코딩, 수학과 콜라보레이션

코딩수학 1

최적의 소방서 위치정하기

행복도시 만들기 프로젝트

시장님, 우리 신도시에는 아직 소방서가 없습니다. 건물에 불을 끄고 위험에 처한 사람들을 구조하기 위해 소방서와 119 구조대가 필요합니다. 그런데 어디에 소방서를 세우는게 좋을까요? 최대한 사람들이 살고 있는 건물과 가장 가까운 곳에 지어야겠죠!

김성기 · 최용호 · 김지원 · 김준석 공저

이모션북스

머리말

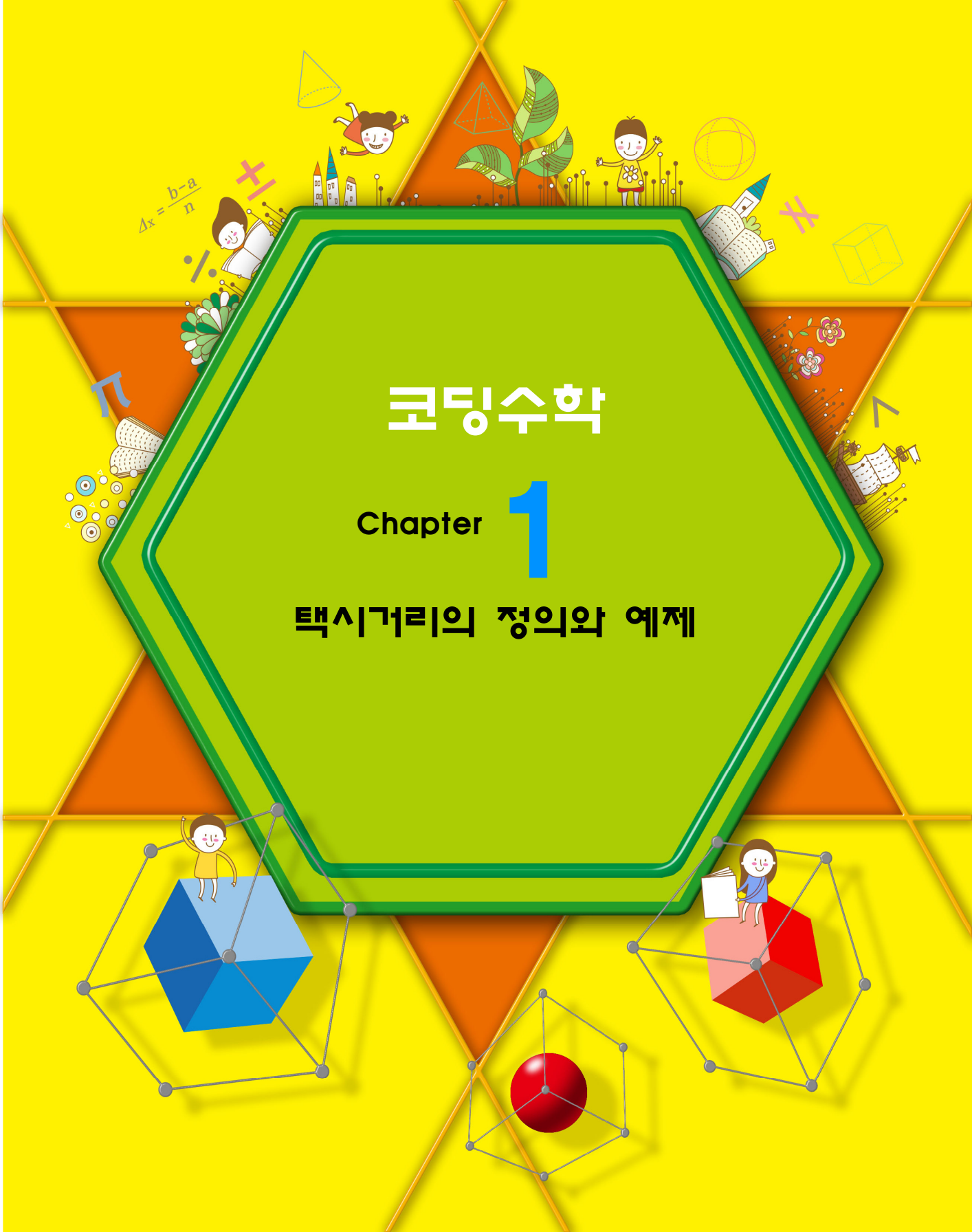
본 저서에서 코딩수학이란 수학과 프로그램 코드를 활용하여 생활 속의 다양한 문제에 대한 답을 구하는 것으로 정의한다. 수학의 엄밀함과 컴퓨터 프로그램의 대용량 계산을 빠른 시간 내에 수행 할 수 있는 능력을 융합하면 시너지 효과를 낼 수 있다. 학생들이 꼭 알았으면 좋을 것 같은 내용을 선별하였다.

차 례

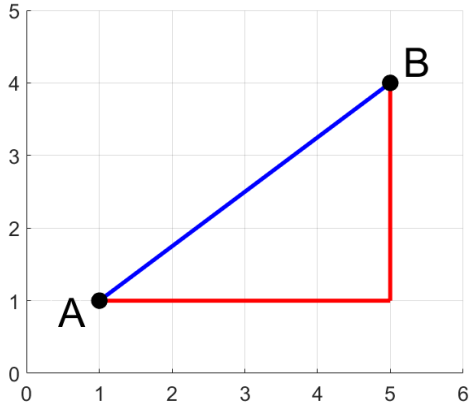
Contents

목표: 한 마을의 모든 집으로부터의 거리의 합이 최소가 되도록 하는 최적의 소방서 위치를 컴퓨터 코딩으로 구하기

프로그램의 특성상 업그레이드 될 경우에는 프로그램 설치 및 명령문이 변경될 경우도 있습니다. 책의 내용에 대해서 질문이나 조언이 있는 경우에 블로그(http://blog.naver.com/cfdkim)에 문의를 해주시면 감사하겠습니다.


코딩수학
Chapter 1
택시거리의 정의와 예제

택시거리(taxicab distance)란 무엇인가?



평면상의 두 점을 각각 $\mathrm{A}(x_1, y_1)$와 $\mathrm{B}(x_2, y_2)$라고 하자. 이때 일반적인 두 점 사이의 최단거리는

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

이다. 하지만 택시거리(taxicab distance)는 도로를 통해서만 갈 수 있다는 제약조건 때문에 다음과 같이 정의된다.

$$d_{taxi} = |x_1 - x_2| + |y_1 - y_2|$$

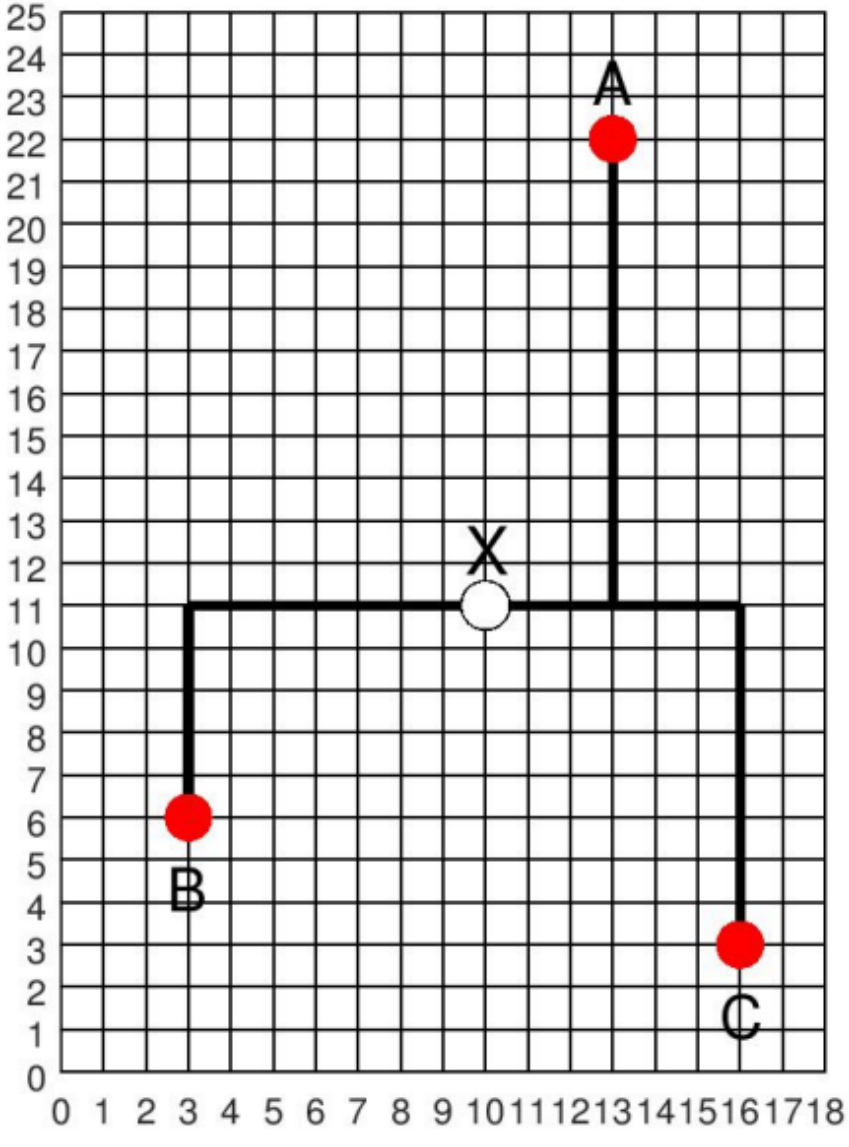
예를 들어 $A(1,1)$와 $B(5,4)$ 사이의 최단거리 d는 공식에 따라

