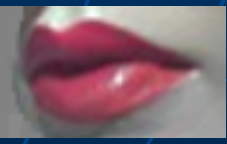
Metode Simulasi Monte Carlo

Review Algoritma
Probabilitas pelemparan coin Tunggal
Probabilitas pelemparan coin Ganda
Nilai π
Nilai Integral
Kasus nilai f(x) = xsin(x)

Oleh:
Tri Budi Santoso
Achmad Basuki
Miftahul Huda

EEPIS-ITS

Speech Signal Processing



Review Algoritma Monte Carlo

- Merupakan dasar dari semua algoritma untuk metode simulasi
- Didasari pada pemikiran penyelesaian suatu masalah untuk mendapatkan hasil lebih baik dengan cara memberi alternatif nilai sebanyak-banyaknya (nilai terbangkit) untuk mendapatkan tingkat ketelitian yang lebih tinggi
- Misal untuk memperoleh tingkat ketelitian sampai 0,01 maka diperlukan pembangkitan nilai sebanyak 10000, dsb.
- Teknik pembuatan program bersifat bebas hampir tidak ada rule yang terlalu mengikat

Setiap masalah simulasi dapat didekati dengan metode Monte Carlo ?

Dengan catatan kunci:

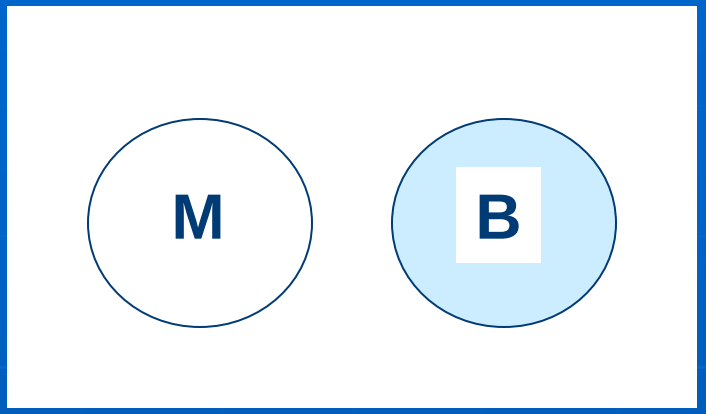
- Mampu memformulasikan masalah
- Membuat Overview Sistem
- Penyederhanaan sistem menuju algorithma
- Menyusun algoritma
- Pembuatan Program

Probabilitas pelemparan coin Tunggal

Sebuah pelemparan coin sebanyak 100 kali diperoleh hasil 45 kali keluar muka dan 55 kali keluar belakang. Dari data ini didapat nilai probabilitas untuk keluarnya
- muka sebagai p(M) = N(M)/N_total
- belakang sebagai p(B) = N(B)/N_total

Dari model diatas susun algorithma dan program
untuk kasus pelemparan sebanyak 1000 kali 2000 kali dsb… sampai dengan 10000 kali.
Apa yang terjadi?

Algorithma



1. Bangkitkan nilai 0/1 sebanyak 1000 kali (N=1000) dengan cara:
   n =(int)rand()%2

2. Klasifikasi
   Jika n=0, maka M=M+1
   Jika n=1, maka B=B+1

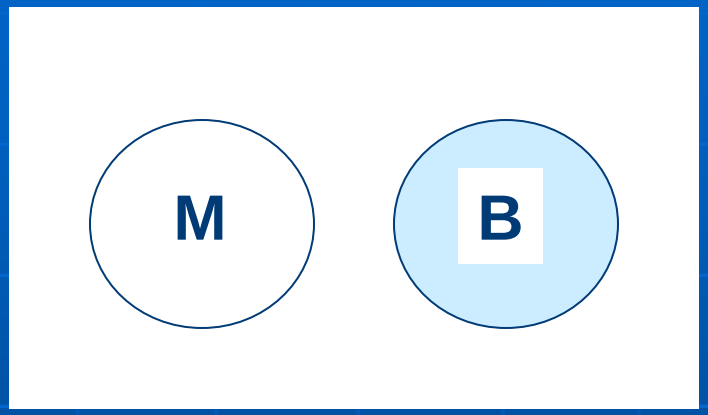
3. Hitung probabilitas M dengan cara M/N dan probabilitas B dengan cara B/N

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

#define N 1001
int nn=N-1;
int x[N];

void bangkit()
{
 for (int i=0;i<nn; i++)
 {
  x[i]=(int)rand()%2;
  //printf("\n x[%d]: %d",i,x[i]);
 }
}
```

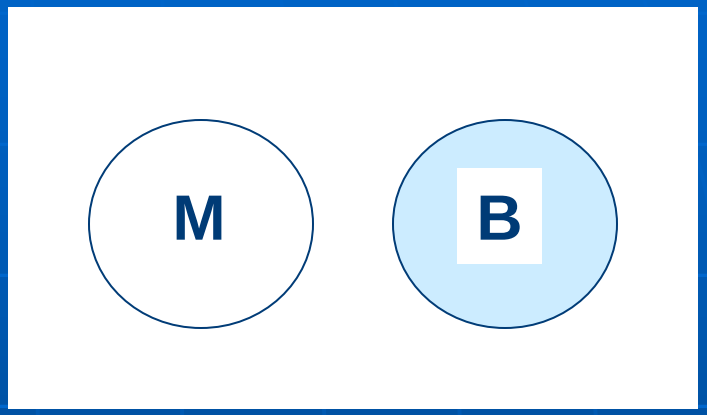
```
void klasfikasi()
{ double M=0.0,B=0.0;

