Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCA UFES
Departamento de Computação

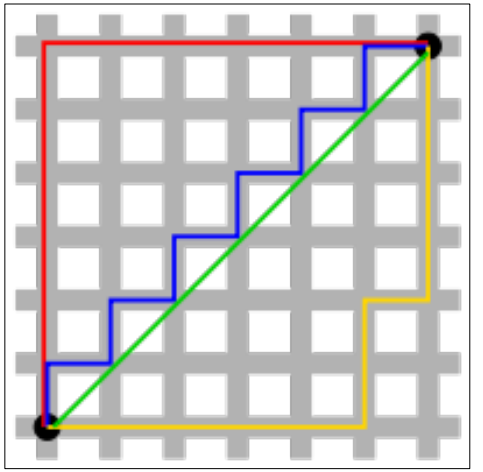
Grades (Grids)

Tópicos Especiais em Programação

Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

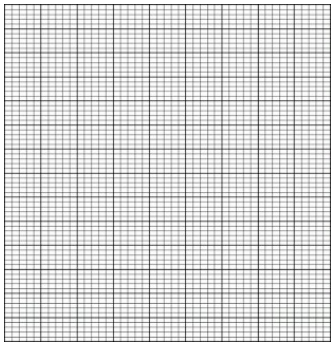
Universidade Federal do Espírito Santo – CCA UFES

Grades



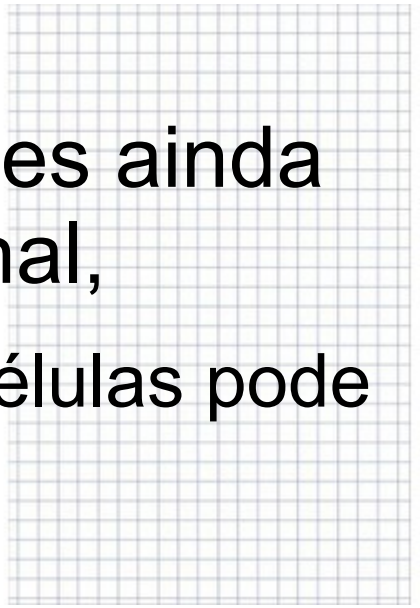
- Grades constituem a base de uma grande variedade de estruturas naturais:
  - Tabuleiros, Quarteirões da cidade;

    (normalmente organizados em uma grade – distância “Manhattan”);
  - O sistema de longitude e latitude define uma grade sobre a terra;
    - Embora sua superfície seja uma esfera ao invés do um plano.
- Grades são onipresentes, porque elas são a forma mais natural para esculpir espaços em regiões, permitindo que as localizações possam ser identificadas.
- Suas células podem ser pontos individuais, mas podemos trabalhar com grades maiores que são grandes o bastante para definir um formato específico.
- Em grades regulares, cada uma de suas células são idênticas e elas seguem um padrão regular.
  - Grades retangulares ou com subdivisões retangulares são os tipos mais comuns, devido a sua simplicidade. Mas grades triangulares, ou hexagonais também são importantes.



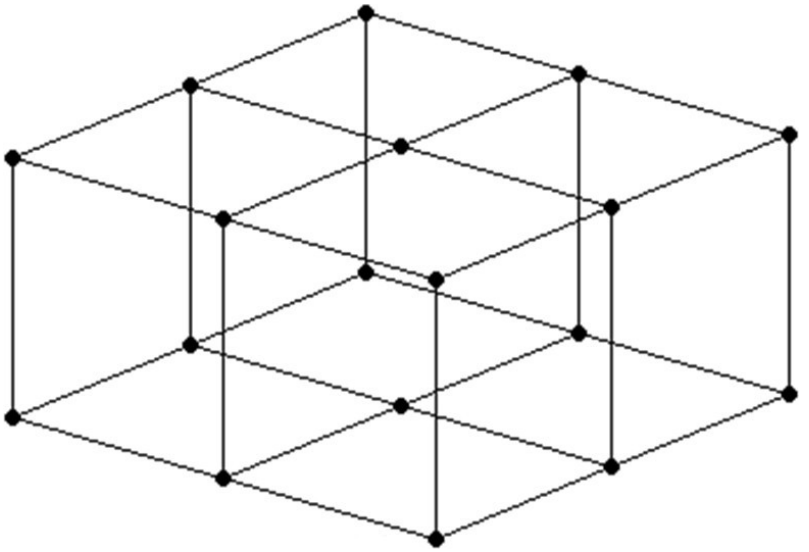
Grades retilíneas

- Grades retilíneas são familiares para qualquer pessoa que tenha usado um pedaço de folha quadriculada.
  - Em tais grades, as células são tipicamente definidas por linhas horizontais e verticais espaçadas regularmente.
- Até folhas com espaços não uniformes ainda produzem uma topologia convencional,
  - embora o tamanho de cada uma das células pode ser diferente.



Grades tridimensionais

- Grades tridimensionais são formadas por camadas de grades planares espaçadas uniformemente e conectadas entre si
  - Possuem linhas perpendiculares entre as camadas;
  - Também têm faces planas, definidas entre dois cubos vizinhos.