$$d = \sqrt{(5-1)^2 + (4-1)^2} = \sqrt{4^2 + 3^2} = \sqrt{25} = 5$$

이다. 그림 상에서 파란색 선이다. 그리고 택시거리 d_{taxi}는 공식에

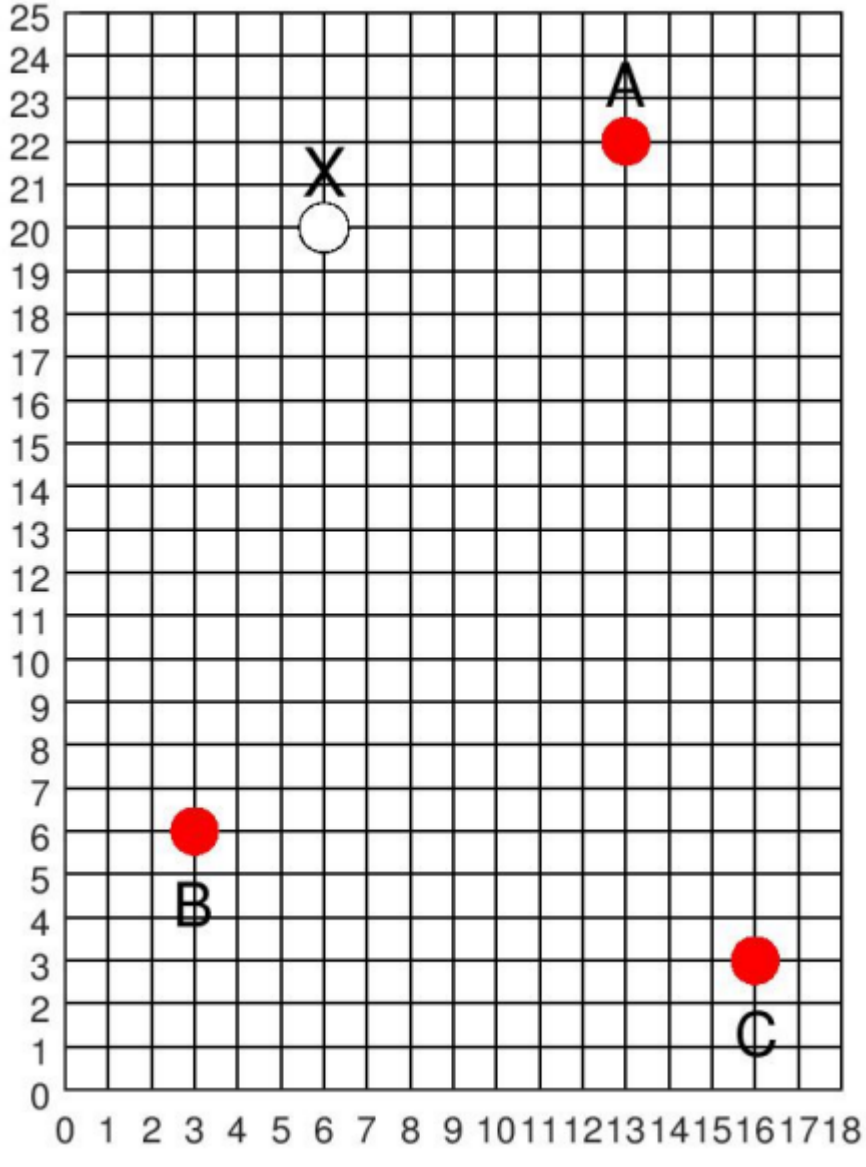
함께 택시거리를 계산해 보자

X에서 A, B, C까지의 거리를 구하고, 거리의 합을 구해보자.



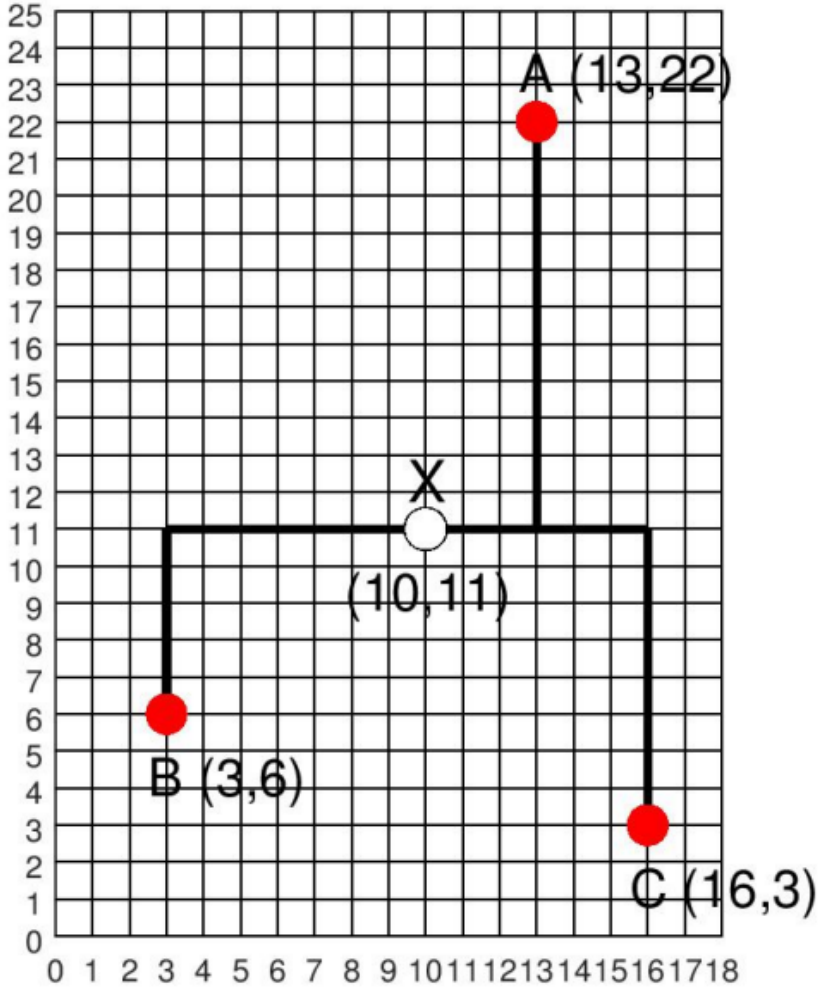
직접 택시거리를 계산해 보자

X에서 A, B, C까지의 거리를 구하고, 거리의 합을 구해보자.



