Instituto Federal Catarinense – Campus Videira
Centro Acadêmico de Ciência da computação
Turma de Ciência da computação 2020
Disciplina: Algoritmos
Docente: Manassés Ribeiro.

Trabalho Final: Cálculo do PI.

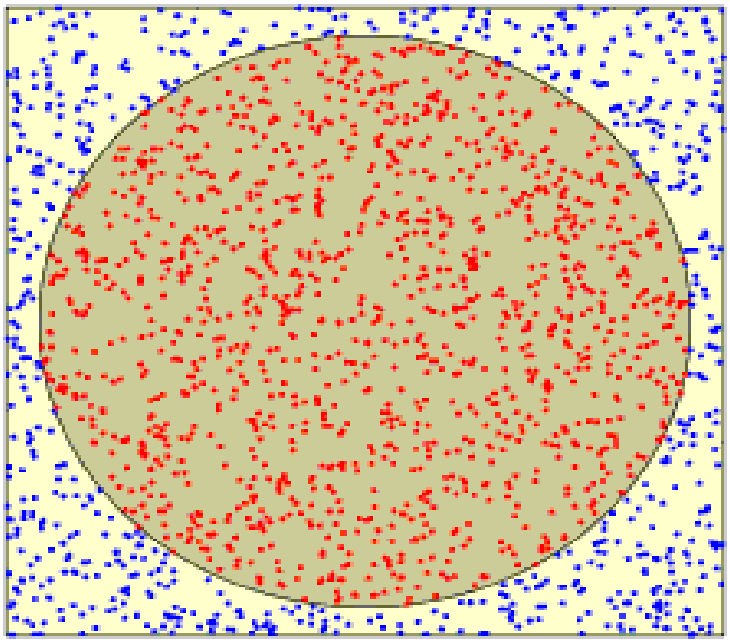
Claudionei Lovato Serafini

Videira,
2021

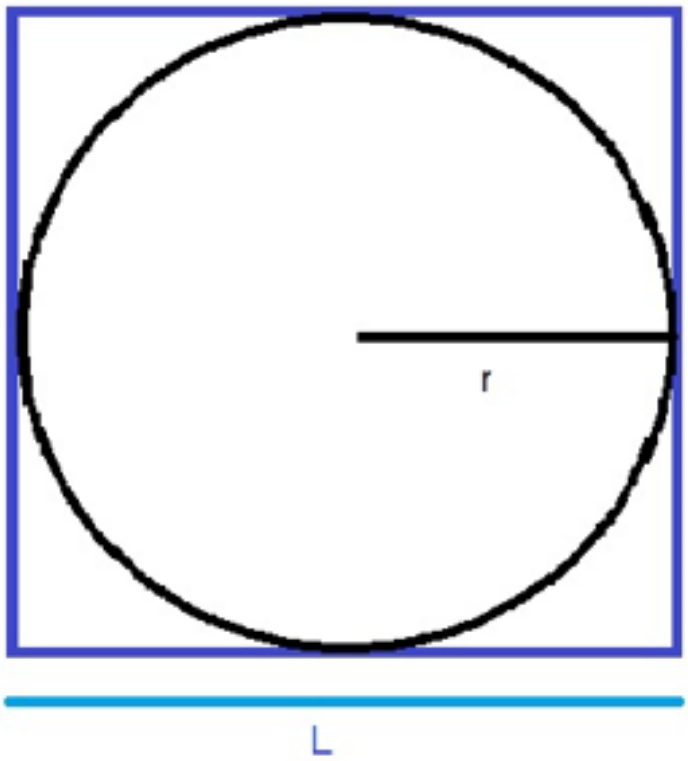
Problema 2: Calcular PI pelo método de monte Carlo

Oque é o método de Monte Carlo? Essa técnica foi, e é usado para várias áreas de estudo, como por exemplo no mundo financeiro, calculando e analisando o mercado para investimentos, além disso esse método foi muito importante na Segunda Guerra Mundial, sendo parte da construção das primeiras bombas atômicas, ele consiste em fazer simulações através de contas matemáticas e algoritmos criando vários cenários diferentes com valores aleatórios, assim, procurando o que chega mais perto do seu resultado desejado.

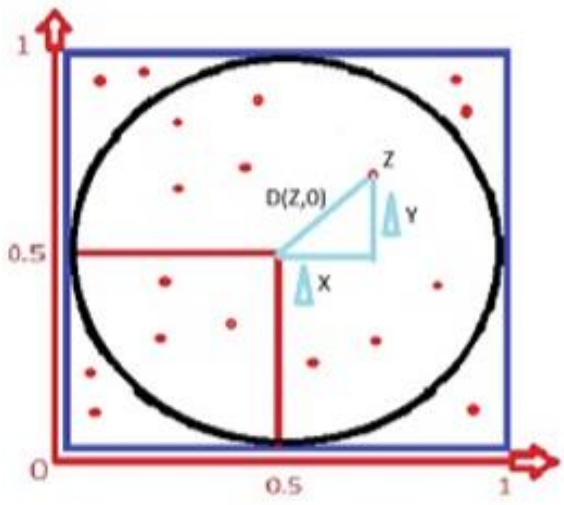
Nesse experimento, vamos usar o método de Monte Carlo para descobrir o valor de PI, será aplicado em algoritmos mais especificamente em linguagem **C**. Como já explicado o método faz simulações com valores aleatórios, então como cenário base usaremos esse gráfico:



Como esse gráfico se encaixa no método de Monte Carlo? Como se pode ver, a imagem acima, se compõe de um quadrado e dentro dele um círculo, seguidos de vários pontos azuis e vermelhos. Como dito anteriormente o método de Monte Carlo gera e lê valores aleatórios, pois bem, cada ponto dentro desta figura é um valor gerado aleatoriamente dentro do cenário, com isso podemos fazer um algoritmo que calcule o valor de PI, com os pontos vermelhos que caíram dentro dividindo pelos pontos azuis.



