بسم الله الرحمن الرحيم

كتاب نمذجة ومحاكاة

جامعة افريقيا العالمية

اعداد : رامى الطيب مصطفى البشير

السـودان

Rami_4@hotmail.com

- يهدف هذا الكتاب الى برمجة بعض تطبيقات انظمة المحاكاة

المصدر الاساسي

مذكرة/ النمذجة والمحاكاة

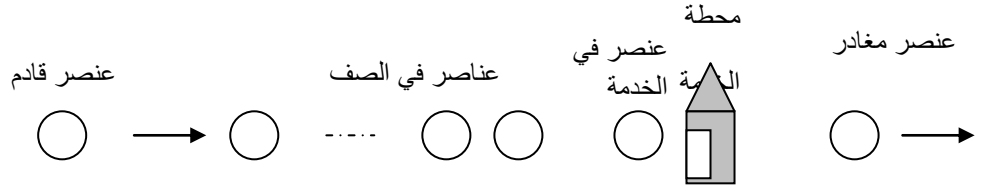
إعداد/أ. سلمى الناصر

فهرسة المحتويات

نظام صفوف بمحطة خدمة واحدة

Single Server Queue

النظام عبارة عن مجموعة من الزبائن ينتظرون أن تتم خدمتهم عن طريق محطة خدمة، عندما يصل الزبون ينتظر في الصف ثم تتم خدمته ثم يغادر، الشكل التالي يوضح ذلك:



مكونات النظام:

١. وقت الإنتظار: الوقت الذي يقضيه الزبون في الصف منذ ساعة وصوله وحتى بداية فترة خدمته.
٢. محطة الخدمة: المكان الذي تتم فيه خدمة الزبون.
٣. وقت الفراغ (المحطة عاطلة عن العمل): الوقت الذي تقضيه المحطة في انتظار الزبون الأقرب.
٤. طول الصف: عدد العناصر في الصف.
٥. فترة الخدمة: طول المدة التي يتم فيها خدمة الزبون.
٦. تتالي الوصول(اوقات الوصول البينية): الفترة الزمنية بين وصول الزبائن.

الهدف من المحاكاة:

- تقدير متوسط زمن تأخير العناصر في الصف.
- تقدير متوسط عدد العناصر في الصف.

نتائج عملية المحاكاة:

- التقدير لإضافة محطة جديدة.
- التقدير لرفع كفاءة المحطة.

أمثلة للنموذج:

- خدمة الصرف في بنك.
- عيادة طبيب.
- محطة خدمة بنزين.
- تسجيل الطلاب.

متغيرات النظام:

١. حالة المحطة عاطلة/مشغولة.
٢. عدد الزبائن في الصف.
٣. أوقات وصول العناصر الى مركز الخدمة (حدث وصول).
٤. فترة خدمة الزبائن.
٥. أوقات مغادرة الزبائن (حدث مغادرة).
٦. قوانين الخدمة: اولوية الخدمة وسنعتبرها FIFS الواصل أولالأ تتم خدمته اولاً "First In First Served".

ترميز:

حر : وقت وقوع الحدث رقم ر (ح. = صفر)

ور : وقت وصول العنصر رقم ر (و. = صفر)

فر : الفترة بين وصول العتصر رقم ر-١ والعنصر رقم ر.

خر : فترة خدمة العنصر رقم ر.

تر : فترة تأخير (انتظار) العنصر رقم ر في الصف.

م ر : وقت مغادرة العنصر رقم ر.

ع ر : وقت فراغ المحطة.

نر : عدد العناصر في الصف عند وصول العنصر رقم ر.

تر = م ر-١ - ور

لر = ور - م ر-١

م ر = ور + تر + خر = م ر-١ + خر

المتغيرات العشوائية:

١. الازمان البينية لوصول الزبائن ف١ ،ف٢،ف٣، .....
٢. ازمان الخدمة (طول مدة الخدمة) : خ١، خ٢،خ٣،.....

تعريف المشكلة:

اذا كان لدينا محطة خدمة يصل اليها الزبائن في اوقات بينية ف١،ف٢،ف٣،.... وتتم خدمتهم في اوقات خ١،خ٢،خ٣،.... الزبون الذي يصل اولاًويجد محطة الخدمة عاطلة يدخل في الخدمة مباشرة. الزبون الذي يصل ويجد المحطة مشغولة يقف في آخر الصف. تستمر المحطة في خدمة الزبائن بنظام الأول في الوصول هو الأول في الخدمة. انظر الشكل أعلاه.

المطلوب هو محاكاة هذا النظام حتى يكمل ن زبون خدمتهم وتقدير مقاييس الأداء التالية:

١. متوسط تأخير الزبون في الصف ($ت_ن$) =
مجموع تأخير الزبائن/عدد الزبائن الذين تمت محاكاتهم

٢. متوسط عدد العناصر في الصف (ن) =
مجموع عدد العناصر في الصف/عدد الزبائن الذين تمت محاكاتهم

٣. متوسط وقت فراغ المحطة (عاطلة $ع_ن$) =
مجموع اوقات الفراغ/عدد الزبائن الذين تمت محاكاتهم

مثال:

لمحاكاة نظام الصرف في بنك، اذا كان لدينا تتالي الوصول ف ر وتتالي الخدمة خ ر التاليين:

ر	١	٢	٣	٤	٥	٦	٧	٨	٩	١٠
ف ر	-	٥٥	٣٢	٢٤	٤٠	١٢	٢٠	١٠	١٥	٢٣
خ ر	٤٣	٣٦	٣٤	٤٠	٣٥	٢٠	٣٠	٤١	١٠	٩
و ر	٠	٥٥	٨٧	١١١	١٥١	١٦٣	١٨٣	١٩٣	٢٠٨	٢٣١
م ر	٤٣	٩١	١٢٥	١٦٥	٢٠٠	٢٢٠	٢٥٠	٢٩١	٣٠١	٣١٠
ت ر	٠	٠	٤	١٤	١٤	٣٧	٣٧	٥٧	٨٣	٧٠
ن ر	٠	٠	١	١	١	٢	٢	٣	٣	٣
ع ر	٠	١٢	٠	٠	٠	٠	٠	٠	٠	٠

