Lecture 11

Random Walks

Hai-Qing Lin

Beijing Computational Science Research Center

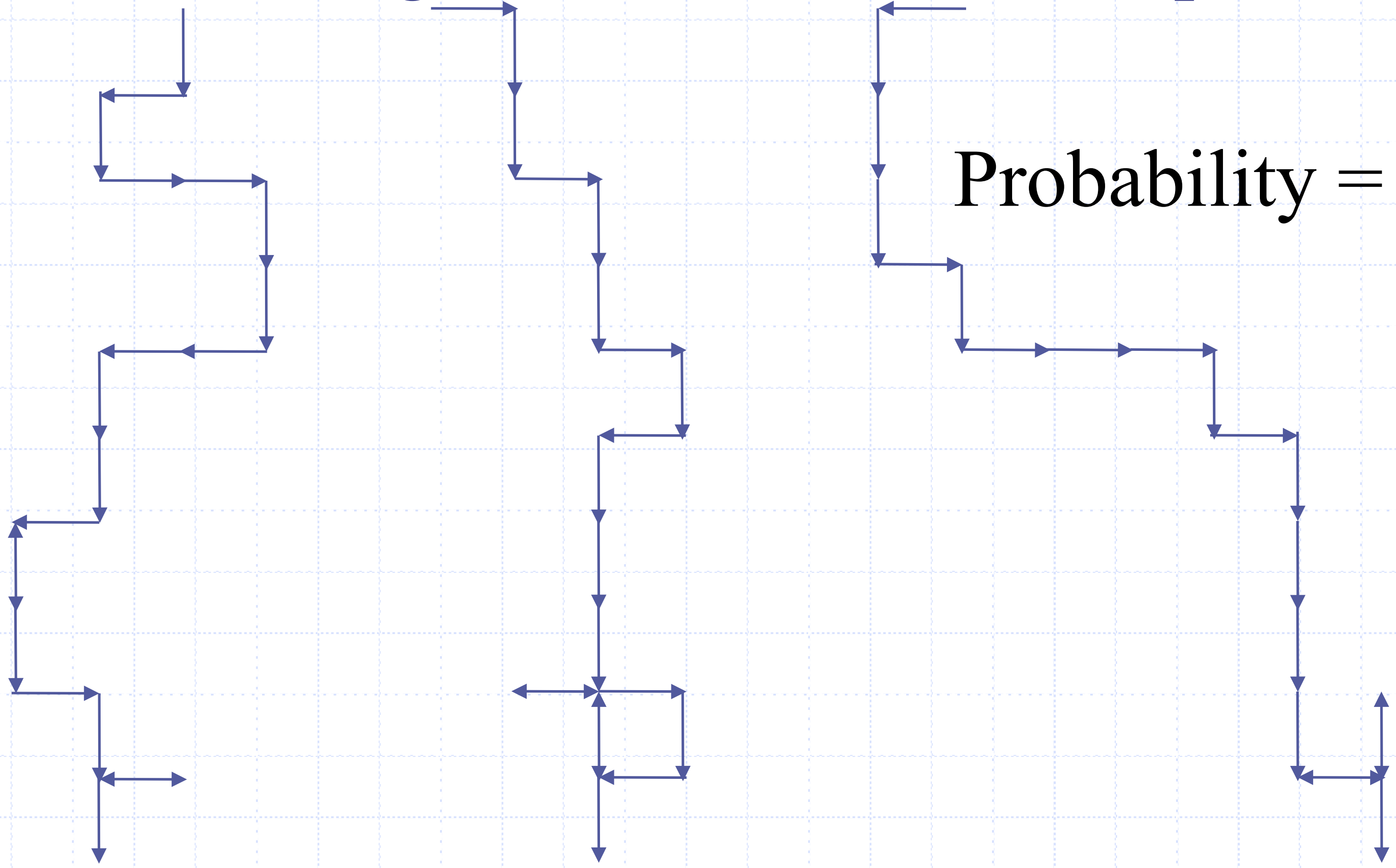
This PowerPoint Notes Is Based on the Textbook '***An Introduction to Computer Simulation Methods : Applications to Physical Systems***', 2nd Edition, Harvey Gould and Jan Tobochnik, Addison-Wesley(1996);
"A First Course in Computational Physics"; "Numerical Recipes";
"Elementary Numerical Analysis"; "Computational Methods in Physics and Engineering".

Introduction/Objective

- **Asymptotic scaling law:**
  for sufficiently ***large N***, one has
  $$\langle \Delta x^2(N) \rangle \sim N^{2\nu} \quad \text{for} \quad (N >> 1)$$
- For 1D, $\nu = 1/2$. (discussed before)
  For 2D and 3D, $\nu = ?$
- Other types of random walk and their applications.
- Difficult part is to generate enough walks with ***large N!*** **How to overcome this difficult?**

Application of Random Walks

1-D RW, e.g., The fall of a rain drop.