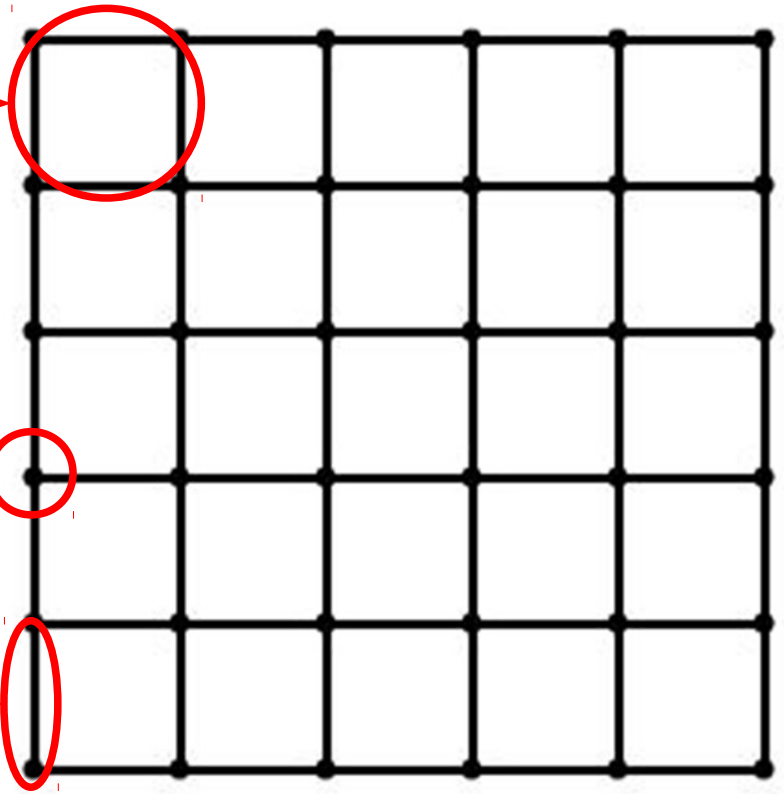
Grades

- Há três componentes importantes de uma grade planar:
  - Interiores das **Células**.

    Ex.: Aplicações geométricas, em que o conteúdo de uma célula descreve uma região no espaço
  - **Vértices**.

    Ex.: Endereçamento de peças em um tabuleiro.
  - **Arestas**.

    Ex.: Rotas para viajar em uma cidade, onde as construções localizam-se no interior das células.



Grades

- Vértices:
  - Cada vértice de uma grade planar toca quatro arestas e atinge o conteúdo de quatro células
    - Exceto os vértices das bordas.
  - Vértices em grades 3D tocam 6 arestas e atingem 8 células;
  - Em **d** dimensões, cada vértice toca **2d** arestas e atinge $\mathbf{2^d}$ células.
- Células:
  - As células em uma grade planar tocam oito faces:
    - Quatro diagonalmente através dos vértices e
    - Quatro através das arestas.
  - As células em grades 3D tocam outras 26 células.
    - Compartilham sua face com 6 células;
    - Compartilham suas arestas com 12 células;
    - Compartilham um vértice com outras 8 células.

Grades

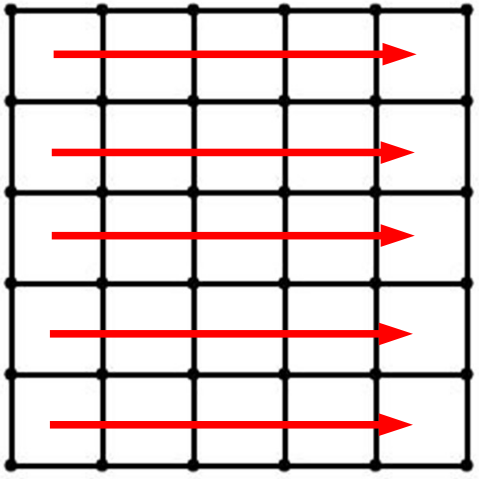
- Frequentemente é necessário percorrer todas as células de uma grade retilínea **n×m**
  - Qualquer percurso/travessia pode ser tratado como um mapeamento de cada um dos **nm** pares ordenados para um único inteiro de 1 a **nm**;
  - Em certas aplicações, a ordem do percurso importa, como em estratégias de avaliação de programação dinâmica.

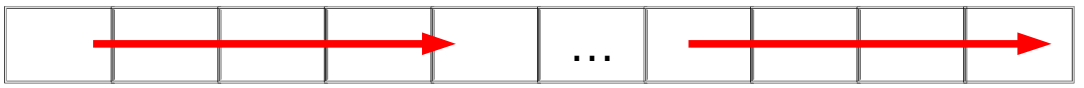
Grades

- Os métodos de percurso/travessia mais importantes são:
  - Orientado por linha (*row major*);
  - Orientado por coluna (*column major*);
  - *Snake Order*;
  - Ordem Diagonal (*Diagonal Order*).
- Em matrizes, para otimizar o uso da cache ou para utilizar operações aritméticas com ponteiros, é interessante saber como o compilador tratará esses dados.

Orientado por linha

- Aqui nós dividimos a matriz entre linhas, então:
  - Os primeiros **m** elementos visitados são os da primeira linha;
  - Os próximos **m** elementos visitados são os da segunda linha;
  - e assim por diante;
- Essa orientação é utilizada dentro dos compiladores das linguagens mais modernas para representar matrizes bidimensionais como um único vetor linear (unidimensional).