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
void main()
{
clrscr();
double a,b,c;
int f[10],w[10],t[10],n[10],h[10],m[10],s[10],j;
for(j=1;j<=10;j++)
{
cout<<"The period between the arrival of the item f["<<j<<"]=";
cin>>f[j];
}
for(j=1;j<=10;j++)
{
cout<<"Racial period of service s["<<j<<"]=";
cin>>s[j];
 }

w[0]=0;
m[0]=0;
int x;
for(int i=1;i<=10;i++)
{
 w[i]=w[i-1]+f[i];

 x=m[i-1]-w[i];
```

```
if(x>0)
{
t[i]=x;
}
else
t[i]=0;

m[i]=w[i]+t[i]+s[i];

int y;
y=w[i]-m[i-1];
if(y>0)
h[i]=y;
else
h[i]=0;
}
int i;
int z;
z=0;
for (i=1;i<=10;i++)
{
for(int j=1;j<=i;j++)
{
if(w[i]<m[i-j])
{

z=z+1;
```

```
}
}
n[i]=z;
z=0;
}
for(i=1;i<=10;i++)
{
  a=a+n[i];
  b=b+t[i];
  c=c+h[i];

 }
 cout<<"Element waiting period t="<<b/10<<endl;
 cout<<"The number of elements in the row n="<<a/10<<endl;
 cout<<"time empty station h="<<c/10;
getch();

}
```

محاكاة نظام مطاردة

١/ وصف النظام:

يتكون النظام من طائرة مقاتلة fighter تقوم بمطاردة طائرة أخرى قاذفة للصواريخ bomber للحاق بها وتدميرها.

دراسة هذا النظام تساعد في المجالات العسكرية حيث تمكن من وضع التصميمات اللازمة لصنع الطائرة المقاتلة fighter والطائرة المهاجمة bomber.

- سنطلق على الطائرة المقاتلة اسم "المطارِد" و على الطائرة الأخرى اسم "الهدف".
- ينطلق الهدف في مسار معلوم أو محدد أثناء المطاردة.
- يقوم المطارِد بتغيير مساره قي إتجاه الهدف.

٢/ متغيرات النظام:

١. سرعة المطارِد V_f .
٢. موقع الهدف (متغير مسيطر عليه).

٣/ الغرض من المحاكاة/الهدف من المحاكاة:

معرفة السرعة المناسبة للمطارِد V_f التي تمكنه من اللحاق بالهدف في زمن محدد من بداية المطارة. أي إيجاد V_f المناسبة التي تمكن المطارِد من الإقترلب من الهدف بمسافة أقل من أو تساوي r (مساة محددة يستطيع منها اسقاط الهدف) في خلال فترة زمنية t.

إذن إذا سقط الهدف في نهاية المحاكاة فإن V_f هي السرعة المطلوبة للمطارِد و إذا لم يسقط الهدف يتم تعديل V_f .

٤/ الإفتراضات:

١. نفترض أن المطاردة تقع في بعدين، أي في المستوى XY
٢. تقسم الفترة الزمنية (0,t) الى n فترة زمنية.حيث t هي الفترة الزمنية للمطاردة.
٣. نفترض أننا نعرف إحداثيات الهدف عند بداية كل فترة زمنية.
٤. نفترض أن المطارِد يتجه فيإتجاه الهدف عند بداية كل فترة زمنية ولمدة Δt

٥/ بناء النموذج:

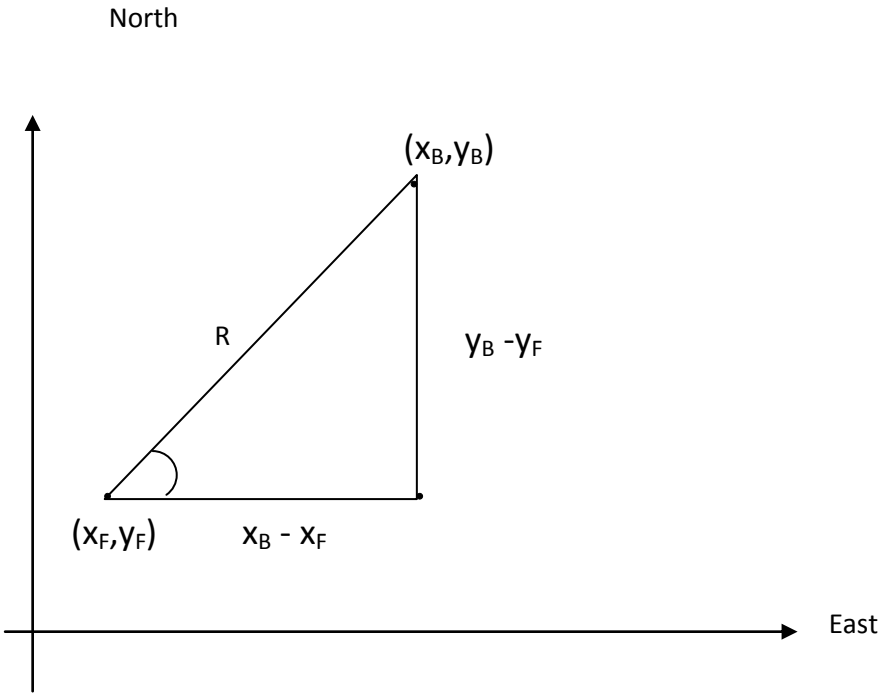
- إذا كان الهدف عند النقطة (x_B,y_B) في نفس الزمن الذي كان فيه المطارِد عند النقطة (x_F,y_F) و كان المطارِد يتجه في هذه اللحظة نحو الهدف.
- بعد مدة زمنية Δt و إذا استمر المطارِد في هذا الإتجاه يصبح عند النقطة (x'_F, y'_F).

حيث:

$$x'_F = x_F + v_F \cos\theta \times \Delta t$$

$$y'_F = y_F + v_F \sin\theta \times \Delta t$$

أنظر الشكل التالي:



$$R = \sqrt{(x_B - x_F)^2 + (y_B - y_F)^2}$$

$$\cos\theta = \frac{x_B - x_F}{R}$$

$$\sin\theta = \frac{y_B - y_F}{R}$$

المسافة الأفقية = $v_F \cos\theta \times \Delta t$

المسافة الرأسية = $v_F \sin\theta \times \Delta t$

(المسافة = السرعة X الزمن)

- بعد كل Δt تتجدد قيمة x'_F, y'_F بناءً على حركة الهدف.
- بعد كل مدة زمنية Δt يتم حساب المسافة بين المطارِد والهدف، إذا كانت أقل من أو تساوي المسافة المطلوبة ($R \le r$) يتم إسقاط الهدف وتنتهي عملية المحاكاة.
- إذا إنتهى زمن المحاكاة بدون إسقاط الهدف يعتبر الهف قد هرب من المطارِد.

توضيحات:

١. t هي مدة عملية المحاكاة، مثلاً عشر دقائق.
٢. T هي ساعة المحاكاة تزداد بقيمة Δt (طريقة التحريك الثابت).
٣. r هي المسافة المناسبة لإسقاط الهدف.
٤. R هي المسافة بين المطارد والهدف في بداية كل فترة زمنية.

مثال:

إذا كانت إحداثيات الهدف في مطاردة بين طائرتين مقاتلتين هي:

T	0	1	2	3	4	5	6	7	8	9
$x_B[T]$	3	7	-1	10	10	5	0	-5	-٥	١
$y_B[T]$	6	12	4	10	40	٥٢	14	4	٢	5

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
int round(double x)
{
  return int(x > 0.0 ? x + 0.5 : x - 0.5);
}
void main()
{
int xb[10],yb[10],xf[10],yf[10],vf,r,R,t,T,dt;
double z,x,y,a,b;
for (T=0;T<=9;T++)
{
cout<<"enter axis coordinates x["<<T<<"]=";
cin>>xb[T];
cout<<"enter axis coordinates y["<<T<<"]=";
cin>>yb[T];
}
r=4;
t=9;
vf=20;
xf[0]=1;
yf[0]=2;
dt=1;
for(T=0;T<9;T++)
{
a=pow((xb[T]-xf[T]),2);
```

```
b=pow((yb[T]-yf[T]),2);
cout<<"y="<<(a+b)<<endl;
z=sqrt(a+b);
R=ceil(z);
cout<<"R="<<R<<endl;
if(R<=r)
{
cout<<"the has been destroied\n";

 cout<<"("<<xb[T]<<","<<yb[T]<<")"<<"/t R="<<R<<endl<<"xf="<<xf[T]<<"yf="<<yf[T];

break;
}
if(T==t)
{
cout<<"the target escaped\n";
break;
}
x=xf[T]+vf*(xb[T]-xf[T])/R*dt;
xf[T+1]=round(x);
y=yf[T]+vf*(yb[T]-yf[T])/R*dt;
yf[T+1]=round(y);
cout<<"xf="<<x<<"yf="<<y<<endl;
}
getch();
}
```

