

Grids

Geometry

Computational Geometry

# Grids

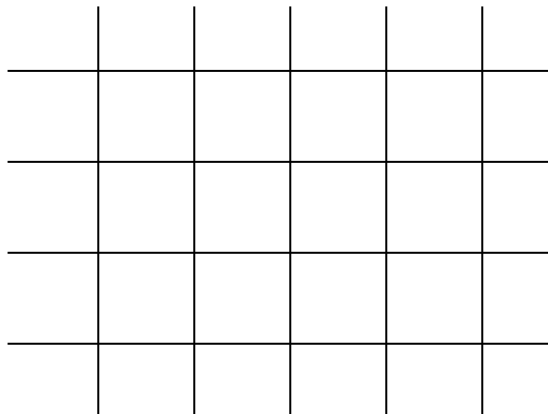
---

- You will encounter many problems whose solutions need grid representation
  - map
  - VLSI layout
  - the most natural way to carve space into regions
- regular pattern
- rectangular or rectilinear grids are common
  - but sometimes you will see triangular or hexagonal grids

# Rectilinear Grids

---

- each cell is defined by horizontal and vertical lines
- spacing between lines can be
  - uniform
  - non-uniform
- three dimensional grid is difficult to visualize



vertices

edges

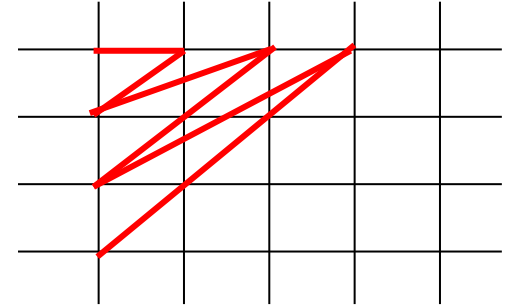
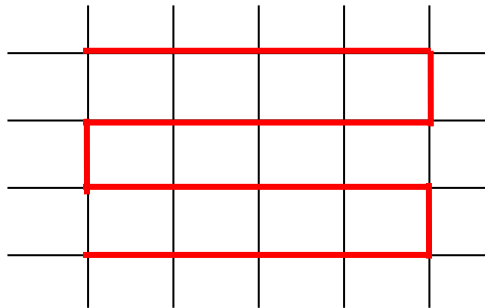
cells

# Graph Representation of Grid

- Grid is a graph (V, E)
  - each vertex contains information
    - city, village, transistor, ..
  - each edge represents relationship between vertices
  - can be 2D, 3D, 4D, ...

- Traversal

- row major
- column major
- snake
- diagonal
- ....



```
snake_order(int n, int m)
{
    int i,j;          /* counters */

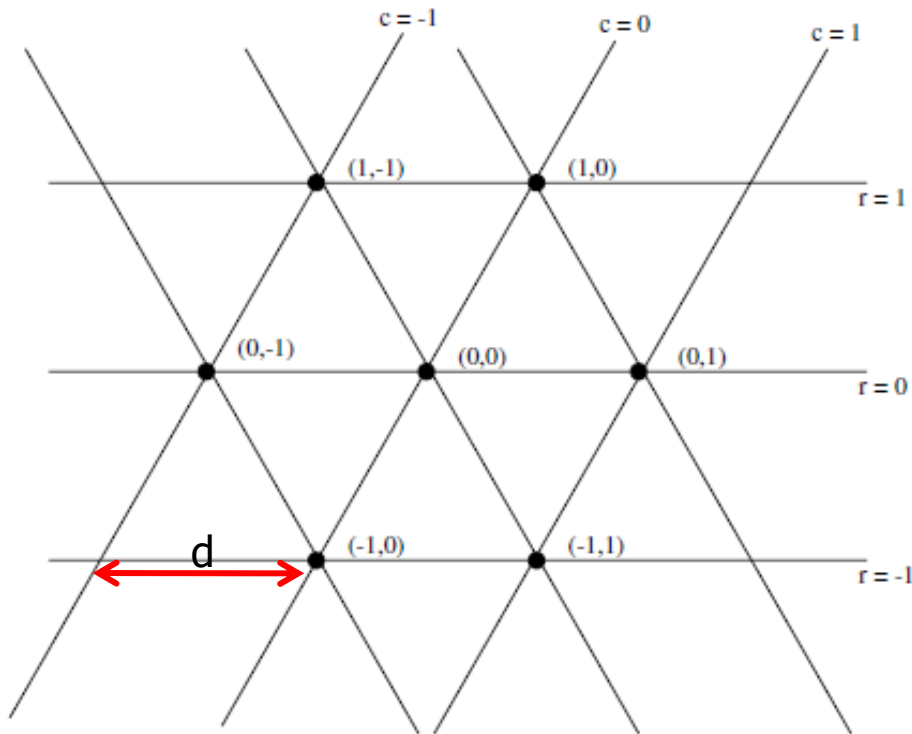
    for (i=1; i<=n; i++)
        for (j=1; j<=m; j++)
            process(i, j + (m+1-2*j) * ((i+1) % 2));
}
```