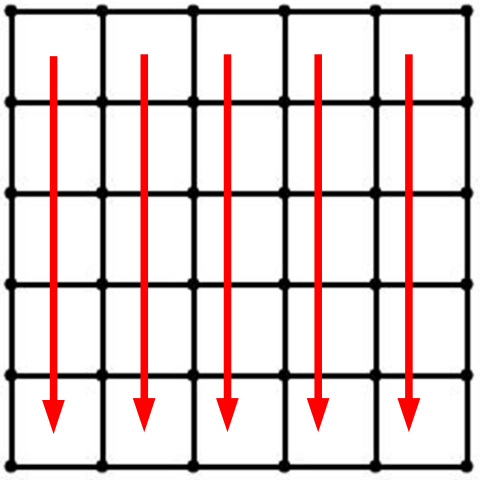


Orientado por linha

```
void row_major(int n, int m)
{
    for (int lin=1; lin<=n; lin++)
        for (int col=1; col<=m; col++)
            process(lin, col);
}
```

Orientado por coluna

- Aqui nós dividimos a matriz entre colunas. Então:
  - Os primeiros **n** elementos visitados são da primeira coluna;
  - Os próximos **n** elementos visitados são da segunda coluna;
  - e assim por diante;
  - Isso pode ser feito simplesmente pela troca dos laços do percurso orientado por linha;
- Sabendo se seu compilador utiliza a orientação por linha ou por coluna é importante para otimizar a performance da cache e também quando for utilizar operações com ponteiros.

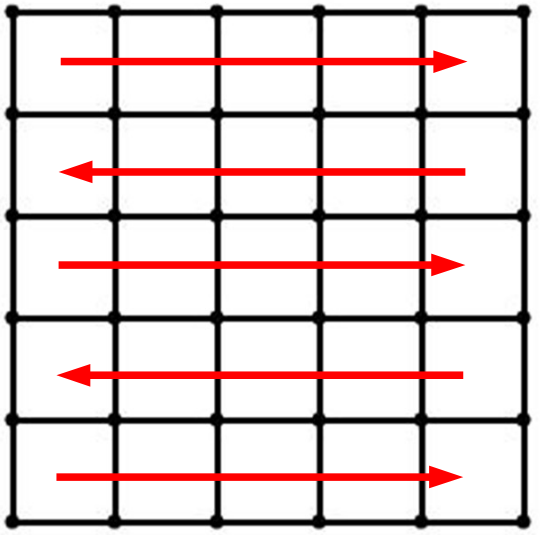


Orientado por coluna

```
void column_major(int n, int m)
{
    for (int col=1; col<=m; col++)
        for (int lin=1; lin<=n; lin++)
            process(lin, col);
}
```

Snake Order

- Ao invés de começar cada linha a partir do primeiro elemento, as direções são alternadas entre uma linha e outra;
- O efeito obtido é o mesmo das impressoras que imprimem da esquerda para direita e da direita para esquerda, economizando tempo e energia.

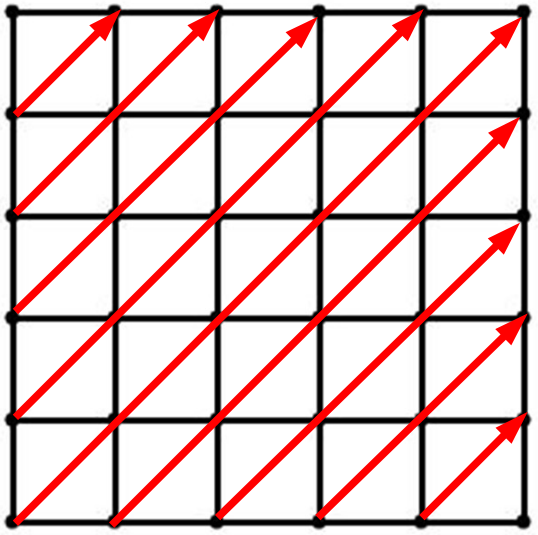


Snake Order

```
void snake_order(int n, int m)
{
   for (int lin=1; lin<=n; lin++)
      for (int col=1; col<=m; col++)
       process(lin, col + (m+1-2*col) * ((lin+1) % 2));
}
```

Ordem diagonal

- Aqui são percorridas as diagonais para cima e para baixo.
- Note que uma grade **n×m** tem **m+n-1** diagonais, cada uma com um número variável de elementos.
- Esta é uma tarefa mais complicada do que parece ser a primeira vista.



Ordem diagonal

```
void diagonal_order(int n, int m) {
    int d,j;    /* contadores de diagonal e de pontos */
    int pcount; /* pontos da diagonal */
    int height; /* linha do menor ponto */

    for (d=1; d<=(m+n-1); d++) {
        height = 1 + max(0, d-m);
        pcount = min(d, (n-height+1));
        for (j=0; j<pcount; j++)
            process(min(m,d)-j, height+j);
    }
}
```