محاكاة نظام بناء سد

١/ وصف النظام:

عبارة عن مشروع لبناء سد (خزان) على مجرى نهر لتغطية إحتياجات الري في مشروعع زراعي ل N سنة.

٢/ الغرض من الدراسة:

معرفة السعة capacity المناسبة التي تفي إحتياجات الري دون التأثير على مجرى النهر بعد الخزان وذلك ل N سنة قادمة.

٣/ المتغيرات التي تصف النظام:

١. سعة الخزان cap
٢. حجم المياه الموجود حالياً في الخزان. V (volume)

٤/ الإفتراضات:

١. نفترض وجود دراسة تمكن من معرفة دالة الإحتياجات Demands

$$Dem = f_D(y,m)$$

لإيجاد إحتياجات الري لكل سنة y و شهر m.

٢. نفترض وجود دراسة لمعرفة الدالة Flow:

$$Flow = f_F(y,m)$$

لإيجاد كمية الماء الوارد للخزان من أعالي النهر لكل سنة y وشهر m.

٣. نفترض وجود دراسة كافية عن الأمطار في منطقة الخزان تمكن من معرفة دالة الأمطار:

$$Rain = f_R(y,m)$$

لإيجاد حجم الامطارالتي تسقط على بحيرة الخزان كل سنة y وشهر m.

٤. نفترض وجود دراسة لمنطقة الخزان لتعريف دالة التبخر Evaporation:

$$Evap = f_E(V,m)$$

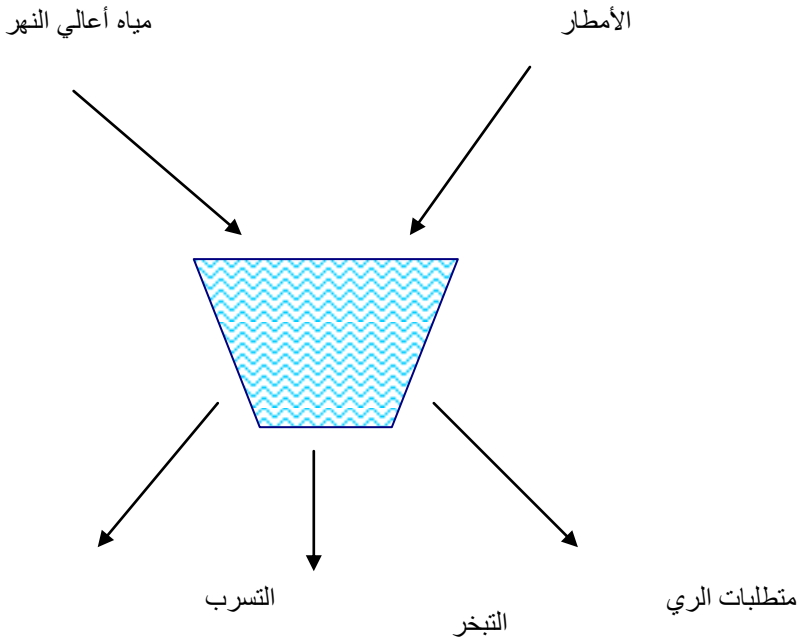
لإيجاد حجم الماء المفقود بالتبخر لكل شهر m من الحجم V (لأن التبخر يتناسب طردياً مع مساحة السطح)

٥. نفترض وجود دراسة كافية للترة في منطقة الخزان لمعرفة الماء المفقود بالتسرب Leak:

$$Leak = f_L(V)$$

والذي يعتمد على حجم الماء الموجود.