Probability = e^{-mgh}.

Examples of **the random path of a raindrop to the ground**.
Reasonable Step Probabilities are 0.1/0.6/0.15/0.15 for U/D/R/L.

Application of Random Walks

- **Two- and D-dimensional random walks**
  - RW on triangular or other type of lattice?
  - Polymers, Diffusions, etc.
  - Sample program (next page).
- **To check random number generator**
- **To solve differential equations**
- …

```
Subroutine Walk(p,N,xcum(),x2cum())

[Declaration, etc. ]
                                        ! x is a vector
     x = x_initial                      ! Initial position of walker for each trial
     DO istep = 1 to N
         [ get random number rnd ]

         p_small = 0.0
         p_large = 0.0
         DO idir = 1, directions

           p_large = p_large + probability(idir)    ! Probability of Walk
           IF p_left <= rnd < p_right then
              x(:) = x(:) + x_unit(:, idir)          ! unit walk
           END IF
           p_small = p_small + probability(idir)

         END DO

          ! collect data after every step
          CALL data(x,xcum(),x2cum(),istep)
      END DO

END Subroutine Walk
```

Modified Random Walks

- For simple RW, the walker has no "**memory**" of the previous step. What happened if the walker remember the nature of their previous step?
- Persistent random walk.
- Restricted random walk.
- Non-reversal random walk.
  - **Self-Avoiding random walk (SAW).**
  - …

Persistent Random Walks

- For persistent random walk, the transition or "jump" probability depends on the previous transition.
- e.g. step $N - 1$ has been made, step N is made in the same direction with probability α, $1 - \alpha$ in the opposite direction. This is different from the biased random walk!
- A persistent RW can be considered as an example of a multi-state walk in which the state of the walk is defined by the ***last transition***.

Restricted Random Walks

- Consider a 1D lattice with "**trap**" sites at $x = 0$ and $x = a$. A walker begins at site x_0 and takes unit steps to the left and right with equal probability, when it arrives at a trap, it vanishes. (e.g., potential)
- The average number of steps τ for the particle to be trapped (the first passage time) is given by

$$\tau = (2\boldsymbol{D})^{-1}\, x_0(a - x_0) \qquad \text{(used to obtain } \boldsymbol{D}\text{)}$$

D is the self-diffusion coefficient in the absence of the traps, and the average is over all possible walks.

Random Walk on Lattices with Traps

■	□	■	■	■	■	□	■	■	□	■	□	■	■	$n = 0$
1	0	1	1	1	1	0	1	1	0	1	0	1	1	
½	0	½	1	1	½	0	½	½	0	0	0	½	1	$n = 1$
½	0	½	¾	¾	½	0	¼	¼	0	0	0	½	½	$n = 2$

Example of the exact enumeration of walks on a given configuration of traps. The filled and empty squares denote regular and trap sites respectively. At step $N = 0$, a walker is placed a each regular site. The number at each site i represent the number of walkers W_i. Periodic boundary conditions are used. The initial number of walkers in this example is $m_0 = 10$. The mean survival probability at step $N = 1$ and $N = 2$ is found to be 0.6 and 0.475 respectively

Lattice Gas Model

- N distinguishable particles occupy $L \times L$ lattice sites, with particle density $\rho = N/L^2 < 1$
- Particle moves randomly to empty nearest neighbor sites **(no double occupancy)**
- Motivation: The diffusing particles are thermal vacancies whose density depends on the temperature
- Interest: the self-diffusion coefficient D of an individual particle
- Program **lattice_gas** on page 380

Lattice Gas Algorithm

- Record the initial position of each particle in an array.
- At each step, choose a particle and one of its nearest neighbor sites randomly.
- The particle moves to its nearest neighbor site if this site is empty or remain in its present position otherwise
- Define $\langle \Delta R(t)^2 \rangle$ as the net mean square displacement per tagged particle after t units of time, then the diffusion coefficient D is obtained as the limit $t \to \infty$ of $D(t)$,

$$D(t) = \frac{1}{2dt} \left\langle \Delta R(t)^2 \right\rangle$$

- Measure of ``**time**' ' : in one unit of time, each particle attempts one jump on the average.

Other Types of Random Walk

- Biased random walk.
- Energy transport in solids:
  hosts (atom absorption centre) and traps (exciton).
- Random walks on a continuum, probability density p(a), generation of the Gaussian distribution, etc.
- Random walks with steps of variable length.
- …

Non-reversal Random Walk

- Each step has to be a "**forward**" step.
- There are different ways of simulating it.