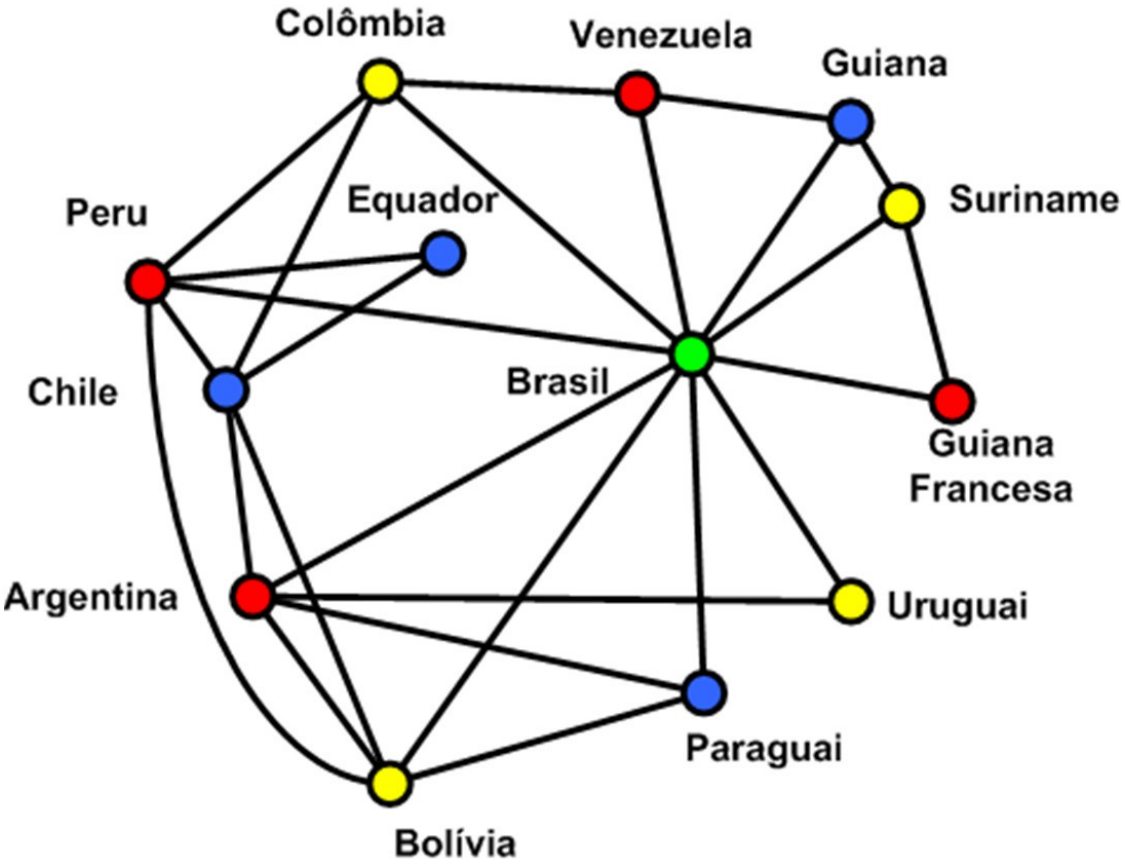
Grades

- Arrays bidimensionais são uma escolh anatural para representar grades retilineares.
- Podemos utilizar m[i][j] para representar tanto o vértice (i,j) quanto a face (i,j), depende do interesse.
  - Os quatro vizinhos adjacentes são determinados ao adicionar ±1 aos índices de linha ou coluna.

Grafos duais

- Um conceito muito útil quando pensamos em problemas que envolvem subdivisões do plano é o de *grafo dual*
  - Possui um vértice para cada região na subdivisão;
  - Arestas ligam vértices de duas regiões vizinhas.
- Um exemplo de aplicação é o de coloração de mapas, e correspondentemente, o teorema das 4 cores;
- Um fato muito interessante é que um grafo dual de uma subdivisão planar também deve ser planar.

Grafo dual


Colômbia
Venezuela
Guiana
Suriname
Equador
Peru
Chile
Brasil
Guiana Francesa
Argentina
Uruguai
Paraguai
Bolívia

Representação

- Grades também podem possuir arestas ponderadas, ou seja, com pesos.
  - Pode-se, por exemplo, criar uma matriz tridimensional **m[i][j][d]**,
  - onde **d** armazena um dentre quatro valores (norte, leste, sul, oeste) que denotam a direção da aresta (i, j).