٥/ بناء النموذج:



إذا كان V هو حجم المياه الموجود في بحيرة السد عند بداية الشهر m للسنة y فإن:

١. المياه التي تأتي (تضاف) الى بحيرة الخزان فب هذا الشهر تعطى بالعلاقة :
Flow + Rain

عايه فإن الحجم الكلي للمياه في بحيرة الخزان هو:

Gross V = V + Flow + Rain

٢. المياه الكلية المفقودة بواسطة التبخر والتسرب تعطى ب:
TLoss = Leak + Evap

عليه فإن حجم الماء الصافي خلال الشهرm هو:

Net V = Gross V – Tloss

الآن بمعرفة حجم الماء الصافي net V و السعة cap و متطلبات الري Dem يمكن تحديد قيم المتغيرات التالية والتي تمثل المخرجات التي تحدد إذا ما كانت السعة المعطاة هي السعة المناسبة للسد:

١. القصور ي مقابلة إحتياجات الري short.
٢. حجم المياه المتدفقة فوق السد الى مجرى النهر spill

```
#include <iostream.h>

#include <conio.h>

void main()

{


int  n,V,cap,Dem,Flow,Rain,Gross_V,Evap,TLoss,Net_V,Short,Spill,y,m,Diff,Leak;

cout<<"enter capacity cap=";

cin>>cap;
```

```
cout<<"enter Demands  dem=";
cin>>Dem;
cout<<"enter number of years =";
cin>>n;
cout<<"enter volume v=";
cin>>V;



for(y=1;y<=n;y++)
for(m=1;m<=12;m++)
{

cout<<"enter flow for year and month("<<y<<","<<m<<")=";
cin>>Flow;
cout<<"enter rain for year and month("<<y<<","<<m<<")=";
cin>>Rain;
Gross_V = V + Flow + Rain;
cout<<"enter   Evap  the volume and month ("<<V<<","<<"m)=";
cin>>Evap;
cout<<"enter   Evap  the volume ("<<V<<")=";
cin>>Leak;
TLoss = Evap + Leak;
Net_V = Gross_V - TLoss;
if (Net_V <=0 )
{
V =0;
Short = Dem;
```

```
Spill = 0;

cout<<"the volume ="<<V<<endl<<"the short="<<Short<<endl<<"the
spill="<<Spill<<endl<<"the year ="<<y<<endl<<"the month ="<<m;

}

else

{

Diff = Net_V - Dem;

if (Diff   <= 0 )

{

  V = 0 ;

  Short = - Diff;

  Spill = 0 ;

 cout<<"the volume ="<<V<<endl<<"the short="<<Short<<endl<<"the
spill="<<Spill<<endl<<"the year ="<<y<<endl<<"the month ="<<m;

 }


else

   if (Diff  <=  cap)

   {

     V = Diff;

    Short = 0;

    Spill = 0;

    cout<<"the volume ="<<V<<endl<<"the short="<<Short<<endl<<"the
spill="<<Spill<<endl<<"the year ="<<y<<endl<<"the month ="<<m;

    }

else

{

    V = cap;

    Short = 0;
```

```
   Spill = Diff - cap;

   cout<<"the volume ="<<V<<endl<<"the short="<<Short<<endl<<"the
spill="<<Spill<<endl<<"the year ="<<y<<endl<<"the month ="<<m;

 }

 }

 }


 getch();

 }
```

محاكاة النمو السكاني

وصف النظام:

النظام عبارة عن أفراد مجتمع – انساني أو حيوني أو بكتريا ...الخ - يتزايدون أو يتناقصون مع الزمن

المتغيرات التي تصف النظام:

المتغير $p(t) \equiv$ عدد السكان في اللحظة t.

الافتراضات:

١. يعتمد نمو السكان على عدد المواليد و الوفيات فقط.
٢. يتناسب عدد المواليد والوفيات طردياً ويعتمد على طول الفترة الزمنية $(t, \Delta t)$ وعلى
عدد السكان في بداية تلك الفترة.

بناء النموذج:

١. الزيادة في عدد السكان في الفترة $(t, \Delta t)$ يعطى بالقيمة $Bp(t)\Delta t$.
٢. النقصان في عدد السكان في الفترة $(t, \Delta t)$ يعطى بالقيمة $Dp(t)\Delta t$.
حيث B يمثل ثابت التناسب للمواليد.

D يمثل ثابت التناسب للوفيات.

النمو السكاني $\propto$ الزيادة في السكان $p(t+\Delta t)$ .....(١)

النمو السكاني $\propto$ النقصان في السكان $p(t)$ .....(٢)

ومنها

$$p(t+\Delta t) = Bp(t)\Delta t.....(!)$$

$$= Dp(t)\Delta t.....(2) \qquad p(t)$$

بطرح(١) من (٢) نتحصل على المعادلة:

$$p(t+\Delta)-p(t)=(B-D)\,p(t)\Delta t.....(3)$$

المعادلة (٣) تمثل نموذج تغير السكان مع الزمن.