# Dual Graphs

---

- $m[i][j]$  can be either
  - a vertex
  - a face
  - Any one can be transformed into the other
  - proof?
- The four-color theorem
  - the map and the graph to solve the problem are different
- How can you represent weights on edges?

# Triangular Lattices



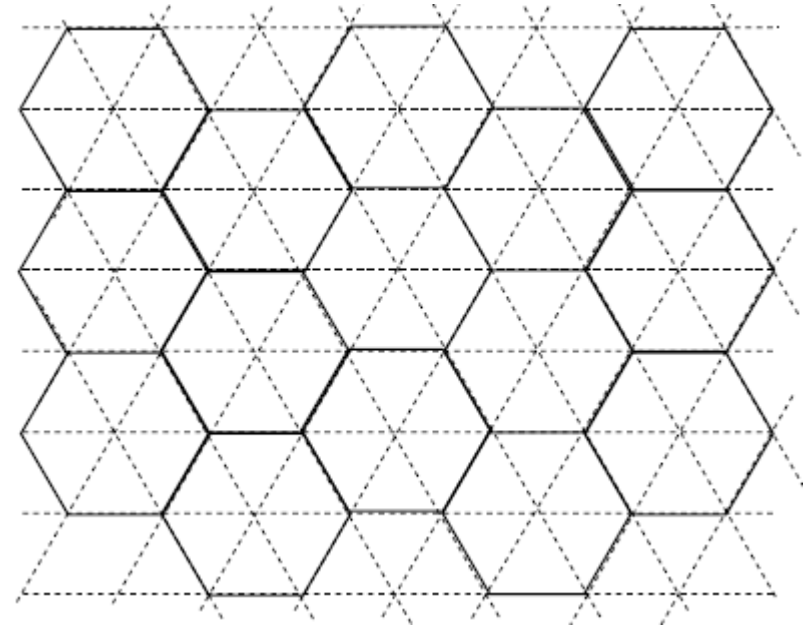
- a vertex needs to be assigned coordinates
  - $(x, y)$  is good enough
  - if edges are removed, it looks like a grid
- Duality?
- Geometrical coordinates

$$(x_g, y_g) = (d(x_t + (y_t \cos(60^\circ))), dy_t \sin(60^\circ))$$

# Hexagonal Lattices

---

- It is a subset of a triangular lattice
  - remove every other vertex
- Interesting properties
  - it is the roundest polygon to fill the space
  - per space/perimeter
    - a circle is the best





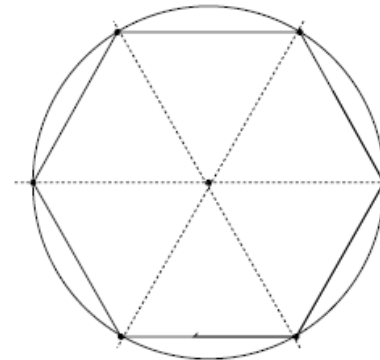
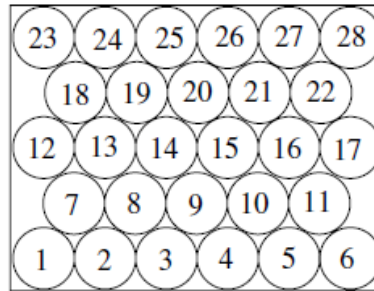


# Exercise

---

- Dish Packing

- How many dishes can be places in a box?