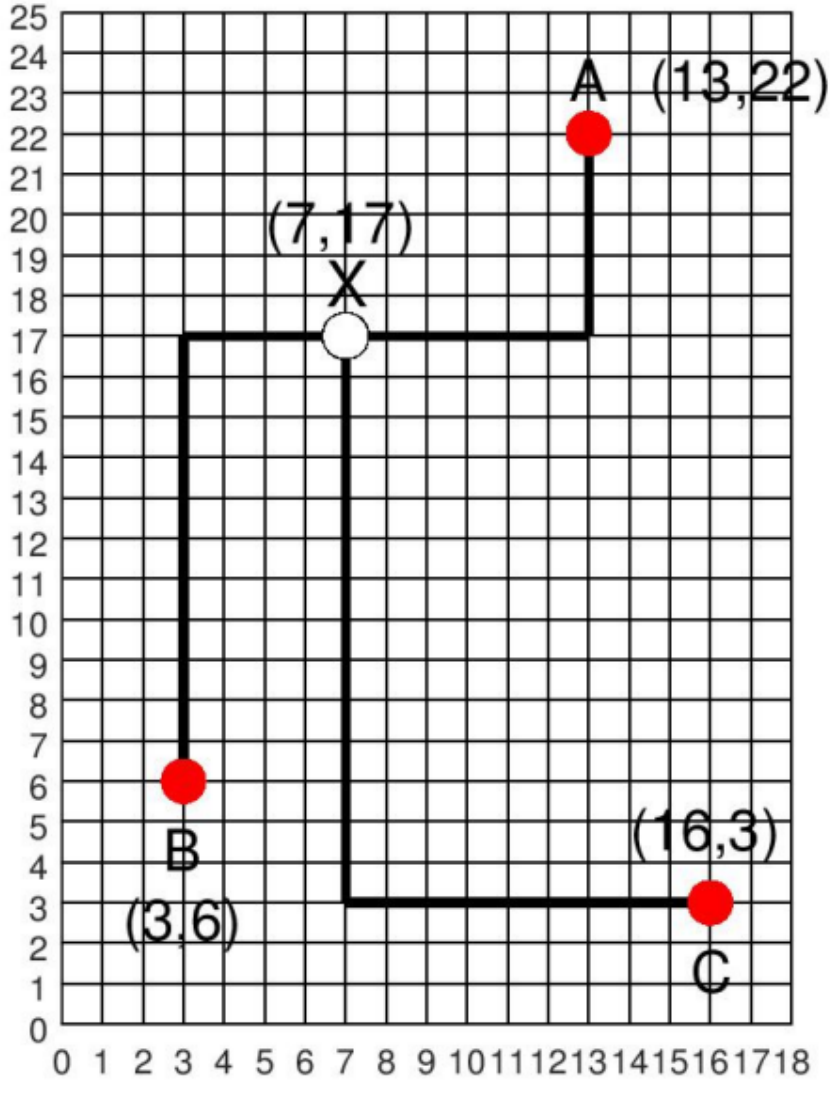
함께 좌표 개념 이해 및 이를 이용한 택시거리 계산을 해보자

좌표의 개념을 이해하고 좌표를 이용하여 두 점 사이의 거리를 구하고, 거리의 합을 계산하여 보자.



직접 좌표를 이용한 택시거리 계산을 해보자

좌표의 개념을 이해하고, 좌표를 이용하여 두 점 사이의 거리를 구하고, 거리의 합을 계산하여 보자.



X에서 A점까지의 거리는?

A점의 x좌표 13, X점의 x좌표 a => $|13-a|$

A점의 y좌표 22, X점의 y좌표 b => $|22-b|$

즉, $|13-a|+|22-b|$

X에서 B점까지의 거리는?

B점의 x좌표 3, X점의 x좌표 a => $|3-a|$

B점의 y좌표 6, X점의 y좌표 b => $|6-b|$

즉, $|3-a|+|6-b|$

X에서 C점까지의 거리는?

C점의 x좌표 16, X점의 x좌표 a => $|16-a|$

C점의 y좌표 3, X점의 y좌표 b => $|3-b|$

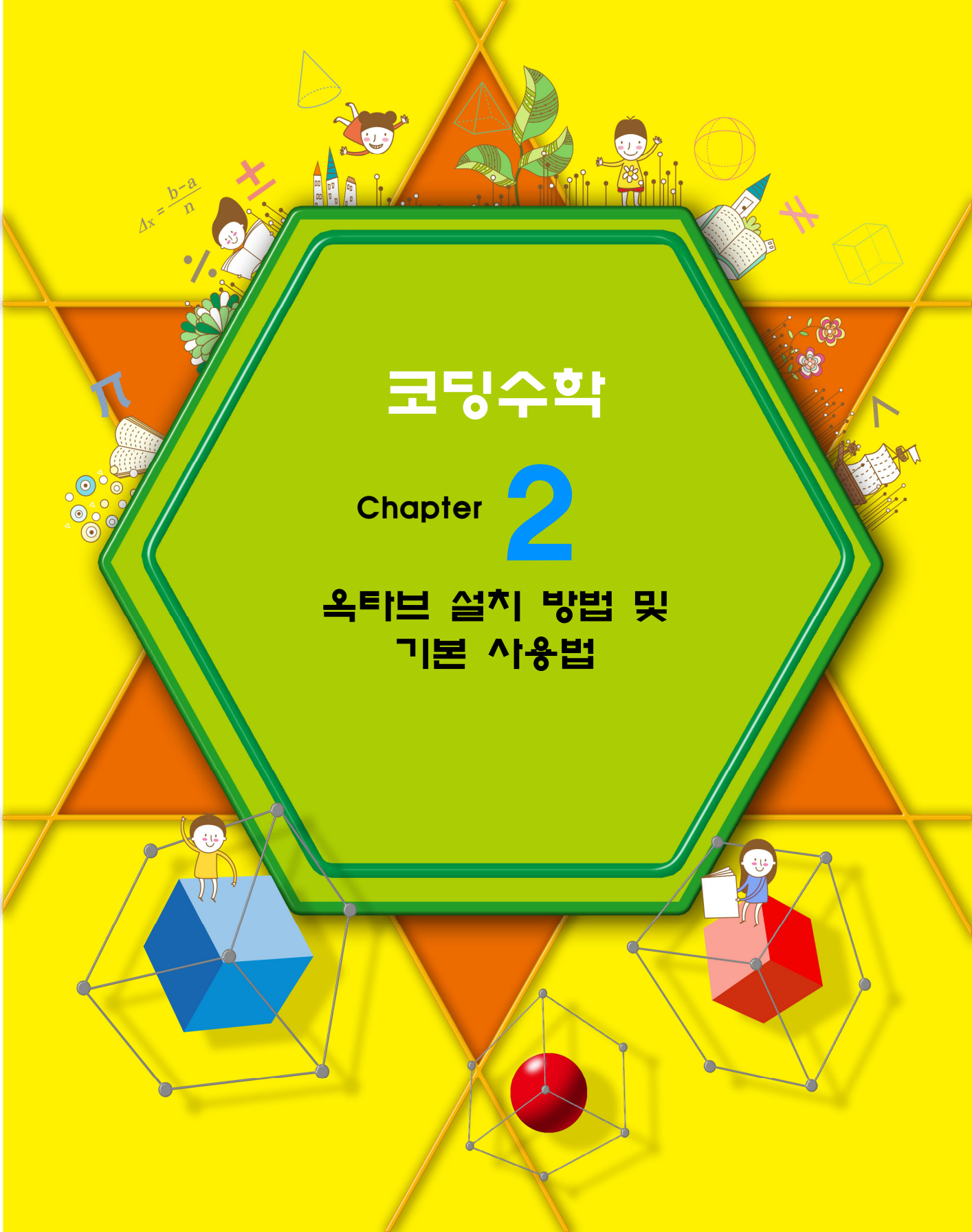
즉, $|16-a|+|3-b|$

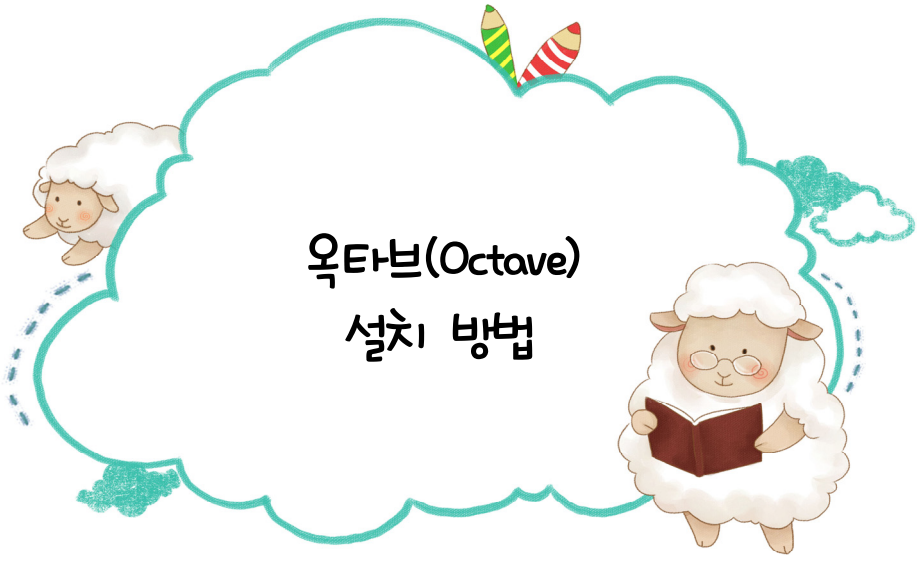
거리의 합은?

