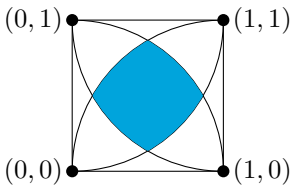
Cmpe 300: Homework 3 — Due: December 30th 17:00

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.

Answer

The function below evaluates whether the point is in the blue area or not, by checking the boundary conditions of four quarter circle.

function IsIn(x, y)
  if $x^2 + y^2 < 1 \wedge (x-1)^2 + y^2 < 1 \wedge x^2 + (y-1)^2 < 1 \wedge (x-1)^2 + (y-1)^2 < 1$ **then**
    return true
  else
    return false
  end if
end function

The function below evaluates the sample to analyze the points that rest in blue area, and returns the percentage of those who rest in blue area.

function NumericalProb($IsIn, x, y, Q$)
  $success \leftarrow 0$
  $i \leftarrow 0$
  for i **to** Q **do**
    $callRandom(\{0, ..., 1\}, x)$
    $callRandom(\{0, ..., 1\}, y)$
    if $IsIn(x, y)$ **then**
      $success \leftarrow success + 1$
    end if
  end for
return $success * 100$ / Q
end function

(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.

Answer

$Point1 = (0.530324, 0.131470)$ $Point2 = (0.427873, 0.840549)$ $Point3 = (0.911731, 0.560807)$
$Point4 = (0.993659, 0.225091)$ $Point5 = (0.164563, 0.522170)$ $Point6 = (0.807351, 0.124272)$
$Point7 = (0.637379, 0.421693)$ $Point8 = (0.343053, 0.717368)$ $Point9 = (0.929043, 0.377408)$
$Point10 = (0.539279, 0.457904)$ $Point11 = (0.799350, 0.391014)$ $Point12 = (0.029634, 0.570495)$
$Point13 = (0.535416, 0.203938)$ $Point14 = (0.570168, 0.518679)$ $Point15 = (0.493724, 0.511799)$
$Point16 = (0.025889, 0.024048)$ $Point17 = (0.643269, 0.453762)$ $Point18 = (0.864597, 0.555001)$
$Point19 = (0.014569, 0.858256)$ $Point20 = (0.780091, 0.179133)$

Point Generation Code, written in C++:

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string>

bool isIn(double x, double y){
    return  (x*x + y*y < 1)
                and ((x-1)*(x-1) + y*y < 1)
                and (x*x + (y-1)*(y-1) < 1)
                and ((x-1)*(x-1) + (y-1)*(y-1) < 1);
}

int main(int argc, char* argv[]){

    srand(time(NULL));
    int quantity = stoi(argv[1]);
    double x[quantity];
    double y[quantity];
    int hit = 0;

    for(int i=0; i<quantity ; i++){
        x[i] = (double)rand()/RAND_MAX;
        y[i] = (double)rand()/RAND_MAX;
    }
    for(int i=0; i<quantity; i++){
        if(isIn(x[i], y[i])){
            hit = hit + 1;
            printf("%f %f Inside.\n", x[i], y[i]);
        }
        else{
            printf("%f %f\n", x[i], y[i]);
        }
    }
    printf("%f\n", hit/(double)quantity);
    return 0;
}
```

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

Answer

Unmarked = Point1, Point2, Point3, Point4, Point6, Point9, Point11, Point12, Point16, Point18, Point19, Point20
Marked = Point5, Point7, Point8, Point10, Point13, Point14, Point15, Point17
Approximation = 40%

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:

   (a) $k = 0$

   Answer

   When there's no list that is sorted in ascending order, there has to be $n \cdot m - 1$ comparisons to evaluate the maximum element among $n \cdot m$ elements.
   The main reason is that no list among m lists are sorted before, hence the greatest element of each list is unknown. Since all lists are unsorted, it is necessary to compare all elements among m lists, which has dimension n. In the end, total comparison is $n \cdot m - 1$. (Check reference to lecture slides, Slide 5, Finding maximum and minimum value of a list. It will provide the complexity of the algorithm we use to solve this problem.)

   (b) $k = m$

   Answer

   When all m lists are sorted in ascending order, it is suffice to check the last element of each list. The last element of each list provides the largest element among the elements of their lists, hence comparing those last elements of each list will eventually provide the largest element among m lists. Hence, it is suffice to say that the lower bound for this problem is $m - 1$. (Check reference to lecture slides, Slide 5, Finding maximum and minimum value of a list. It will provide the complexity of the algorithm we use to solve this problem.)

   (c) $0 < k < m$

   Answer

   If there exist k list that are already sorted in ascending order, checking their last elements will be suffice to determine the largest element among k lists. The $m - k$ unsorted lists should be iterated to determine the largest element among $(m - k) \cdot n$ elements. Hence, $(k + (m - k) \cdot n) - 1$ comparisons are necessary to find out the largest element. (Check reference to lecture slides, Slide 5, Finding maximum and minimum value of a list. It will provide the complexity of the algorithm we use to solve this problem.)