بقسمة طرفي المعادلة (٣) على Δt تكون:

$$\frac{p(t+\Delta t)-p(t)}{\Delta t}=(B-D)\,p(t)$$

بقسمة الطرفين على $p(t)$ ثم أخذ النهاية للطرفين عندما $\Delta t \to 0$

$$\frac{1}{p}\lim_{\Delta t\to 0}\frac{p(t+\Delta t)-p(t)}{\Delta t}=\lim_{\Delta t\to 0}B-D$$

$$\frac{1}{p}\frac{dp}{dt}=B-D$$

بالضرب × dt

$$\frac{dp}{p}=(B-D)dt$$

ثم التكامل بالنسبة ل t

$$\ln p=(B-D)t+c$$

$$p=e^{(B-D)t+c}$$

$$p = e^{c}.e^{(B-D)t}$$

put $m = e^{c}$

$$p = me^{(B-D)t}$$

لإيجاد ة ضع $t = o$

$$p(0) = me^{(B-D)(0)} = m$$

$$\therefore p(t) = p(0)e^{(B-D)t}$$

ملاحظات على النموذج:

١. إذا كانت $B-D < 0$ - أي أن عدد المواليد أقل من عدد الوفيات – فإن النموذج لا يتناسب مع النمو الطبيعي على المدى البعيد لأن عدد السكان سيتناقص الى الصفر و هذا ليس طبيعياً لمعظم حالات النمو:-

$$p(\infty) = me^{-\infty} = 0$$

٢. إذا كان $B-D < 0$ - عدد الوفيات أقل من عدد المواليد – فإن النموذج أيضا لا يتوافق مع النمو الطبيعي عندما تكون t كبيرة(∞) عدد السكان سيكون ∞.
٣. في هذا النموذج تم تجاهل الكوارث الطبيعية و الحروب والوبائيات وتم الاعتماد على المواليد و الوفيات فقط.

مثال:

يتناسب نمو سكان قرية مع عدد سكانها، بعد سنة واحدة كان عدد السكان ١٠٠٠ نسمة وبعد اربع سنوات أصبح ٣٠٠٠ نسمة، أوجد:

١. عدد السكان الموجودين في البداية.
٢. عدد السكان بعد ١٠ سنوات.

(تلميح: ضع $k = B-D$ ثابت)

```
#include <iostream.h>
#include <math.h>
 #include <conio.h>
 int round(double x)
{
   return int(x > 0.0 ? x + 0.5 : x - 0.5);
}
void main ()
{
int p0,p1,y1,y2,a,yn,pn;
double e;
   cout<<" number Population Year 1=";
   cin>>p0;
   cout<<"number Population Year 2=";
   cin>>p1;
   cout<<"Year 1=";
   cin>>y1;
   cout<<"Year 2=";
   cin>>y2;
   cout<<"After how many years = ";
   cin>>yn;
   e=2.718281828;
   a=y2-y1;
   double k;
   k=p1/p0;
   k=log (k)/a;
```

```
    k=pow(e,k);
     p0=round(p0/k);
    cout<<"The population status of the beginning of the year ="<<p0;
    pn=p0*pow(e,yn*k);
    cout<<endl<<" Population after  "<<yn<<" years ="<<pn;
  getch();
}
```

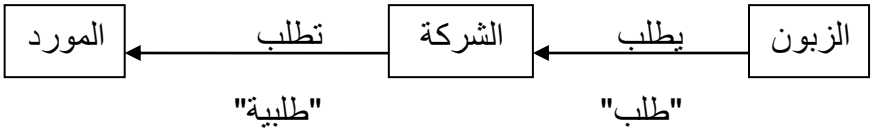
محاكاة نظام مخزن

وصف النظام:

شركة تبيع منتج واحد وتريد تقدير كمية البضاعة التي يجب أن توجد في مخزن الشركة لكل شهر من ال n شهر القادمة.

ترميز:

١. سنطلق على طلب الزبون لبضاعة من الشركة: طلب (D)demand.
٢. سنطلق على على طلب الشركة من المورد: طلبية order.



- الزمن بين طلبات الزبائن عشوائي.
- وكذلك حجم الطلبات D متغير عشوائي يتبع للتوزيع المتظم وليس له علاقة بزمن الطلب.
- في بداية الشهر تتقوم الشركة بمراجعة المخزون لديها. ثم تقرر كم وحدة يجب طلبها من المورد Supplier. إذا طلبت الشركة z وحدة فإن ذلك سيكلفها

$$k + iz$$

حيث k = تكلفة التجهيزات (الترحيل، عمال النقل،... الخ)

i = التكلفة التزايدية لكل وحدة شهرية.

- عندما يتم طلب طلبية فإن زمن وصولها من المورد ايضا متغير عشوائي يسمى lead time أو delivering lag يتوزع بين نصف شهر و شهر.
- لدى الشركة سياسة تخزين (s,S) لحديد حجم الطلبية كالآتي:

$$I \langle s \qquad z = \begin{cases} S - I \\ 0 \end{cases}$$
$$I \geq 0$$

حيث:

I = البضاعة الموجودة في المخزن في بداية الشهر.

s = أقل كمية بضاعة يجب وجودها في المخزن.

S = أكبر كمية بضاعة يجب وجودها في مخزن الشركة.