Tendo como base na imagem acima já temos por conhecimento que a **área da circunferência** = **π** * **r²**, em que **r** é o raio da circunferência, e também a **área do quadrado** = **L²**, onde L é o lado do quadrado. Fazendo uma relação entre o **r** do círculo e o **L** quadrado podemos transformar de **L²** para **4r²**, pois **L** é a mesma coisa que **(**2**r) ²**. Sendo a formula de PI: **π = área do círculo / área do quadrado (π r² / 4r²),** podemos cancelar os elementos em comum tendo por final **π = 4 * área do círculo / área do quadrado.**



Com base na imagem anterior, é escolhido um ponto Z que foi aleatoriamente gerado, calculamos a distância entre ele e o ponto de origem. Se traçar uma linha entre o ponto Z e a origem percebesse que se forma um triangulo retângulo, com sua hipotenusa D(Z,0), e seus catetos que são dados como), nisso podemos aplicar o teorema de Pitágoras onde a formula seria : $D(Z, 0) = \sqrt{(\Delta x)^2 + (\Delta y)^2}$, com a formula de Pitágoras conseguiremos calcular a distância entre os dois pontos, onde se o ponto tiver um valor menor ou igual a 0.5, ele vai estar dentro do círculo, se tiver um valor maior que 0.5, estará fora.

Iniciei abrindo um registro **< *typedef struct* >**, com as variáveis *float x* e *float y* que guardará os valores das coordenadas, e nomeie a struct como **Ponto**.

```
// |------------------------------------------------ Struct ------------------------------------------------|

typedef struct{
        float x;
        float y;
}Ponto;
```

Então comecei a com um modulo para o usuário informar os valores dos pontos, esse modulo levou o nome de ***carregaPontos***, onde o usuário irá definir o máximo de pontos que irá ter no cenário, retornando assim um valor de inteiros.

```
//Usuario carrega quantidade de pontos para serem distribuidos.
int carregaPontos(){
       int point_max;
       printf("\n\n|============== |ESCREVA| ==============|\n|      O número maximo de pontos,      |\n|      para calcular o valor de PI.
       scanf("%i",&point_max);
       return point_max;
}
```

O valor que retornará do modulo ***carregaPontos*** será atribuído para a variável **tam** de valor de inteiros, que servira para grande parte das funções do algoritmo.

```
//Main
int main(void){
        int tam = carregaPontos();
```

Com o cenário já delimitado com a quantidade de pontos, geramos as coordenadas aleatórias para *x e y* de 0 até 1, com a função rand(), enviando para o vetor vet_p[].

```
void gerarRandom(Ponto** vet_p, int tam){
        int i;
        Ponto* p;
        srand(time(NULL));

        for(i = 0; i < tam;i++){
                p = alocaMemoriaDoPonto();
                p->x = rand() / (float)RAND_MAX;
                p->y = rand() / (float)RAND_MAX;
                vet_p[i] = p;
        }
}
```

O vetor vet_p foi criado anteriormente na main como um “objeto” que recebe/passa os valores por parâmetro.

```
//Main
int main(void){
        int tam = carregaPontos();
  Ponto** vet_p;
```

Com os valores gerados no nosso cenário e guardados em suas respectivas variáveis, usando o teorema de Pitágoras, e subtraindo pela metade do raio, conseguimos obter os pontos que estão dentro ou fora do círculo, usando uma condição se a distância for menor que a metade do raio, será considerado dentro do círculo.

```
float calculaPontosCirculo(Ponto** vet_p, int tam){
        int i;
        float distancia;
        float pCirc;

        for(i = 0;i < tam;i++){
                distancia = sqrt((pow((vet_p[i]->x - 0.5),2)) + (pow(vet_p[i]->y - 0.5,2)));
                if(distancia <= 0.5){
                        pCirc ++;
                }
        }
        return pCirc;
}
```

Os valores são armazenados na variável criada na main denomina de pCirculo, que representa os pontos que caíram dentro do círculo.

```
//Main
int main(void){
        int tam = carregaPontos();
        Ponto** vet_p;

        //Aloca memoria vetor dinamico:
        vet_p = alocaMemoriaVetorDinamico(tam);

        gerarRandom(vet_p, tam);
        float pCirculo = calculaPontosCirculo(vet_p, tam);
        escrevaResultados(pCirculo, tam);

        //liberar a memória
        desalocaVetorDinamico(vet_p, tam);
        return 0;
}
```

Por final os valores são enviados para o modulo **escrevaResultados**, para serem calculados com o valor aproximado de **π,** com o uso da formula já definida **π = 4 * área do círculo / área do quadrado,** e em seguida imprimidos.

```
void escrevaResultados(float pCirculo, int tam){
        float pi = 4 * (pCirculo / (float)tam);
        printf("|\n|============= |RESULTADO| =============|");
        printf("\n|\n|         Valor de PI: %.5f",pi);
        printf("\n|     Pontos dentro do circulo: %.0f", pCirculo);
        printf("\n|      Pontos fora do circulo: %.0f", tam-pCirculo);
        printf("\n|\n|=============== |ALUNO| ===============|\n|      Claudionei Lovato Serafini.
}
```

Faltou coisa, mas ia passar de 700 linhas, porem está resumido.