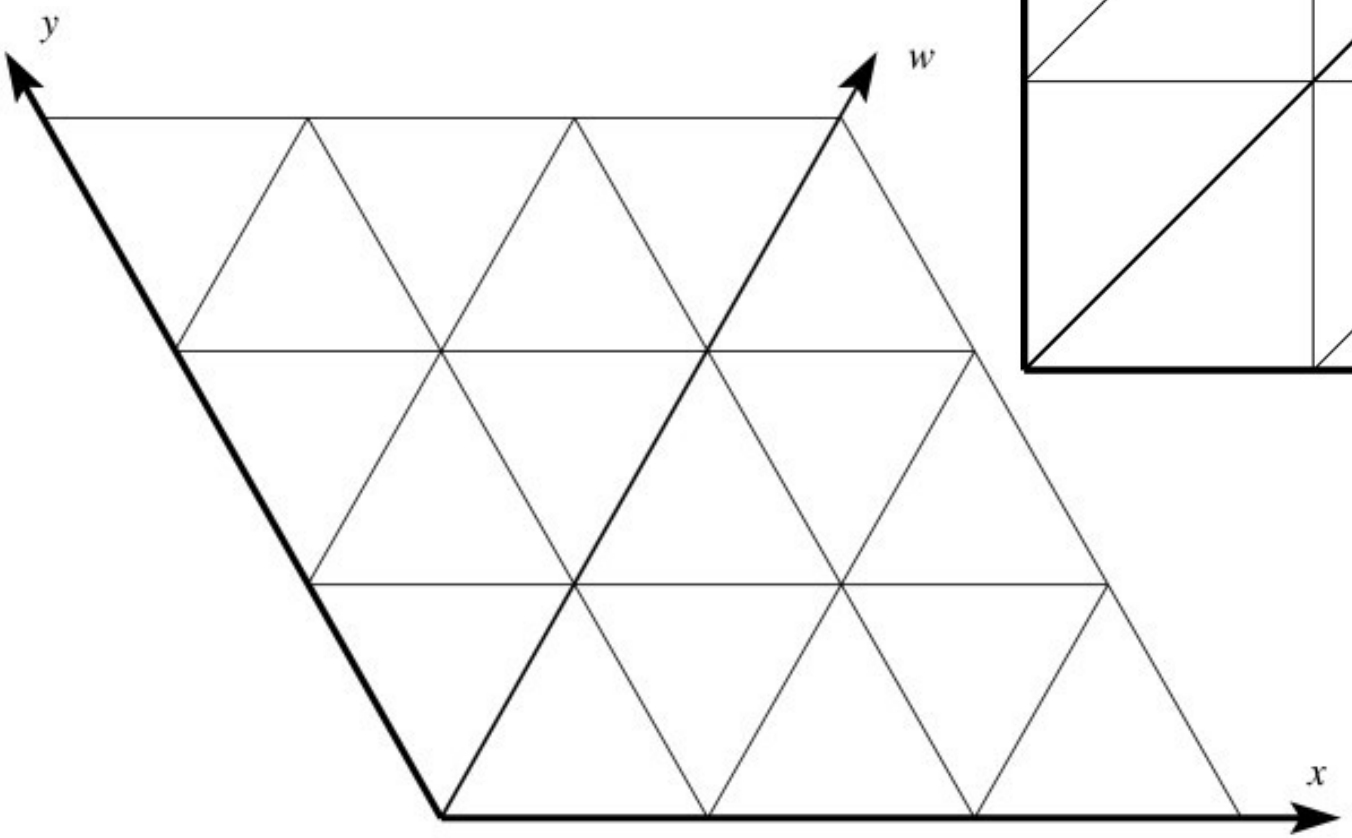
Grades triangulares e hexagonais

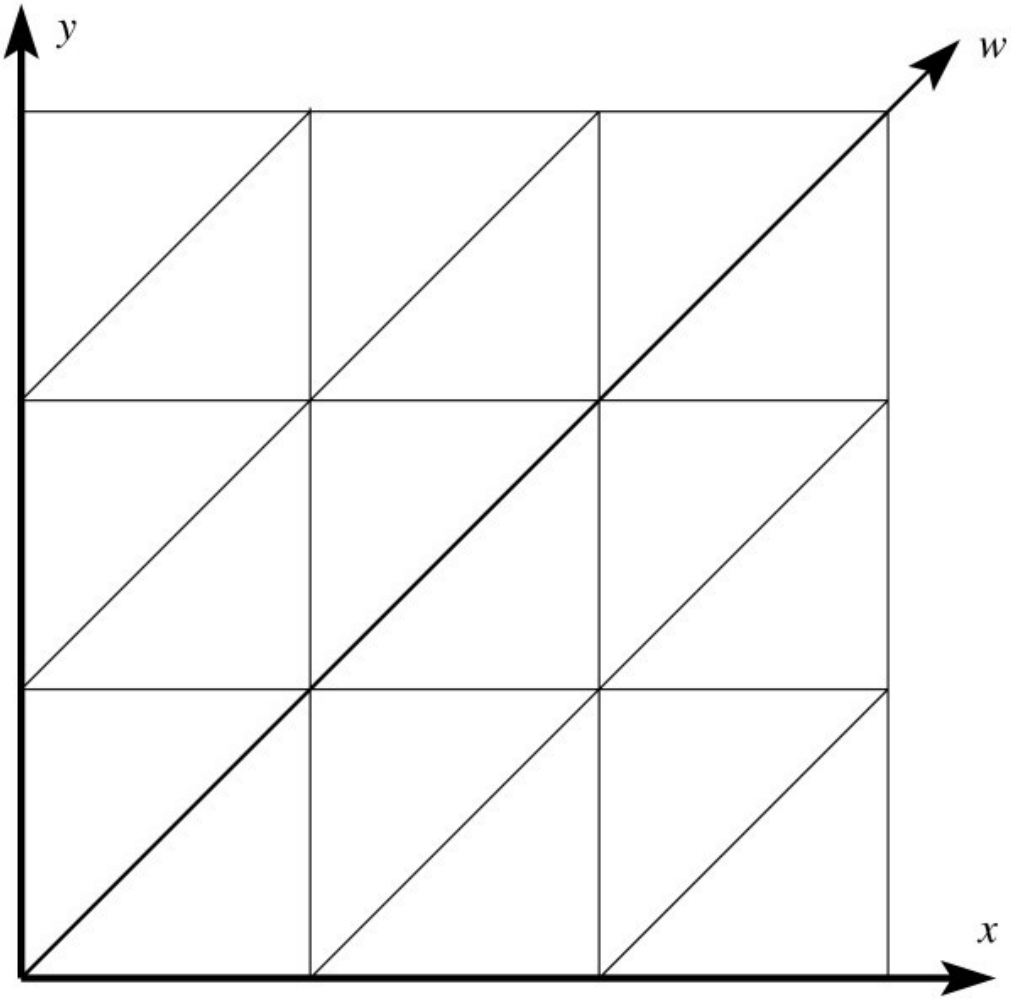
- Além das grades retilíneas, outras grades de interesse são as grades triangulares e hexagonais:
  - Ambas são intimamente relacionadas;
  - Na verdade, as grades hexagonais são essencialmente grades triangulares com alguns vértices removidos.

Grades triangulares

- Grades triangulares são construídas a partir de conjuntos de três linhas igualmente espaçadas:
  - Uma “linha” eixo horizontal;
  - Uma “coluna” eixo a 60°da horizontal;
  - Um eixo “diagonal” a 120°da horizontal.
- Os vértices da grade triangular são formados pela interseção desses três eixos, de modo que cada face é um triângulo equilátero:
  - Cada vértice é ligado em outros seis.

Grades Triangulares





Grades Triangulares

- Para manipularmos grades triangulares, precisamos identificar os vizinhos de cada um dos vértices e suas coordenadas.
- Para isto, precisamos manter informações de dois tipos de sistemas de coordenadas:
  - Coordenadas triangulares/hexagonais;
  - Coordenadas geométricas.