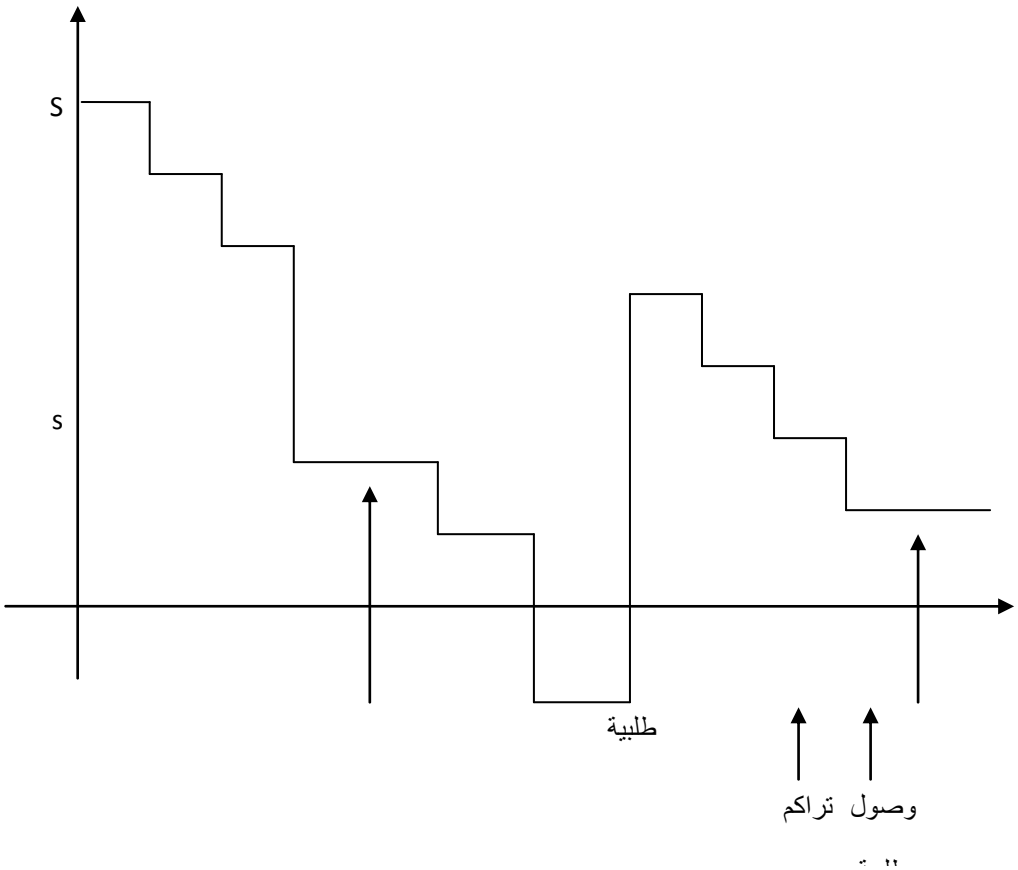
- عندما يطلب الزبون بضاعة فإنه يعطى ما يطلب اذا كانت البضاعة الموجودة في المخزن على الاقل مساوية للطلب (≤) . اذا كان الطلب كبير فسيكون هناك باقي على الشركة توفيه في المرات القادمة، وفي هذه الحالةفإن المخزون الجديد (المخزون القديم ناقص الطلب) عدد سالب ويسمى التراكم (backlog).
- توجد تكاليف أخرى على الشركة (عدا تكلفة الطلبيات) هي:
  ١. تكلفة الملكية holding cost
  ٢. تكلفة التقصير shortage cost

لحساب التكاليف أعلاه يجب أولاً تعريف القيم الثلاثة التالية:

١. $I(t)$ = محتوى المخزن في الزمن t. نلاحظ ان $I(t)$ يمكن ان يكون صفر، سالب أو موجب.

٢. $I^+(t)$ = عدد الوحدات الموجودة فعلياً في المخزن في الزمن t. $= I^+(t)$ حيث $\max\{I(t),0\}$
$I^+(t) \geq 0$.

٣. $I^-(t)$ = الكمية التي لم توفى من الطلب – التراكم – في الزمن t ، $I^-(t) = \max\{-I(t),0\}$ وأيضاً $I^-(t) \geq 0$



حساب تكلفتي الملكية والتقصير:

- في هذا النموذج افترضنا ان تكلفة الملكية للوحدة الواحدة هو h = 1$ للشهر والتي تتضمن الايجار، التأمين، الماء، الكهرباء ...الخ. وهي مطلوبة حتى في حالة $I^+(t) = 0$ وبما ان الغرض من المحاكاة هو مقارنة سياسات الطلبيات فقد تم تجاهل هذه التكلفة – تم اعطاءها ١$ للوحدة – لأنها لاتؤثر أو تتأثر بنوعية السياسة المتبعة. الآن: متوسط عدد الوحدات الموجودة في المخزن لكل شهر ل n شهر القادمة هي:

$$\overline{I^+} = \int_0^n \frac{I^+(t)}{n} dt$$

** وذلك لأنها متغير عشوائي يتبع للتوزيع المنتظم الذي دالته الاحتمالية = $\frac{1}{n-0} = \frac{1}{n}$

اذن تكلفة الملكية للشهر هي:

hcost = $\overline{I^+}$ h .....(1)

- ايضا نفرض أن تكلفة التراكم backlog للوحدة المتراكمة يساوي $\Pi = 5\$$، متوسط عدد الوحدات المتراكمة:

$$\overline{I^-} = \int_0^n \frac{I^-(t)}{n} dt$$

اذن متوسط تكلفة التراكم للشهر هي:

scost = $\overline{I^-}$ Π .....(2)

- بعد هذا يتم حساب التكلفة الكلية والتي تتكون من:

تكلفة الطلبيات + تكلفة الملكية + تكلفة التقصير

للشهر للمقارنة بين السياسات المتبعة. السياسة التي تعطي أقل تكلفة هي الأفضل.

الأحداث في النظام:

١. حدث وصول طلبية من المورد الى الشركة.

٢. حدث طلب زبون لمنتج من الشركة.
٣. نهاية المحاكاة بعد n شهر.
٤. جرد مخزن واحتمال حدوث طلبية في بداية الشهر.
الاحداث ١،٢،٤ تسبب تغير في حالة النظام.

خوارزميات الاحداث أعلاه:

١/ خوارزمية حدث وصول طلبية:

١. البداية.
٢. وصول الطلبية.
٣. اضافة حجم الطلبية الى كمية البضاعة الموجودة في المخزن.
٤. منع خروج اي طلبية.
٥. النهاية.

٢/ خوارزمية حدوث طلب بضاعة من الشركة:

١. البداية.
٢. حدوث الطلب.
٣. حساب حجم الطلب.
٤. انقاص كمية الطلب من كمية البضاعة الموجودة في المخزن.
٥. الترتيب للطلب القادم.
٦. النهاية.

٣/ خوارزمية جرد المخزن:

١. البداية.
٢. هل $I(t) \leq s$ ؟ اذا كانت الاجابة نعم اذهب الى الخطوة رقم (٣). والا اذهب الى الخطوة رقم (٦)،
٣. حساب حجم الطلبية $S - I(t)$.
٤. حساب تكلفة الطلبية.
٥. الترتيب لحدث وصول طلبية.
٦. الترتيب لحدث الجرد التالي.
٧. النهاية.

٤/ الخوارزمية العامة للنظام:

١. البداية.
٢. إدخال قيم s ، S.