$|13-a|+|22-b|+|3-a|+|6-b|+|16-a|+|3-b|$

좀 더 쉬운 방법은 없을까?

실제로 계산을 하면 아마도 시간도 오래 걸리고 많은 인내력을 필요로 할 것이다. 좀 더 많은 점들이 있으면 손으로 계산하기가 더욱 힘들 것이다. 옥타브(Octave)라는 무료 프로그램을 이용하여 코드를 작성하고 답을 구해보자.


코딩수학
Chapter 2
옥타브 설치 방법 및
기본 사용법



프로그램을 다운받아보자

1. 프로그램 다운로드

1.1 옥타브(Octave) 홈페이지를
https://www.gnu.org/software/octave/
접속하여 Download버튼을 클릭한다.

1.2 Download를 클릭하면 기본 화면이 GNU/Linux로 세팅이 되어있다.

Install

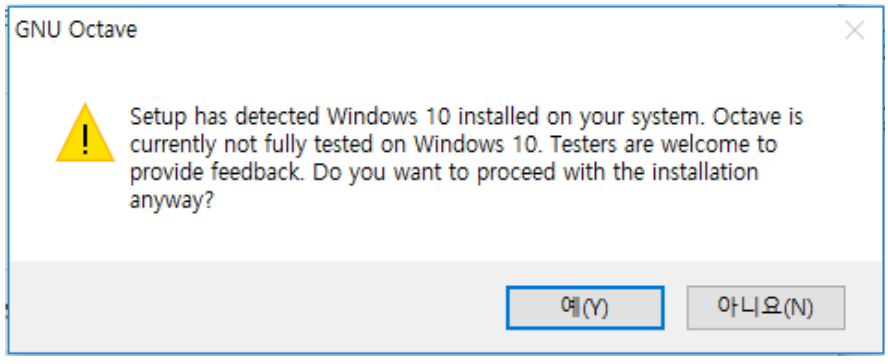
Source | GNU/Linux | macOS | BSD | Windows

Windows binaries with corresponding source code can be downloaded from https://ftp.gnu.org/gnu/octave/windows/.

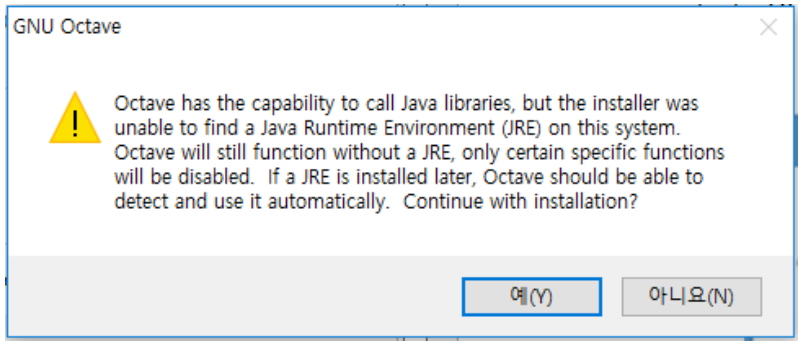
Index of /gnu/octave/windows

Name	Last modified	Size	Description
Parent Directory		-	
octave-4.2.1-w64.zip.sig	2017-02-24 09:00	95	
octave-4.2.1-w64.zip	2017-02-24 09:00	378M	
octave-4.2.1-w64-installer.exe.sig	2017-02-24 08:51	95	
octave-4.2.1-w64-installer.exe	2017-02-24 08:51	184M	
octave-4.2.1-w32.zip.sig	2017-02-24 08:46	95	
octave-4.2.1-w32.zip	2017-02-24 08:46	280M	
octave-4.2.1-w32-installer.exe.sig	2017-02-24 08:40	95	
octave-4.2.1-w32-installer.exe	2017-02-24 08:40	170M	

* 내 컴퓨터가 32비트인지 64비트인지 확인하는 방법
  내 컴퓨터 아이콘을 마우스 오른쪽 클릭하여 속성으로 들어간 다음 시스템 종류를 확인하면 된다.

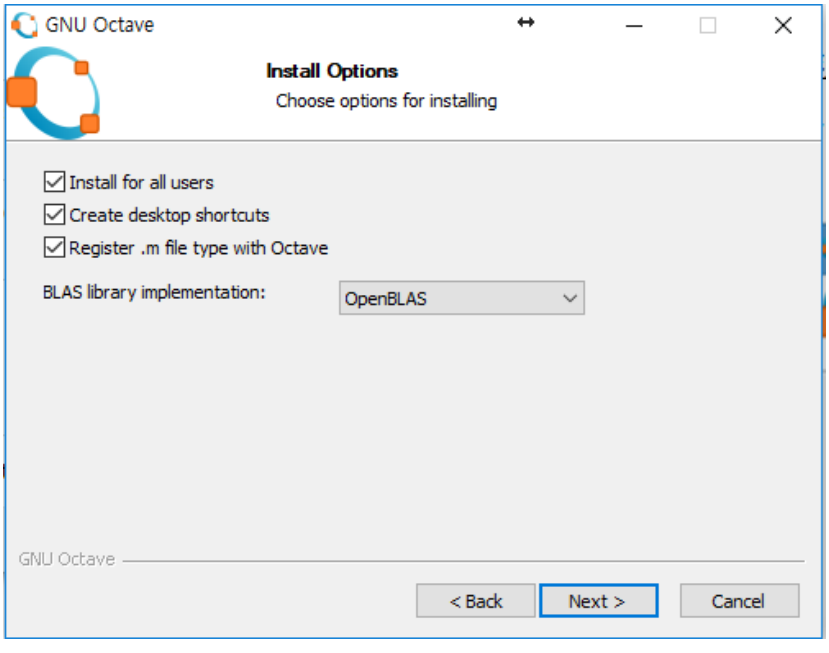


다음 경고 메시지는 java Runtime Environment가 기존에 설치되지 않았다는 것이다. '예(Y)'를 클릭하여 다음 단계로 넘어가자.

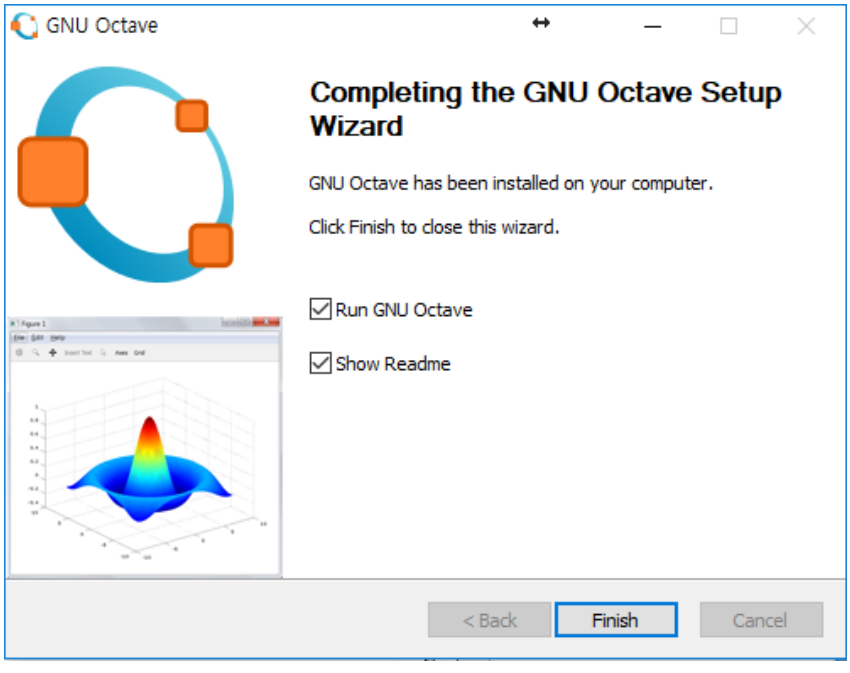


2.3 이제 본격적으로 Octave 설치가 된다. 'Next >'를 클릭하여 다음 단계로 넘어가자.