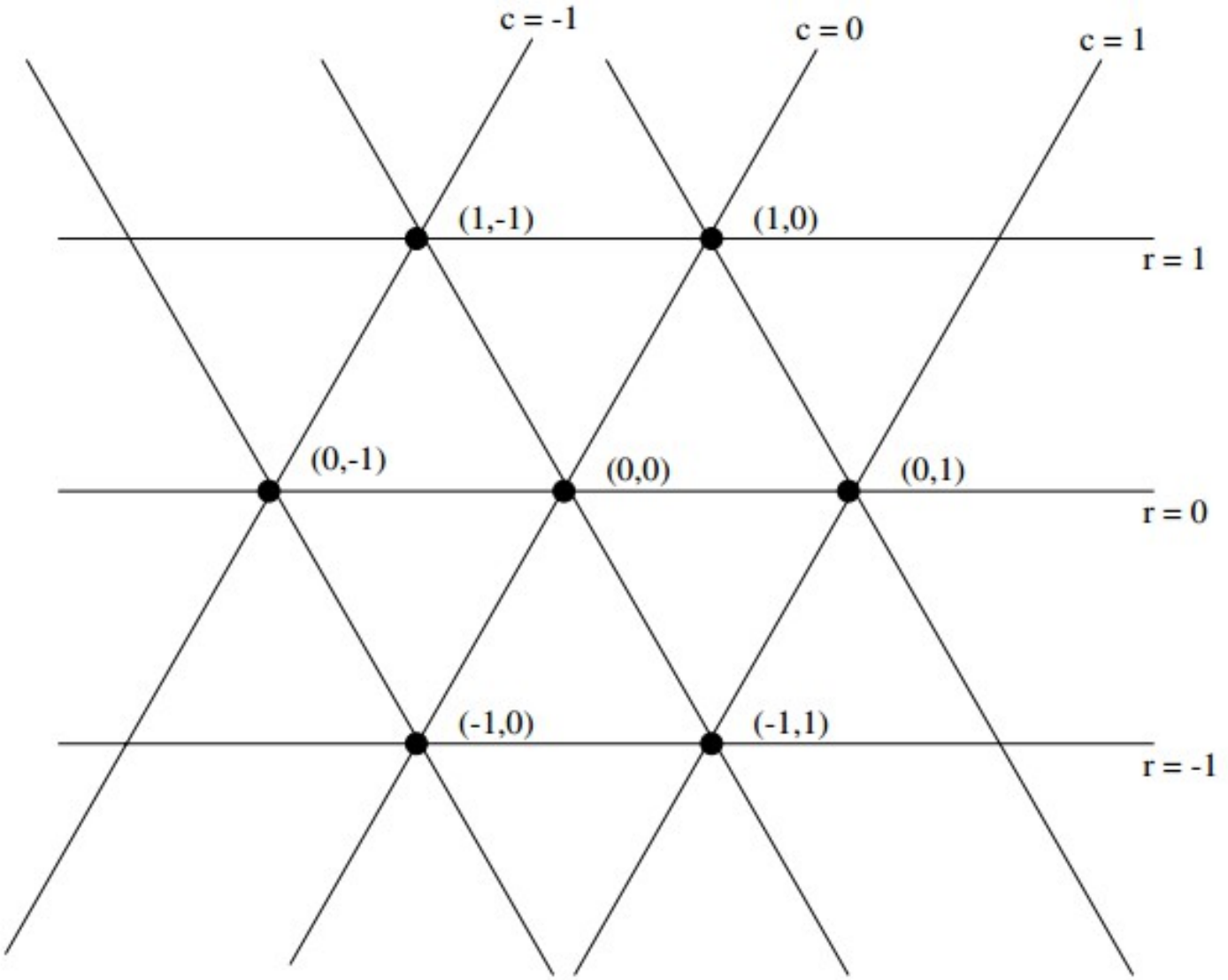
Coordenadas triangulares/hexagonais

- Em coordenadas triangulares/hexagonais, um vértice é designado com origem da grade, ponto (0, 0);
  - Podemos atribuir coordenadas de tal forma que os vizinhos de cada vértice possam ser facilmente obtidos.
  - Dentro de um sistema de coordenada linear, os vizinhos podem ser obtidos adicionando ±1 em cada linha ou coluna.
- Apesar de a interseção de três linhas definirem um vértice, duas dimensões são suficientes para determinar uma localização
  - Usamos as dimensões de linha e coluna neste sistema de coordenadas.

Um vértice (x, y) está x linhas acima da origem e y colunas (de 60°) à direita da origem;

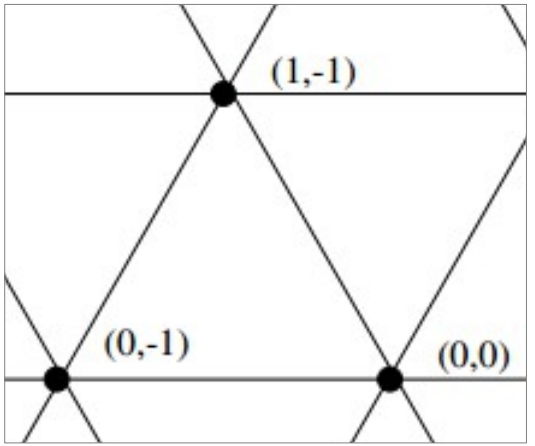
Os vizinhos de um determinado vértice v podem ser acessados pela adição dos seguintes pares de coordenadas em sentido anti-horário:

(0,1), (1, 0) , (1, -1), (0, -1), (-1, 0) e (-1, 1).