٣. ادخال k ،i ،I(0)،n.
٤. t = 0.
٥. .حساب قيمة الطلبية من الشركة z بإستخدام بالقانون:

$$z = \begin{cases} S - I & I \langle s \\ 0 & I \geq 0 \end{cases}$$

٦. كمية البضاعة الموجودة في المخزن بعد وصول الطلبية هو:

$$I(t) = I(t) + z$$

٧. اذا طلب الزبون بضاعة قدرها D فإن:

$$I(t) = I(t) + D$$

$$I^{+}(t) = \max\{I(t),0\}$$

$$I^{-}(t) = \max\{-I(t),0\}$$

٨. حساب تكلفة الطلبية للشهر t كالتالي:

$$DCost = k + iz$$

٩. حساب تكلفة الملكية للشهر t .
١٠. حساب تكلفة التقصير للشهر t.
١١. t = t+1 .
١٢. تكرار الخطوات ٥ – ١١ حتى حدوث شرط نهاية المحاكاة t = n.
١٣. حساب متوسط التكاليف الثلاثة ل n من الشهور.
١٤. حساب التكلفة الكلية.
١٥. تكرار الخوارزمية لسياسات مختلفة ومقارنتها.

ملحوظة: يتم توليد المتغير D كمتغير عشوائي.

مثال:

اذا كانت الوحدات الموجودة في مخزن الشركة في بداية المحاكاة هي I(0) = 60 ، وانه لاتوجد طلبيات متأخرة بين الشركة والمورد. حاكى نظام جرد الخزن ل ١٢٠ شهر بإستخدام التكلفة التكلية للمقارنة بين السياسات ال ٩ التالية:

	1	2	3	4	5	6	7	8	9
s	20	20	20	20	40	40	40	60	60
S	40	60	100	60	60	80	100	80	100

الحل:

السياسة الاولى $s=20$, $S=40$

نفرض ان $k = 20, i = 2,$

عند $t = 0$: $I(0) = 60$

$Z = 0$; $I(t) > s$

$DCost = 20 + 0(2) = 20$

نفرض ان $D = 30$ اذن

$I(0) = 60 - 30 = 30$

$I^{+}(0) = \max\{30,0\} = 30$ …… $hcost = 30$

$I^{-}(0) = \max\{-30,0\} = 0$ ……. $scost = 0$

عند $t = 1$

$Z = 0$; $I(0) > 0$

$Dcost = 20 + 2(0) = 20$

نفرض ان $D = 20$ اذن

$I(1) = 30 - 20 = 10$

$I^{+}(1) = \max\{10,0\} = 10$ ….. $hcost = 10$

$I^{-}(0) = \max\{-10,0\} = 0$ ……. $scost = 0$

عند $t = 2$

$Z = 40 - 10 = 30$; $I(1) < s$

Dcost = 20 + 2(30) = 20 +60 = 80

I(t) = 10 +30 = 40

نفرض ان D = 50 اذن

I(1) = 40-50 = -10

$I^{+}(1)$ = max{-10,0} = 0 …….hcost = 0

$I^{-}(0)$ = max{-(-10),0} = 10 ….. scost =5(10) =50

عند t = 3

Z = 40-(-10) = 50 ; I(2) < s

Dcost = 20+2(50) = 20+100 = 120

I(t) = -10 +50 = 40

نفرض ان D = 40 اذن

I(1) = 40-40 = 0

$I^{+}(1)$ = max{0,0} = 0 ….. hcost = 0

$I^{-}(0)$ = max{0,0} = 0 ….. scost = 0

** يتم حساب باقي التكاليف بنفس الطريقة.

متوسط تكاليف الطلبيات = $\frac{20+20+80+120}{4} = \frac{240}{4} = 60$

متوسط تكاليف الملكية: $\frac{30+10+0+0}{4} = \frac{50}{4} = 12.5$

متوسط تكاليف التقصير = $\frac{0+0+50+0}{4} = \frac{50}{4} = 12.4$

التكلفة الكلية = ٦٠ + ١٢,٥ + ١٢,٥ = ٦٠ + ٢٥ = ٧٥

```
#include <conio.h>
#include <iostream.h>
void main()
{
int i,is,z,dcost[9],hcost[9],scost[9],s1[9],s2[9],t,k,d;
double hc,sc,dc;

for(t=1;t<=9;t++)
{
cout<<"enter Less the amount of goods must exist in the store s=";
cin>>s1[t];
}
for(t=1;t<=9;t++)
{
cout<<"enter The largest amount of goods should their presence in the store S=";
cin>>s2[t];
}



i=2;
is=60;
k=20;
for(t=0;t<=3;t++)
{
if(is>s1[1])
{
```

```
z=0;
dcost[t]=k+i*z;
cout<<"enter demand d=";
cin>>d;
is=is-d;
if(is>0)
hcost[t]=is;
else
hcost[t]=0;
if((-1*is)>0)
scost[t]=is;
else
scost[t]=0;
}
else {
z=s2[1]-is;
dcost[t]=k+i*z;
is=is+z;
cout<<"enter demand d=";
cin>>d;
is=is-d;
if(is>0)
hcost[t]=is;
else
hcost[t]=0;
if((-1*is)>0)
scost[t]=-1 * is*5;
```

```
else
scost[t]=0;
}
}

for(t=0;t<=3;t++)
{
hc=hc+hcost[t];
cout<<scost[t]<<endl;
sc=sc+scost[t];
dc=dc+dcost[t];
}
dc=dc/4;
sc=sc/4;
hc=hc/4;
 cout<<"the avrg order dcost="<<dc<<endl;
 cout<<"the avrg holding cost hcost="<<hc<<endl;
 cout<<"the avrg shortage cost scost="<<sc<<endl;
 cout<<"the sum all cost ="<<(hc+dc+sc);

 getch();
}
```

توليد الأرقام العشوائية على الحاسوب

Random Numbers Generating on Computer

تستخدم الارقام العشوائية كمدخلات لنموذج المحاكاة، لأنها تكون شبيهة بالبيانات الحقيقية للنظام وهي دقيقة في وصف الأنظمة المعقدة.