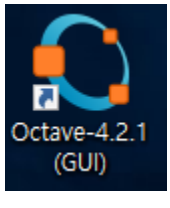
2.5 프로그램 설치에 관한 옵션선택이다. 기본 값으로 두고, ‘Next >’를 클릭하여 다음 단계로 넘어가자.



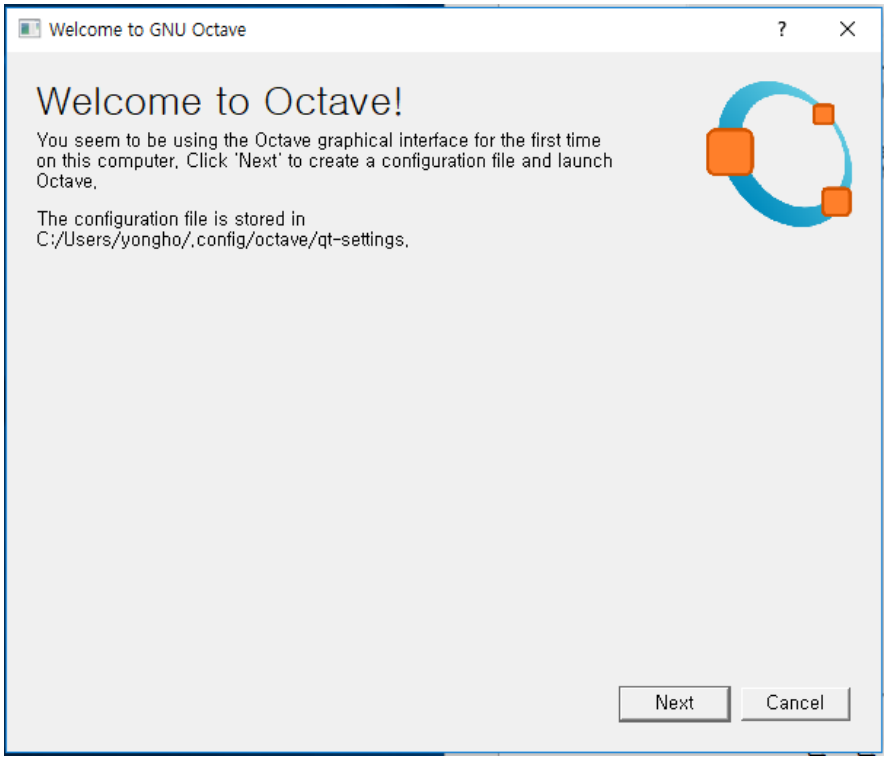
2.7 다음과 같은 화면이 나올 경우, 정상적으로 설치가 완료된 것이다. “Finish”를 클릭하여 프로그램 설치를 종료하자.



2.8 프로그램 설치를 마치면 Octave 프로그램이 실행되나 종료하고 다시 시작하자. 바탕화면을 보면 다음과 같은 Octave GUI 아이콘이 있다. 더블클릭하여 프로그램을 실행하자.

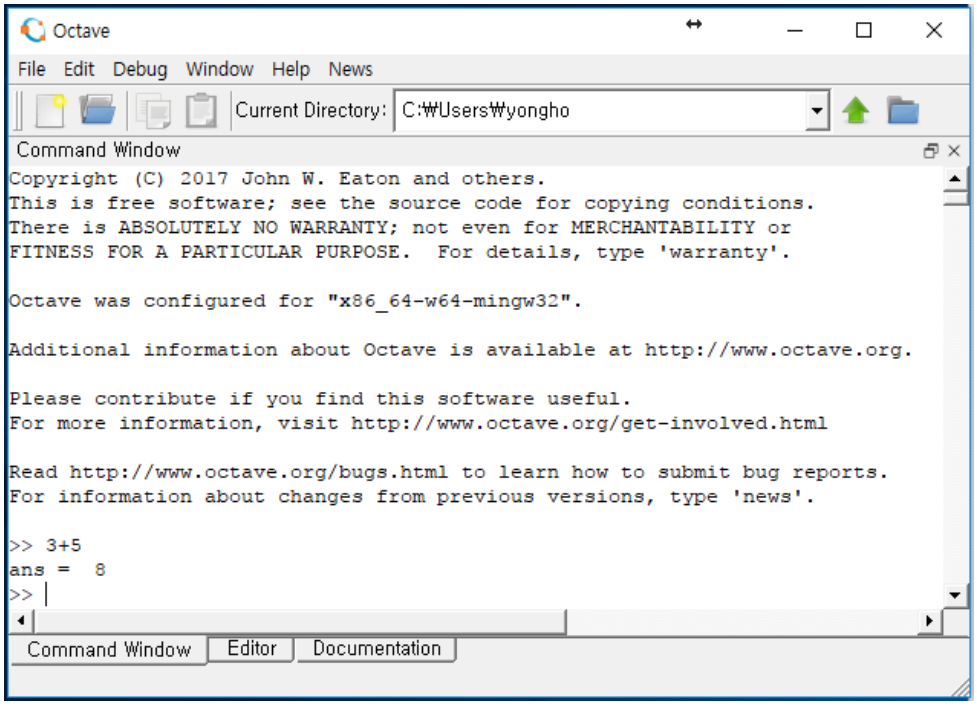


'Next >'를 클릭하여 다음 단계로 넘어가자.

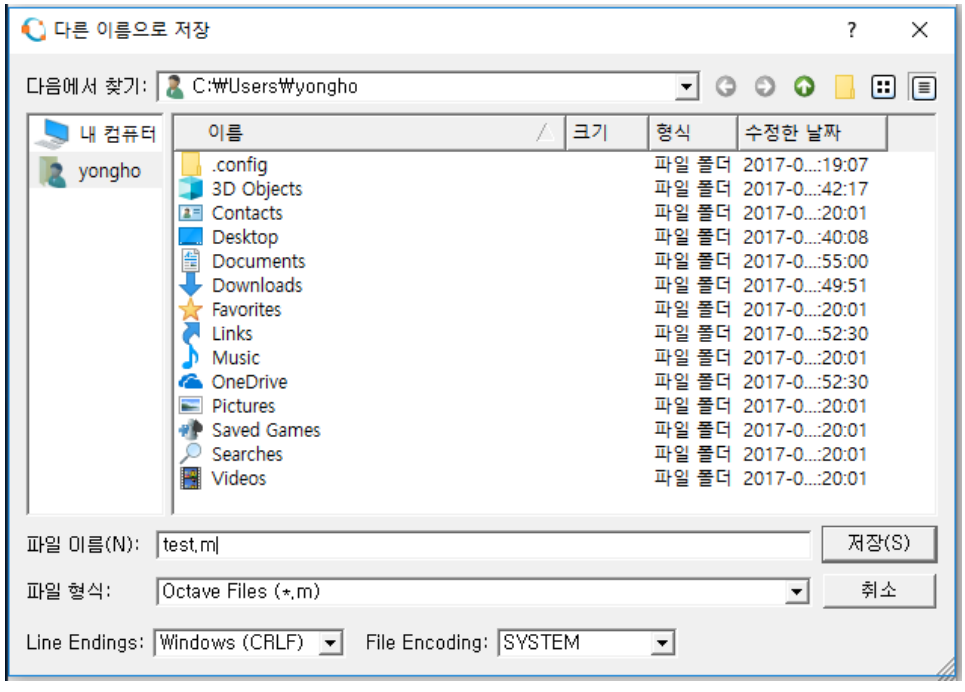


'Finish'를 클릭하여 기본설정을 완료한다.