 for (int i=0;i<nn; i++)
 {
  if(x[i]==0)
            M=M+1;
  else
            B=B+1;
 }
 printf("\n Nilai Terbangkit");
 printf("\n M=%f \t B=%f",M,B);
 printf("\n\n Probabilitas");
 printf("\n p(M)=%f \t p(B)=%f",M/nn,B/nn);
}

void main()
{
 bangkit();
 klasfikasi();
}
```

Probabilitas pelemparan coin Ganda

Coin 1



Coin 2



Dari teori peluang akan muncul:
MM ➔ ¼
MB atau BM ➔ ½
BB ➔ ¼

Dengan metode Monte Carlo dapatkan tingkat ketelitian sampai 0.01 untuk menyelesaikan kasus tersebut….

Model Sistem menuju Algoritma

Untuk mendapatkan ketelitian sampai 0,01 maka harus dilakukan pelemparan sebanyak 1000 (N_total) kali.
Dari hasil pelemparan catat keluarnya angka-angka:

$$N(MM) = \sum(MM) = ...kali$$
$$N(MB) = \sum(MB) = ...kali$$
$$N(BB) = \sum(BB) = ...kali$$

Dari hasil diatas hitung peluang dengan cara:

P(MM) = N(MM)/N_total
P(MB) = N(MB)/N_total
P(BB) = N(BB)/N_total

Algorithma

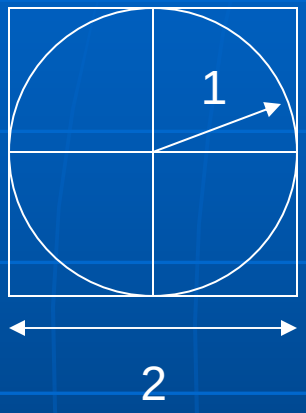
1. Bangkitkan nilai 0/1 sebanyak 1000 kali (N=1000) dengan cara:
   n1 =(int)rand()%2 dan n2 =(int)rand()%2

2. Klasifikasi
   Jika n1=0 dan n2=0, maka MM=MM+1
   Jika n1=0 dan n2=1 atau n1=1 dan n2=0
   maka MB=MB+1
   Jika n1=1 dan n2=1, maka BB=BB+1

3. Hitung probabilitas MM dengan cara N(MM)/N dan probabilitas untuk nilai MB serta BB

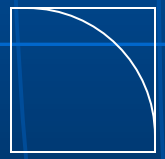
Metode Monte Carlo Untuk Menghitung Nilai π

Suatu lingkaran berjari-jari 1 diletakkan ke kotak bujursangkar bersisi 2.



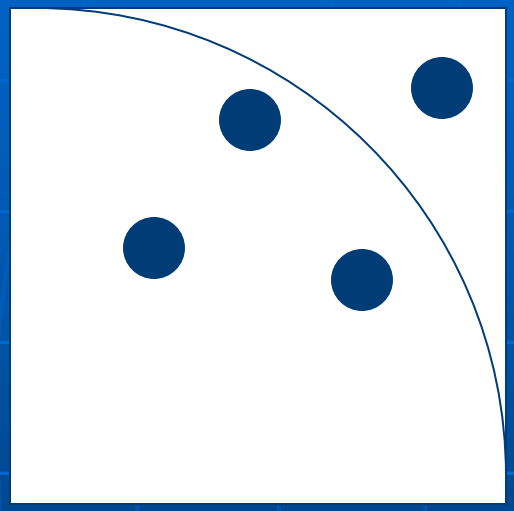
Dari gambar disamping didapatkan perbandingan:
Luas lingkaran / Luas bujur sangkar = πr^2 / (2x2)

Asumsikan bahwa r =1, dan lanjutkan dengan mengambil satu kotak serta seperempat lingkaran sehingga gambarnya menjadi:



Maka perbandingan luas menjadi $\frac{1}{4}\pi r^2$ / 1x1 = $\frac{1}{4}\pi r^2$

Anda lemparkan sebuah kelereng kecil secara bebas ke dalam kotak sebanyak 4 kali, maka kemungkinan kelereng akan jatuh di dalam lingaran atau diluar lingkaran. Tetapi tetap dalam kotak tersebut…..



Bayangkan bila anda melakukan pelemparan kelereng sebanyak 1000 kali (N=1000).

Kemudian kita tetapkan bahwa posisi kelereng berada dalam lingkaran terjadi sebanyak M kali