مزايا الأرقام العشوائية:

١. الرقم العشوائي يجب ان يكون اقرب الى ان يتبع التوزيع المنتظم.
٢. البرنامج الذي يولد الارقام العشوائية يجب ان يكون سريع.
٣. البرنامج الذي يولد الارقام العشوائية يجب ان يكون صغير.
٤. الخوارزمية التي تولد الارقام العشوائية يجب ان تكون دورتها طويلة حتى لاتكرر الارقام.
٥. يجب تجنب التكرار وتوليد الرقم صفر.

اسلوب فيبوناتشي:

يولد ارقام لا تتوافق مع شرط الارقام العشوائية ويأخذ الصيغة التالية:

$$x_{r+1} = (x_r + x_{r-1}) \bmod m$$

حيث ينتج هذا الاسلوب سلسلة من الارقام العشوائية ذات دورة تكرارية طولها اكبر من m.

```
#include <conio.h>
#include <iostream.h>
void main()
{
int m,i;
int x[10];
cout<<"enter m=";
cin>>m;
```

```
cout<<"enter x0=";
cin>>x[0];
 cout<<"enter x1=";
cin>>x[1];
for(i=1;i<=9;i++
)
{
x[i+1]=(x[i]+x[i-1])%m;
}
  for(i=2;i<=9;i++)
{
cout<<x[i]<<",";
}
getch();
}
```

اسلوب التطابق:

هو الاسلوب الافضل لأنه يولد أرقام عشوائية تتبع للتوزيع المنتظم كما يتميز بدورات طويلة.
وينقسم الى ثلاثة انواع:

١. اسلوب التطابق المختلط.
٢. اسلوب التطابق المضاعف.
٣. اسلوب التطابق الاضافي.

التطابق المختلط:

يستخدم لتوليد الرقم العشوائي اعتماداً على الرقم العشوائي السابق له مباشرة، بإفتراض رقم عشوائي بذرة (seed) ويحسب العدد رقم x_{n+1} من العدد x_n بالقانون:

$$x_{n+1} = (ax_n + b) \bmod m \ldots\ldots\ldots\ldots\ldots\ldots (*)$$

حيث a,b,m اعداد صحيحة.

m تمثل عدد القيم المختلفة للأرقام العشوائية.

x_{n+1} تمثل الباقي عند قسمة ax_n+b على m.

مثال:

اذا كانت $x_0=2$,$a = 3$, $b = 1$, $m = 16$ استخدم التطابق المختلط لتوليد ارقم عشوائية بإستخدام القيم المعطاة.

الحل:

بالتعويض في المعادلة * نحصل على:

$X_{0+1} = X_1 = (3*X_0 + 1) \bmod 16 = (3*2 + 1) \bmod 16 = 7$

$X_{1+1} = X_2 = (3*X_1 + 1) \bmod 16 = (3*7 + 1) \bmod 16 = 22 \bmod 16 = 6$

$X_{2+1} = X_3 = (3*X_2 + 1) \bmod 16 = (3*6 + 1) \bmod 16 = 19 \bmod 16 = 3$

$X_{3+1} = X_4 = (3*X_3 + 1) \bmod 16 = (3*3 + 1) \bmod 16 = 10$

$X_{4+1} = X_5 = (3*X_4 + 1) \bmod 16 = (3*10 + 1) \bmod 16 = 31 \bmod 16 = 15$

$X_{5+1} = X_6 = (3*X_5 + 1) \bmod 16 = (3*15 + 1) \bmod 16 = 46 \bmod 16 = 14$

$X_{6+1} = X_7 = (3*X_6 + 1) \bmod 16 = (3*14 + 1) \bmod 16 = 43 \bmod 16 = 11$

$X_{7+1} = X_8 = (3*X_7 + 1) \bmod 16 = (3*11 + 1) \bmod 16 = 34 \bmod 16 = 2$

$X_{8+1} = X_9 = 7$

إذن التتابع هو:

2,7,6,3,10,15,14,11,2

```
#include <conio.h>
#include <iostream.h>
void main()
{
int a,b,m;
int x[9];
cout<<"enter a=";
cin>>a;
cout<<"enter b=";
cin>>b;
cout<<"enter m=";
cin>>m;
cout<<"enter x0=";
cin>>x[0];
for(int i=0;i<=8;i++)
{
x[i+1]=(a*x[i]+b)%m;
}
  for(int i=0;i<=9;i++)
{
cout<<x[i]<<",";
}
getch();
}
```

التطابق المضاعف:

عندما تكون b = 0 في التطابق المختلط يسمى التطابق المضاعف وتتحول الصيغة الى:

$$x_{n+1} = (ax_n) \bmod m$$

```
#include <iostream.h>
void main()
{
int a,b,m;
int x[9];
cout<<"enter a=";
cin>>a;
cout<<"enter b=";
cin>>b;
cout<<"enter m=";
cin>>m;
cout<<"enter x0=";
cin>>x[0];
for(int i=0;i<=8;i++)
{
x[i+1]=(a*x[i]+b)%m;
}
  for(int i=0;i<=9;i++)
{
cout<<x[i]<<",";
}
getch();
}
```

التطابق المضاف:

ويأخذ الصيغة التالية:

$$x_{n+1} = (ax^2{}_n + bx_{n-1} + c) \bmod m$$

حيث a,b,c,m أعداد صحيحة.

```
#include <conio.h>
#include <iostream.h>
#include <math.h>
void main()
{
int a,b,c,x[9],z,m;
cout<<"enter a=";
cin>>a;
cout<<"enter b=";
cin>>b;
cout<<"enter c=";
cin>>c;
cout<<"enter m=";
cin>>m;
cout<<"enter xn=";
cin>>x[1];
cout<<"enter x(n-1)=";
cin>>x[0];

for(int i=1;i<=9;i++)
{
z= (a*pow(x[i],2))+(b*x[i-1])+c;
x[i+1]=z%m;
}
```

```
  for(int   i=0;i<=9;i++)
{
cout<<x[i]<<",";
}
getch();
}
```