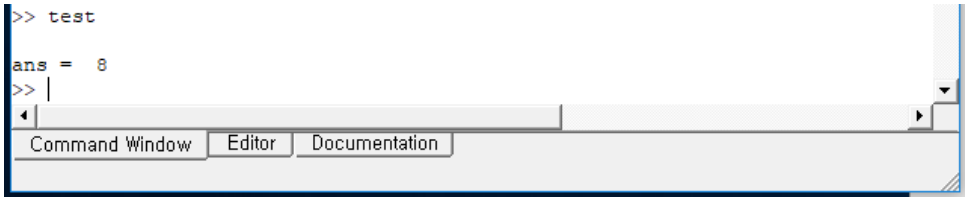
3+5를 입력하고 Enter 키를 누르면 ans = 8 이라는 결과를 얻는다.



그러나 보통 명령문이 많은 프로그램을 작성하게 된다. 프로그램의 아랫부분에 'Editor' 탭을 클릭한다.



이제 결과를 Command Window창에서 확인해보면 다음과 같이 나온다.



이 책에서 사용하는 옥타브 문법

코딩수학에서 사용된 문법

1. 반복문

조건에 따라 특정 부분의 명령문을 여러 번 수행하고자 할 때 사용한다

- 문법 -

```
for 반복변수 = 초기값 : 최종값
    명령문 1
    명령문 2
     ⋮
end
```

초기값이 1씩 증가하면서 최종 값보다 작거나 같을 때까지 for ~ end 안의 명령문을 반복적으로 실행한다.

코드설명

```
clear
% 메모리 초기화
s=0;
% 합에 대한 변수 0 으로 세팅
for i=1:10
   i
   s=s+i;
% i 의 값은 루프를 반복할 때 마다 10이 될 때까지
1씩 증가함
% s에 i 의 값을 더해줌
end
s
% 1부터 10까지의 합을 출력
```

코드명: TestIf.m

```
clear
s=100;
x=50;
if x<s
   s=x;
end
s
```

코드설명

```
clear
% 메모리 초기화
s=100;
x=50;
% s 값은 100, x 값은 50으로 설정
if x<s
% 만약 x의 값이 s 값보다 작다면
   s=x;
% s의 값에 x 값을 복사한다.
end
s
% s 값을 출력
```

코드설명

```
clear; clf;
% 변수 및 그림 초기화
nx=18; ny=25;
for j=1:ny
% i의 값을 1부터 1씩 증가하면서 ny값(25)보다 작거나
같을 때까지 반복
 line([0 nx], [j j])
% line함수를 호출하여 가로선을 하나씩 추가
end
axis image
% 화면 비율 조정
```

4. 세로선 그리기

코드명: ColumnLine.m

```
clear; clf;
nx=18; ny=25;
for i=1:nx
 line([i i], [0 ny])
end
axis image
```

코드설명

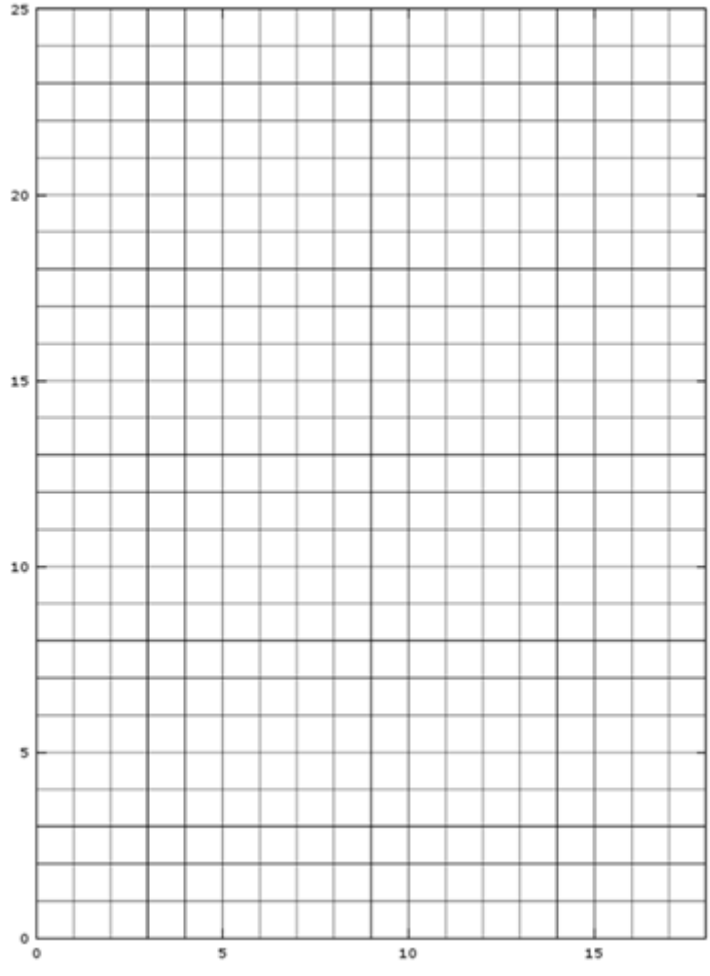
```
clear; clf;
% 변수 및 그림 초기화
nx=18; ny=25;
for i=1:nx
% i의 값을 1부터 1씩 증가하면서 nx값(18)보다 작거나 같을 때까지 반복
 line([i i], [0 ny])
% line함수를 호출하여 세로선을 하나씩 추가
end
axis image
% 화면 비율 조정
```

5. 격자선 그리기 (가로선 +세로선)

코드명: RowColumnLine.m

```
clear; clf;
nx=18; ny=25;
for j=1:ny
 line([0 nx], [j j])
end
for i=1:nx
 line([i i], [0 ny])
end
axis image
```