Coordenadas Geométricas

- Os vértices de uma grade triangular também correspondem a pontos geométricos no plano.
- Note que os vértices se localizam em linhas de meio espaçamento,
  - A coluna é a 60°e não a 90°como em grades retilíneas.



Coordenadas Geométricas

- Considere que:
  - Cada ponto da grade está a uma distância **d** de seus 6 vizinhos;
  - O ponto (0, 0) em coordenadas triangulares está de fato no ponto geométrico (0, 0);
  - Pela geometria, o ponto geométrico triangular de coordenada (x_t, y_t) deve estar no ponto geométrico:

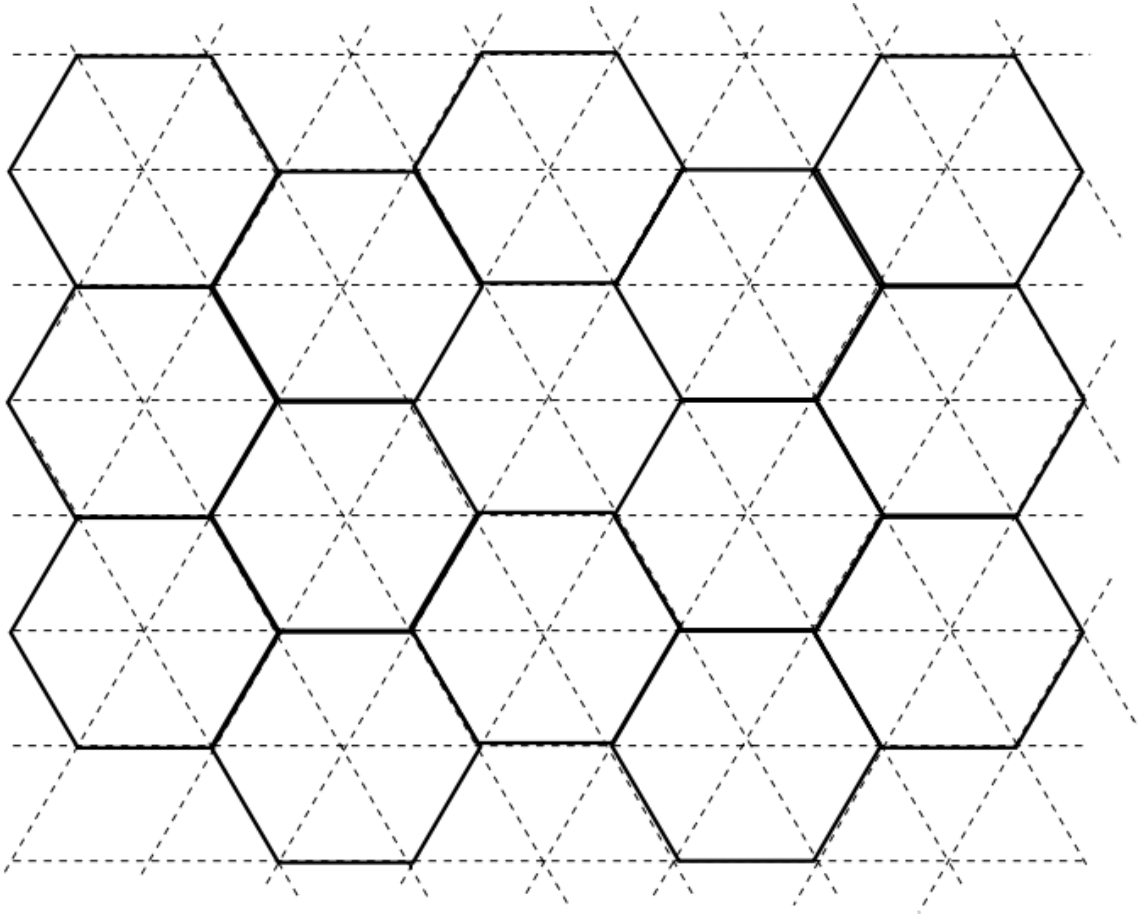
$$(x_g, y_g) = (d(x_t + (y_t \cos(60^o))), dy_t \sin(60^o))$$

- Nós nem sequer precisamos utilizar funções trigonométricas para esse cálculo, pois:

$$\cos(60°) = \frac{1}{2} \text{ e } sen(60°) = \sqrt{3}/2$$

Grades Hexagonais

- Apagando os vértices centrais apropriados transforma uma grade triangular em uma grade hexagonal.
  - Assim, as faces se transformam em hexágonos regulares;
  - Cada face é adjacente a 6 outras;
  - Os vértices possuem grau 3.



Grades Hexagonais

- Grades hexagonais possuem propriedades interessantes e úteis, principalmente por serem mais “arredondados” do que os quadrados;
  - Os círculos são considerados eficientes porque englobam a maior quantidade de área por unidade de perímetro.
- Além disso, estruturas hexagonais são mais rígidas que estruturas retilíneas.
- A coordenada hexagonal do ponto (xh, yh) se refere ao centro do disco na linha horizontal xh e na coluna (60°) yh;
- A coordenada geométrica de tal ponto é uma função do raio **r** do disco, metade do diâmetro **d**, descrito anteriormente.
  - Distância entre dois vértices de uma grade triangular.