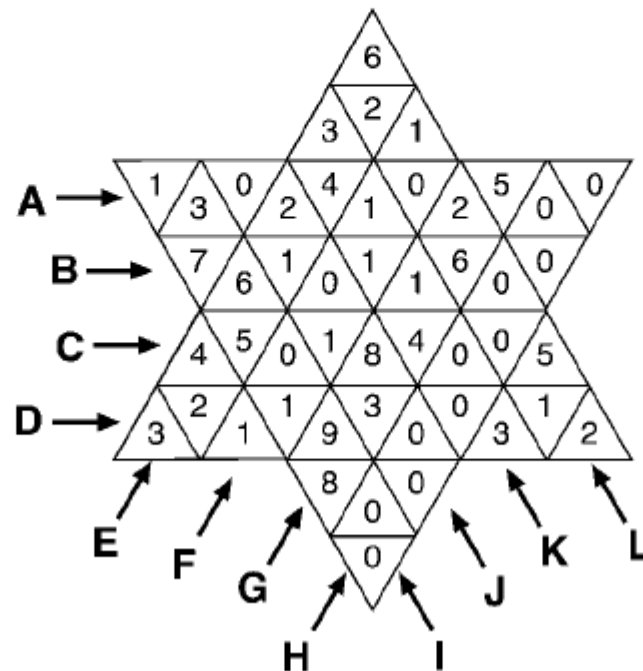


- Flying path

- find the shortest flying distance from Seoul to Bagdad

# Problem 1: Star

A board contains 48 triangular cells. In each cell is written a digit in a range from 0 through 9. Every cell belongs to two or three lines. These lines are marked by letters from *A* through *L*. See the figure below, where the cell containing digit 9 belongs to lines *D*, *G*, and *I* and the cell containing digit 7 belongs to lines *B* and *I*.



*Sample Input*

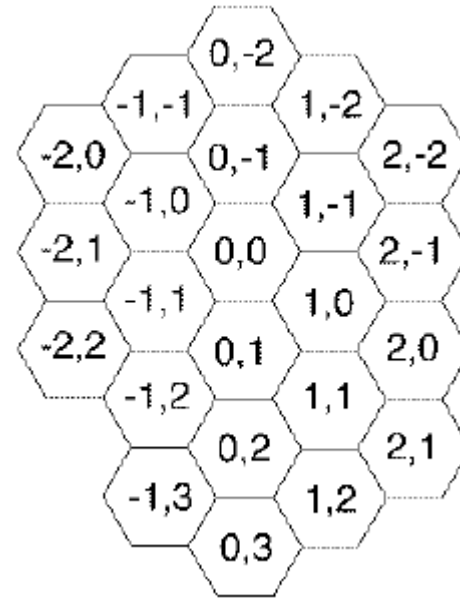
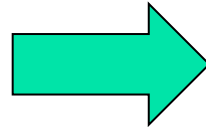
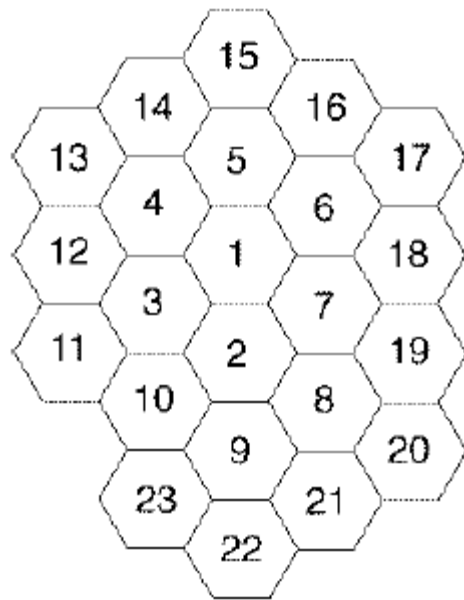
5 7 8 9 6 1 9 0 9 8 4 6