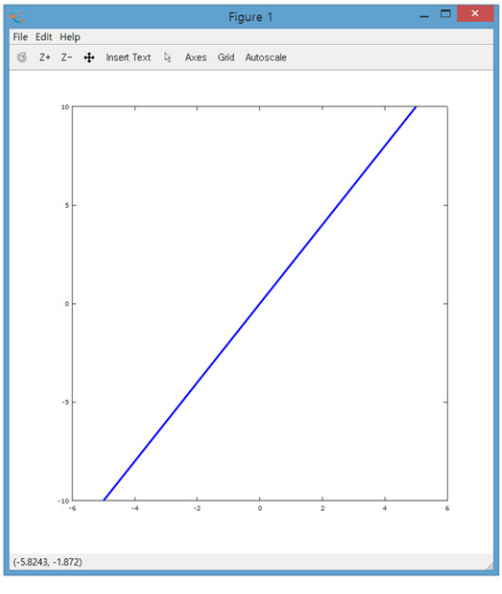
결과



코드설명

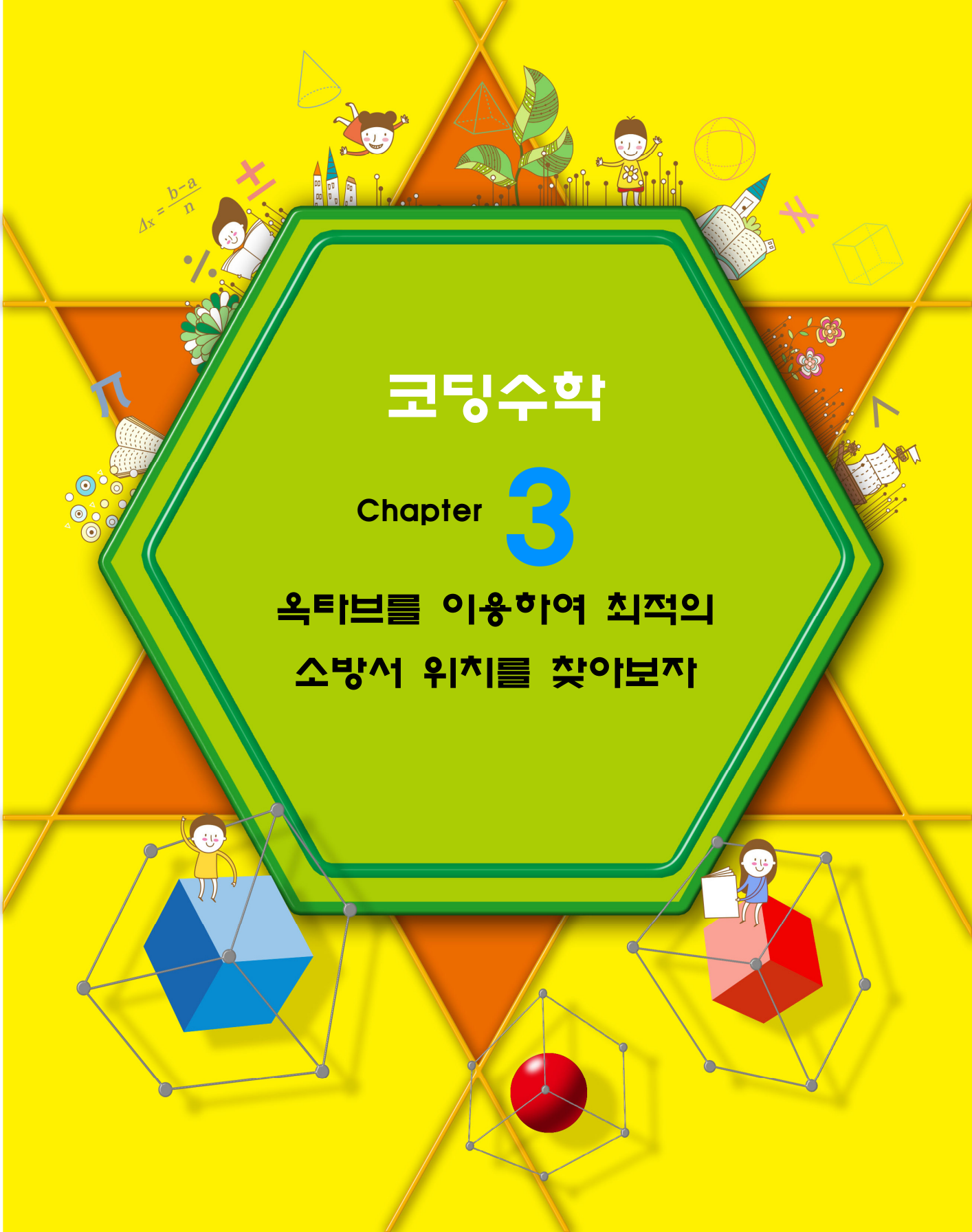
```
clear; clf;
% 변수 및 그림 초기화
x = [-5 -4 -3 -2 -1 0 1 2 3 4 5];
% x축 데이터 생성
y = [-10 -8 -6 -4 -2 0 2 4 6 8 10];
% y축 데이터 생성
plot(x,y,'b-','linewidth',3)
% 그래프 출력, 선: 파란색, 폭: 3
```

그래프 결과



코드설명

```
clear; clf;
% 변수 및 그림 초기화
x = [-5 -4 -3 -2 -1 0 1 2 3 4 5];
% x축 데이터 생성
y1 = [-10 -8 -6 -4 -2 0 2 4 6 8 10];
% 첫 번째 y축 데이터 생성
y2 = [-5 -9 -5 -3 -1 3 1 2 5 9 7];
% 두 번째 y축 데이터 생성
plot(x,y1,'b-','linewidth',3)
% 첫 번째 데이터는 파란색으로 플롯
hold on
% 첫 번째 y축 데이터 플롯 후 그림 잡아두기
plot(x,y2,'r-','linewidth',3)
% 두 번째 y축 데이터는 빨강색으로 플롯
```



코딩수학

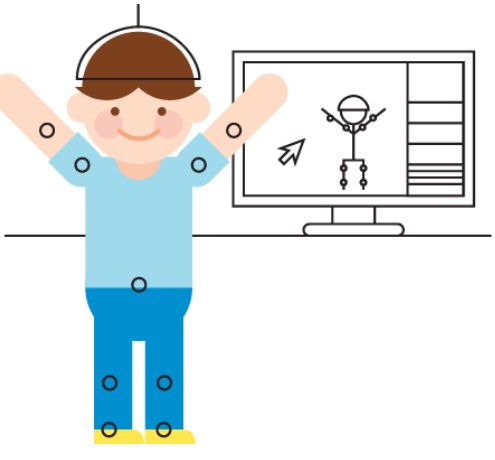
Chapter 3

옥타브를 이용하여 최적의 소방서 위치를 찾아보자

임의의 점들과 특정 점과의 택시거리 구하기

2차원 랜덤 좌표의 점들과 특정위치와의 거리의 합 구하기

x, y의 각 범위를 설정하고 x, y축의 정수 값에 대해 격자를 생성한다. randi를 사용하여 주어진 범위 내에서 x값 30개 y값 30개의 정수 값들을 무작위로 뽑는다. 그 점들을 plot을 사용하여 그림으로 표현해보고, 최소 거리에 해당할 점을 스스로 생각해본다. 그 점을 코드(i,j)에 넣고 만든 30개의 점들과 (i,j)의 실제 택시거리를 출력해본다.



코드설명

```
clear; clf; hold on
%변수 및 그림 초기화
nx=18; ny=25;
for j=0:ny
  line([0 nx],[j j])
end
% y=0, y=1, ... y=25까지 x축에 평행한 가로격자를 만든다.
for i=0:nx
  line([i i],[0 ny])
end
% x=0, x=1, ... x=18까지 y축에 평행한 가로격자를 만든다.
NoH=30;
% 주어진 격자내의 점 중에서 무작위로 30개의 점을 뽑는다.
x=randi([0 nx],NoH,1);
% 0,1,2, ... 18중 30개의 x 값을 뽑는다.
y=randi([0 ny],NoH,1);
% 0,1,2, ... 25중 30개의 y 값을 뽑는다.
```

명령문 창 결과

```
〉〉 FindDistance

d =   355
```

임의의 값들(벡터) 중에서 최솟값 찾기

30개의 임의의 데이터에서 최솟값 구하기

코드명: minA.m

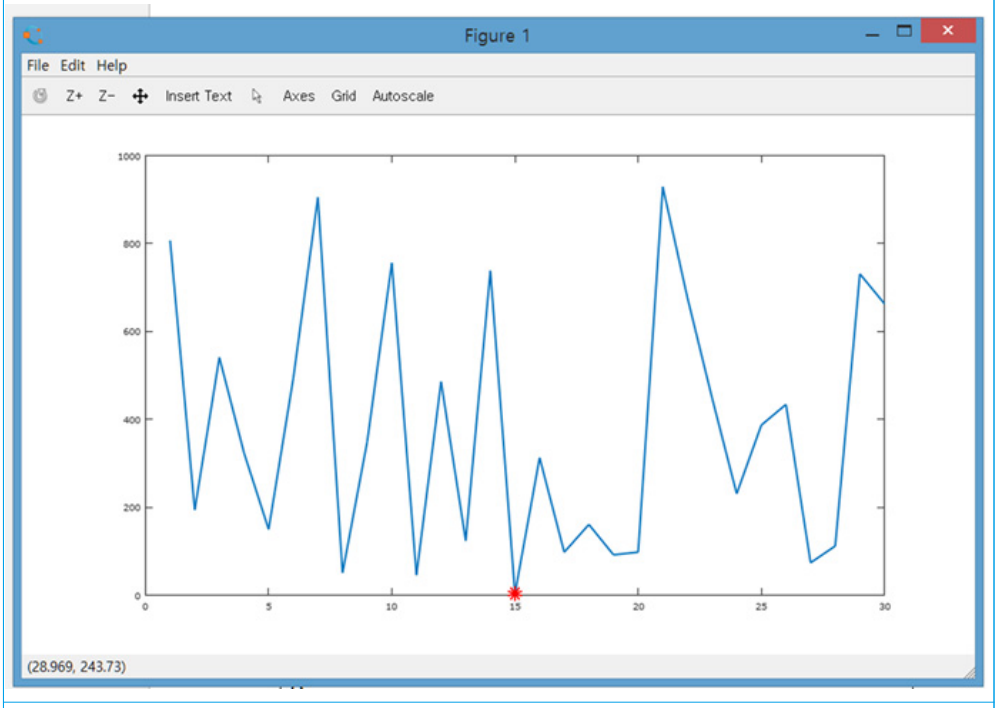
```
clear
n=30;
A=randi([1 1000],n,1)
s=1000;
for i=1:n
   if A(i)<s
       s=A(i);
       m=i;
   end
end
[s m]
clf;
plot(A,'linewidth',2); hold on
plot(m, A(m),'ro','linewidth',20)
```

```
% 최솟값과 벡터의 몇 번 째값이 최솟값인지 출력
clf;
plot(A,'linewidth',2); hold on
% 30개의 랜덤 값을 플롯
plot(m, A(m),'ro','linewidth',20)
% 최솟값을 플롯
```