Grades Hexagonais

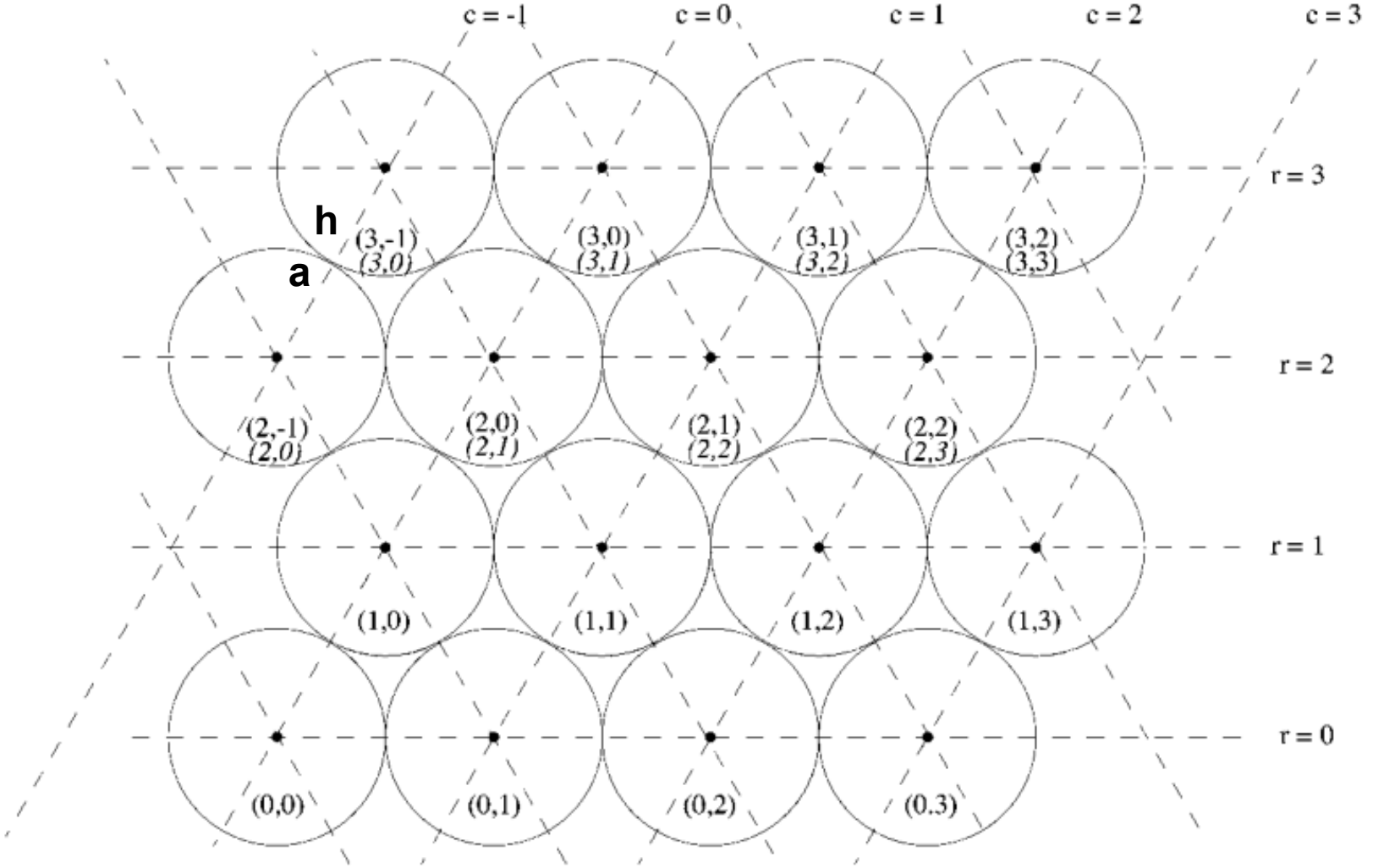
```
hex_to_geo
  (int xh, int yh, double r, double &xg, double &yg)
{
  yg = (2.0 * r) * xh * (sqrt(3)/2.0);
  xg = (2.0 * r) * xh * (1.0/2.0) + (2.0 * r) * yh;
}

geo_to_hex
  (double xg, double yg, double r, double &xh, double &yh)
{
  xh = (2.0/sqrt(3)) * yg / (2.0 * r);
  yh = (xg - (2.0 * r) * xh * (1.0/2.0) ) / (2.0 * r);
}
```

Grades Hexagonais

- A natureza linha/coluna do sistema de coordenadas hexagonal possui uma propriedade muito útil:
  - Podemos armazenar eficientemente um arranjo de hexágonos em uma matriz m[linhas][colunas];
  - Utilizando os *offsets* descritos para grades triangulares podemos encontrar os 6 vizinhos de cada hexágono.
- Entretanto há um problema:
  - No sistema de coordenadas hexagonal, os hexágonos definidos pelas coordenadas (hx, hy) formam um arranjo em formato de diamante, e não em formato de retângulo convencional;
  - Para resolver isto, são definidas coordenadas matriciais tal que (ax, ay) se refere à posição em um retângulo com ponto inferior esquerdo (0, 0) em uma matriz.

Grades Hexagonais



```
array_to_hex(int xa, int ya, int &xh, int &yh)
{
    xh = xa;
    yh = ya - xa + ceil(xa/2.0);
}

hex_to_array(int xh, int yh, int &xa, int &ya)
{
    xa = xh;
    ya = yh + xh - ceil(xh/2.0);
}
```

Problemas

- Fáceis:
  - *Ant on a Chessboard*;
  - *(2/3/4)-D Sqr/Rects/Cubes/Boxes ?*;
- Intermediários:
  - *Star*;
  - *Dermuba Triangle*;
- Difíceis:
  - *The Monocycle*;
  - *Airlines*.