*Sample Output*

40 172

## Problem 2: Convert coordinate numbers

---



# Geometry

---

- You are supposed to know enough geometry
- How can you apply this knowledge to solve problems?
  - even a simple solution with pencil and paper needs severe elaboration for a program
- You should be familiar with two disciplines
  - geometry
  - representation of geometric problem/solution as a software program

# Lines

---

- representations

- two points:  $(x_1, y_1)$  and  $(x_2, y_2)$
- a single point and a slope:  $y = mx + b$ 
  - $m = (y_1 - y_2)/(x_1 - x_2)$
  - what if  $(x_1 - x_2)$  is 0? – needs special attention for division
- more general:  $ax + by + c = 0$ 

```
typedef struct {  
    double a;  
    double b; // default value is 1  
    double c;  
} line
```
- any one of the above can be converted to any other

# Excercises

---

- find an intersection of two lines

$$l_1 : y = m_1x + b_1 \text{ and } l_2 : y_2 = m_2x + b_2$$

- check is they are parallel
- else find

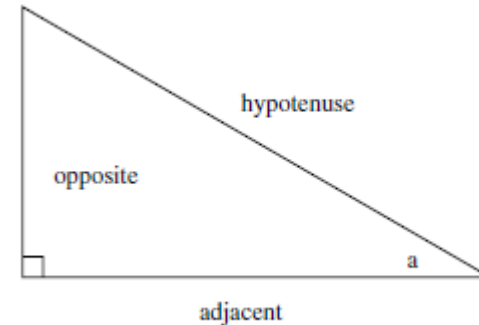
$$x = \frac{b_2 - b_1}{m_1 - m_2}, \quad y = m_1 \frac{b_2 - b_1}{m_1 - m_2} + b_1$$

- what else?
  - angles of two lines
  - closest point on a line
  - rays – half line with an origin

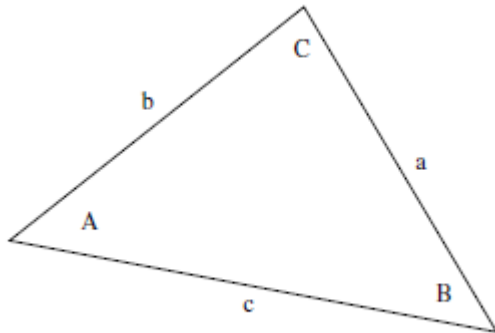
# Triangles and Trigonometry

- angles: use radians since most libraries use them
- make yourself comfortable with terminologies

- right triangle
- perpendicular lines
- internal/external angle
- equilateral
- hypotenuse



- Remember them?



$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

$$a^2 = b^2 + c^2 - 2bc \cos A$$

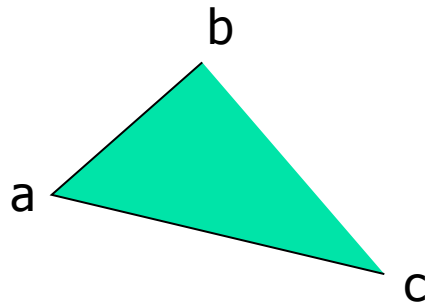
# Triangle Area

---

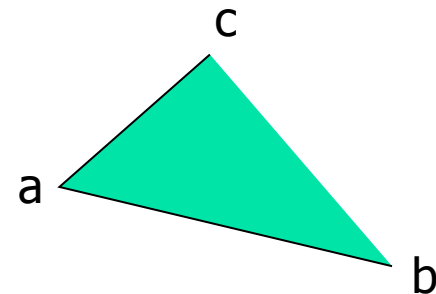
singed triangle area

$$2 \cdot A(T) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = a_x b_y - a_y b_x + a_y c_x - a_x c_y + b_x c_y - c_x b_y$$