명령문 창 결과

```
>> minA
A =
   807
   194
   541
   324
   150
   493
   905
    51
   349
   756
    46
   486
```

그래프 결과

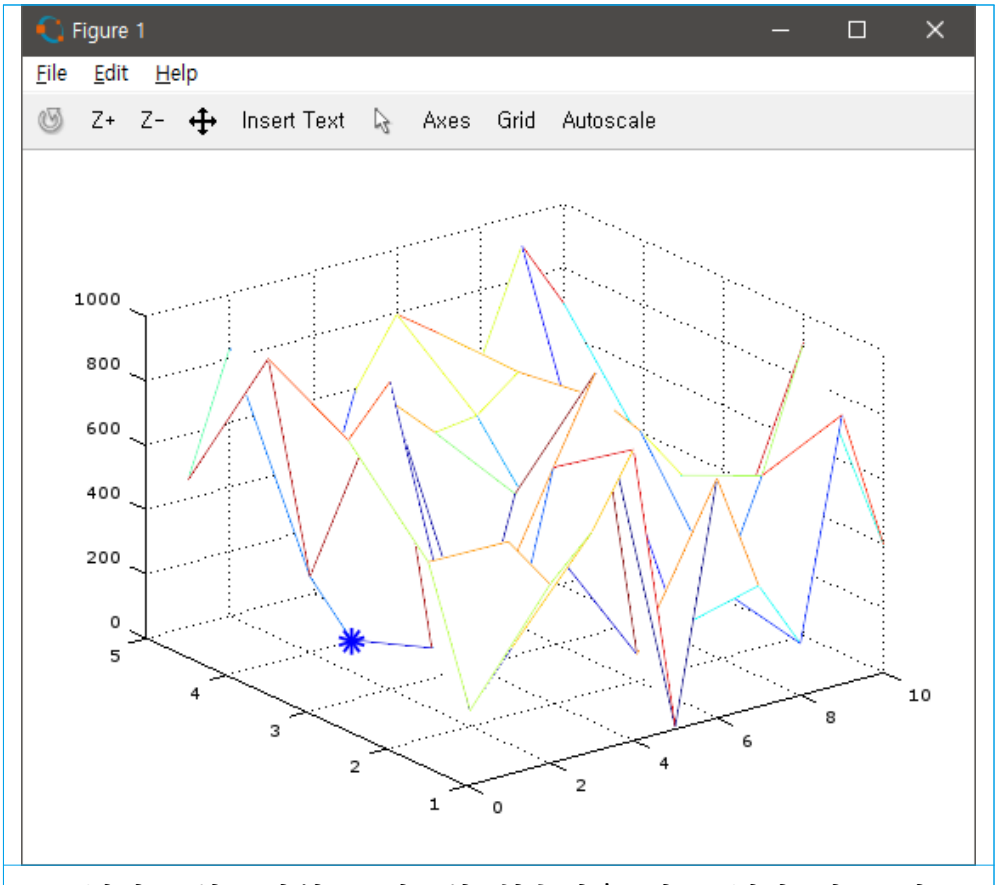


30개의 랜덤 값과 최솟값을 플롯

코드설명

```
clear
% 변수초기화
nx=10;
% nx는 행의 갯수
ny=5;
% ny는 열의 갯수
B=randi([1 1000], nx, ny)
% 임의로 1부터 1000까지의 정수를 10행 5열로 생성
mesh(B')
% 생성된 행렬을 그래프로 출력
s=1000;
% 행렬 B가 가질 수 있는 최댓값 1000으로 초기화
for i=1:nx
  for j=1:ny
     if B(i,j)<s
  % 만약 행렬의 값이 현재 최솟값 보다 작다면
        s=B(i,j);
 % 현재 최솟값을 더 작은 행렬 값으로 변경
        mini=i;
        minj=j;
 % mini에 행번호를 minj에 열 번호를 저장한다.
```

그래프 결과



행렬 B의 3차원 그래프와 최솟값을 갖는 행과 열 표시.

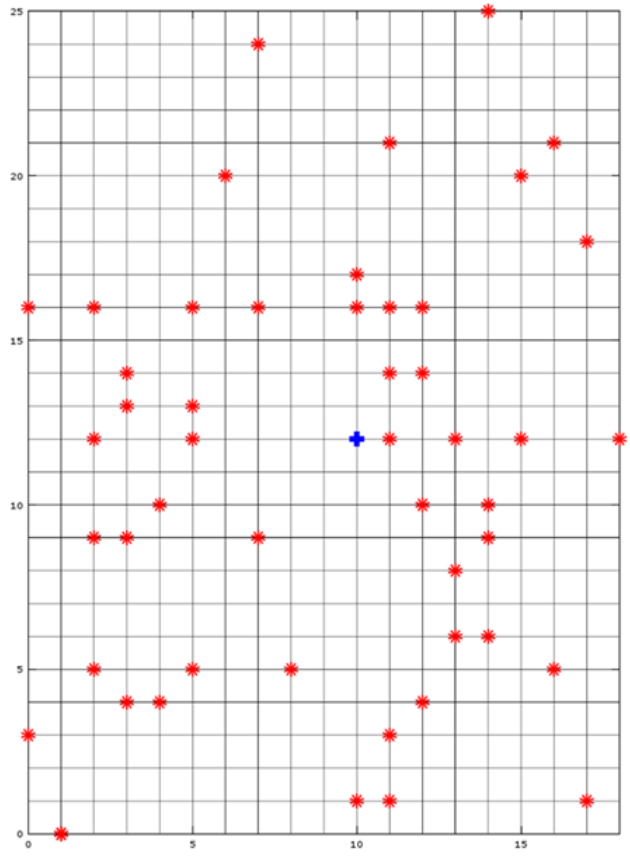
```
for i=0:nx
  for j=0:ny
    d=0;
    for k=1:NoH
     d=d+abs(x(k)-i)+abs(y(k)-j);
    end
    fire(i+1,j+1)=d;
    end
end
s=(nx+ny)*NoH;
for i=1:nx+1
  for j=1:ny+1
    if fire(i,j)<s
      s=fire(i,j);
      m=i; n=j;
    end
    end
end
plot(m-1,n-1,'bs','linewidth',30); axis image
figure(2); clf; mesh(fire'); hold on
plot3(m,n,'bo','linewidth',30)
axis([0 nx 0 ny 0 max(max(fire))])
```

```
  for j=0:ny
    d=0;
    for k=1:NoH
     d=d+abs(x(k)-i)+abs(y(k)-j);
    end
% 임의의 점들과 특정 점과의 택시거리 구하기 참조
    fire(i+1,j+1)=d;
% fire의 0행이나 0열은 없으므로 i와 j에1을 더해서 저장해준다.
    end
end
% 주어진 좌표인 (0,0), (0,1), ... , (18,25) 모든 점에 대한 택시거리를 계산하여 fire 행렬에 저장한다.
s=(nx+ny)*NoH;
% 모든 점에 대한 택시거리가 (nx+ny)*NoH을 넘지 않는다.
for i=1:nx+1
  for j=1:ny+1
    if fire(i,j)<s
      s=fire(i,j);
      m=i; n=j;
```

명령문 창 결과

```
>> FireStationLocation
ans =    10      12     585
```

그래프 결과



빨간 점은 무작위로 추출한 50개의 집이다. 위 50개 빨간 점으로부터 최단거리를 가진 점이 파란점이고 최적의 소방서 위치이다.