lists, we just have to find the maximum value among the maximums of the lists. So, it takes $m - 1$ comparisons. In total there are $m - 1$ comparisons.
   So lower bound is: **m-1**

   (c) $0 < k < m$
   Finding maximum of a list takes $n - 1$ comparisons. Since it will be done for $m - k$ lists, there are $(m - k).(n - 1)$ operations until now. Other k lists are already sorted in ascending order. Therefore, last elements of them are the maximums. Finally we have to find the maximum value among the maximums of the lists. So, it takes $m - 1$ comparisons. In total there are $(m - k).(n - 1) + m - 1$ comparisons.
   So lower bound is: **(m-k).(n-1)+m-1**