positive



negative





# Circles

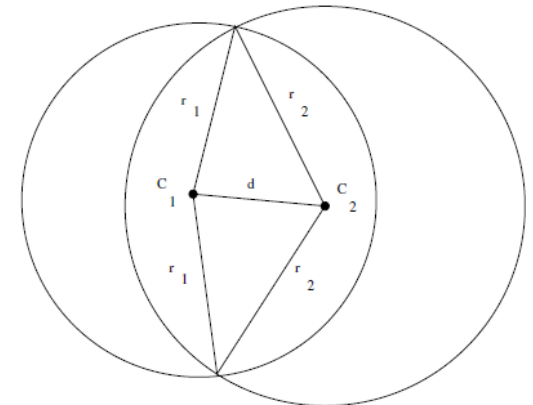
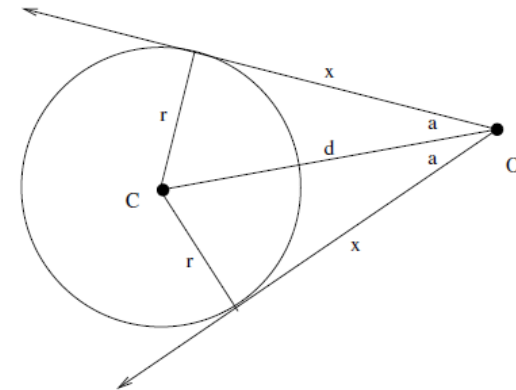
- representation

```
typedef struct {  
    point c;           /* center */  
    double r;         /* radius */  
} circle;
```

- formula:  $r^2 = (x - x_c)^2 + (y - y_c)^2$

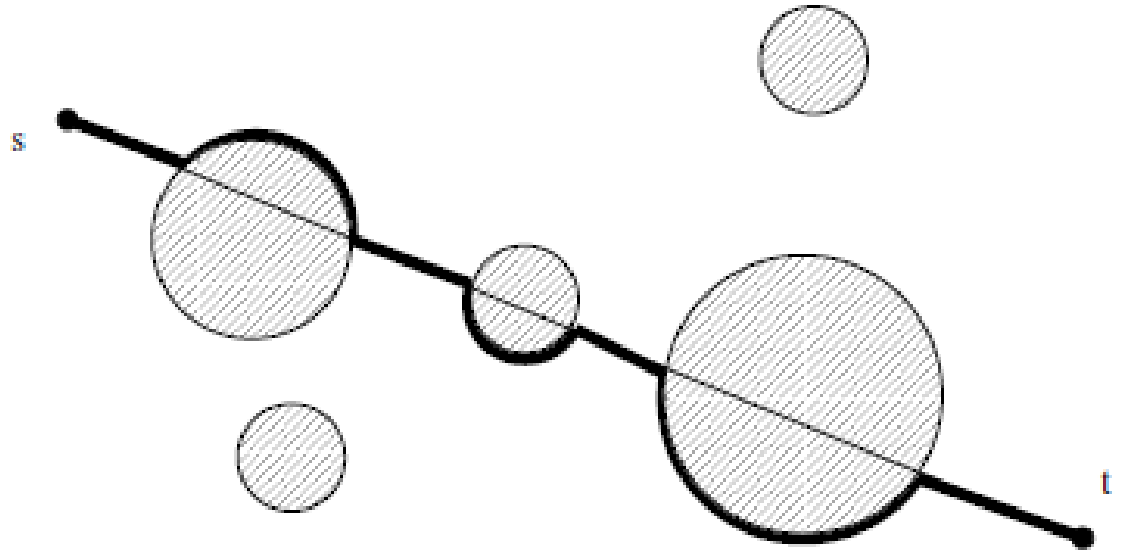
- problems

- tangent line
- intersection points



# Excercise

---



- find the travel distance
  1. find circles intersecting the s-t line
  2. find the length of the chord
  3. compute the arc length