Banyaknya nilai M dibanding total pelemparan N akan senilai dengan perbandingan luas seperempat lingkaran dibanding 1 kotak atau

N/M = ¼ π

Sehingga:

$$\pi = \frac{N}{4M}$$

Algorithma

1. Buat inisialisasi m =1, N=1000
2. Bangkitkan nilai x secara random yang memiliki nilai 0~1
3. Bangkitkan nilai y secara random yang memiliki nilai 0~1
4. Hitung r_2 =x*x + y*y
5. Jika nilai r<1, maka m=m+1,
   Jika tidak, maka m=m

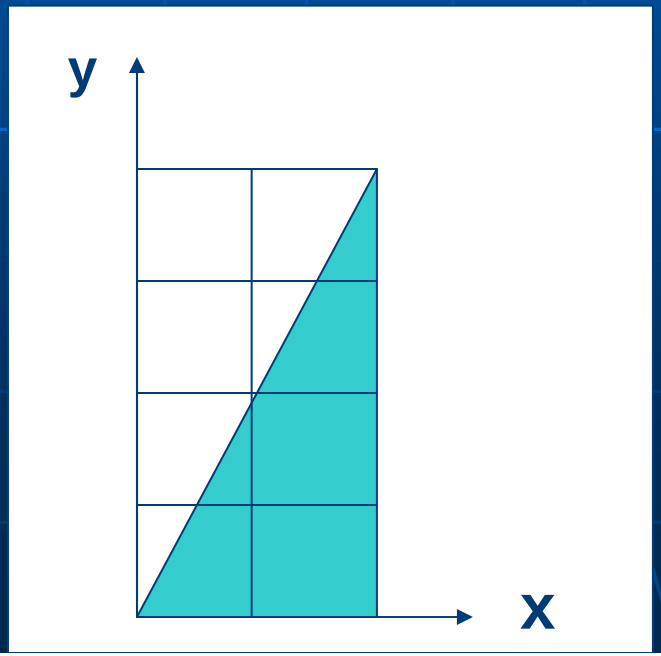
6. Lakukan proses looping ini sampai 1000 kali.
7. Setelah proses looping selesai, hitung nilai p = 4*m/N.

Menghitung Nilai Integral

Mencari nilai integral dalam hal ini dengan suatu fungsi f(x) =2x

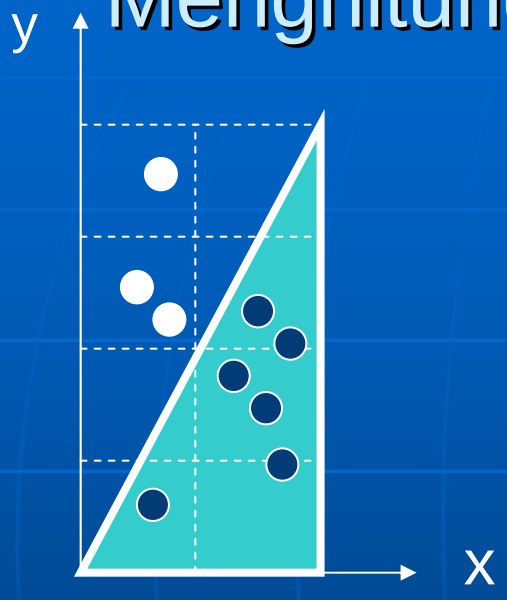
Luas: $$Luas = \int_{x1}^{x2} f(x)dx = \int_{0}^{2} 2xdx$$

Secara matematis didapatkan sebagai:



$$Luas = x^2\Big|_0^2 = 4$$

Menghitung dengan Metode Monte Carlo



Luas area dicari = yang berwarna atau daerah dibawah garis fungsi f(x) = 2x

Dengan dasar pemikiran tersebut diperoleh suatu perbandingan:

$$\frac{Luas\ area\ diarsir}{Luas\ total\ kotak} = \frac{luas\ yang\ dicari}{luas\left(2\times 4\right)}$$

BIla kita melakukan pelemparan coin sebanyak N kali, dan coin jatuh di bawah garis f(x) =2x sebanyak M kali. Maka:

$$\frac{luas\ yang\ dicari}{8} = \frac{M}{N}$$

Algorithma

- Bangkitkan 2 bilangan acak x(0~2) dan y(0~4) sebanyak N =1000
- Bila y<f(x) maka x diterima, bila tidak ditolak
- Ulangi langkah (2) dan (3), dan hitung banyaknya titik yang diterima.
- Luas atau integral f(x) adalah hasil perbandingan (banyaknya titik yang diterima dengan banyaknya bilangan terbangkit) dikalikan luas kotak

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

#define N 1001
int nn=N-1;
double pi;
void monte_carlo()  //program untuk nilai pi
{
 double R,x,y,m=0.0;
 for (int i=1;i<=nn;i++) {
            x=(double)rand()/(RAND_MAX);
            y=(double)rand()/(RAND_MAX);
            R = x*x + y*y;
            if(R<1.0){
             m=m+1.0;
            }
            else
            m=m;
      }
            pi=4*m/(nn);
            printf("\nSimulated Annealing menghasilkan pi= %f",pi);
}
```

```
void main()
{
            monte_carlo();
}
```