# Libraries

---

```
#include <math.h>

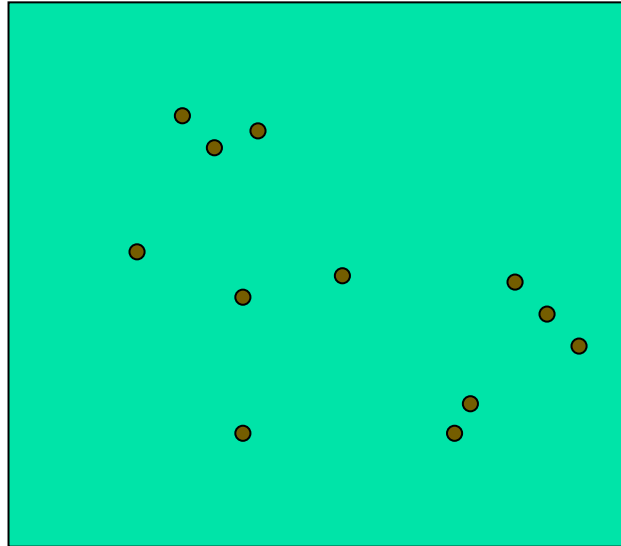
double cos(double x);    /* compute the cosine of x radians */
double acos(double x);  /* compute the arc cosine of [-1,1] */

double sin(double x);   /* compute the sine of x radians */
double asin(double x); /* compute the arc sine of [-1,1] */

double tan(double x)    /* compute the tangent of x radians */
double atan(double x); /* compute the principal arctan of x */
double atan2(double y, double x); /* compute the arc tan of y/x */
```

# Excercise

---



- 50- cm square plate
- place a circle of 5 cm to contain most chips

# Computational Geometry

---

- You need to handle geometry data
  - graphics
  - navigation
  - CAD
- Objects are usually
  - lines and polygons
- computational geometry deal with them using a computer
  - the most fascinating and growing area in mathematical sciences

# Line Segments

---

- a portion of a line

```
typedef struct {  
    point p1, p2;  
} segment;
```

- Are two segments intersect?
  - first, deal with degeneracy cases
    - parallel
    - total overlap

# Polygons

---

- definition

- a closed chain of non-intersecting line segments

```
typedef struct {  
    int n;          /* number of vertices */  
    point p[MAXPOLY]; /* vertices */  
} polygon;
```

- convex

- all internal angles are acute
- “signed triangle area function” tests if a point is left or right of a line

```
bool ccw(point a, point b, point c)  
{  
    double signed_triangle_area();  
  
    return (signed_triangle_area(a,b,c) > EPSILON);  
}
```

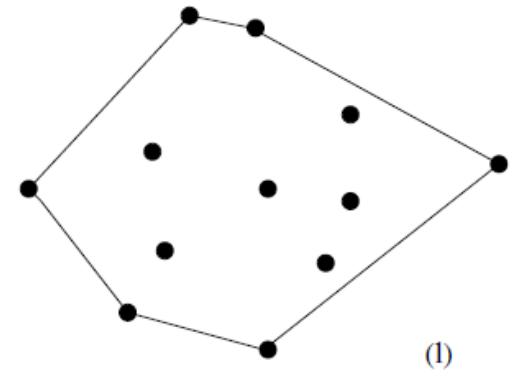
# Convex Hulls

---

- similar to sorting
- The smallest polygon containing a set of points

- Solutions

- find the leftmost-lowest point (origin)
- sort points according to angles to the origin
- add one by one
- be careful of degeneracy





# Finding area of a polygon

- triangulation
  - convex is easy
  - then, make it a convex
- ear cutting method
  - find an ear, calculate its area
  - do this until there remains a triangle
- How to test if a point is inside a polygon?