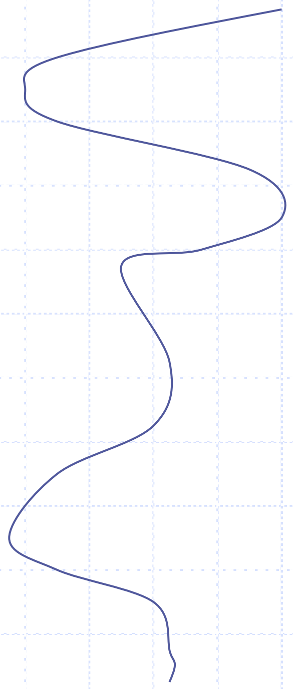
Self-Avoiding Walk (SAW)

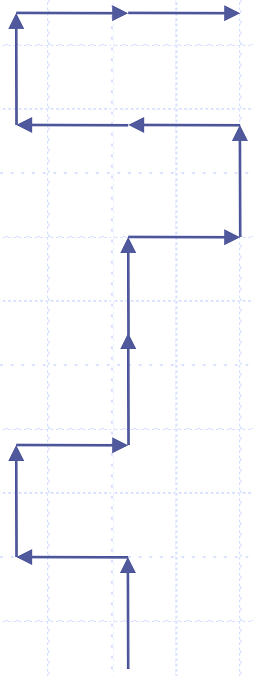
- ***No lattice site can be visited more than once***
- E.g. Polymer Physics (de Gennes)
- Difficulties in generating self-avoiding walks: how to generate **long** chain? (why?)

Application to Polymers

(a) Schematic illustration of a linear polymer in a good solvent.

(b) Example of the corresponding walk on a square lattice.


(a)

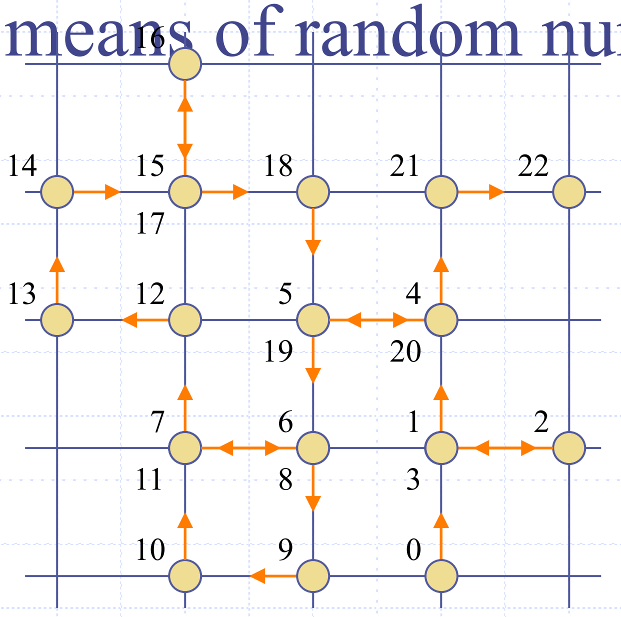

(b)

Self-avoiding Walks

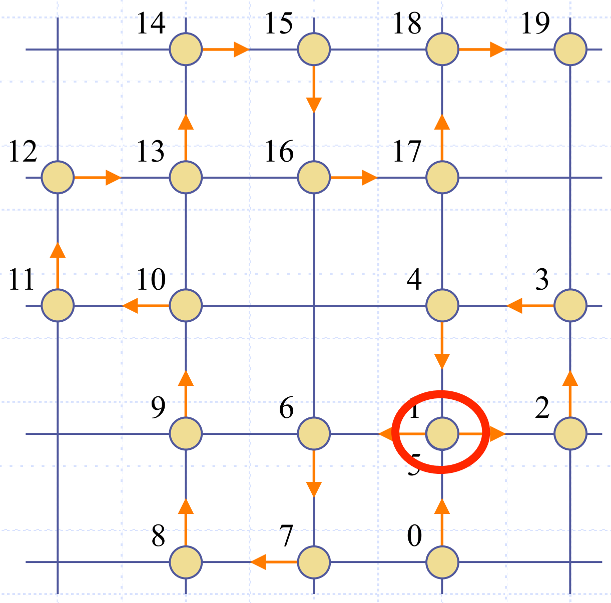
- In the *simple random walk* problem no constraint was placed on the possible sites the walker could visit. At each step the walker has completely forgotten where it came from. Each site is considered as an origin from which there are 2d (for a simple d-dimensional hyper cubic lattice) possible directions to go.
- For a self-avoiding random walk, the walker is not allowed to go to all sites.
- There are situations where there are no choice left at all!

Examples of Random Walks on the Square Lattice

- Sites are numbered in the order that they are visited.
- Bonds with arrows are selected consecutively by means of random numbers.

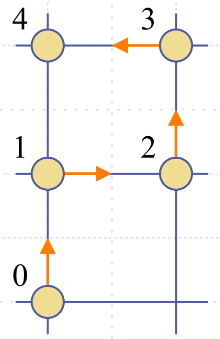


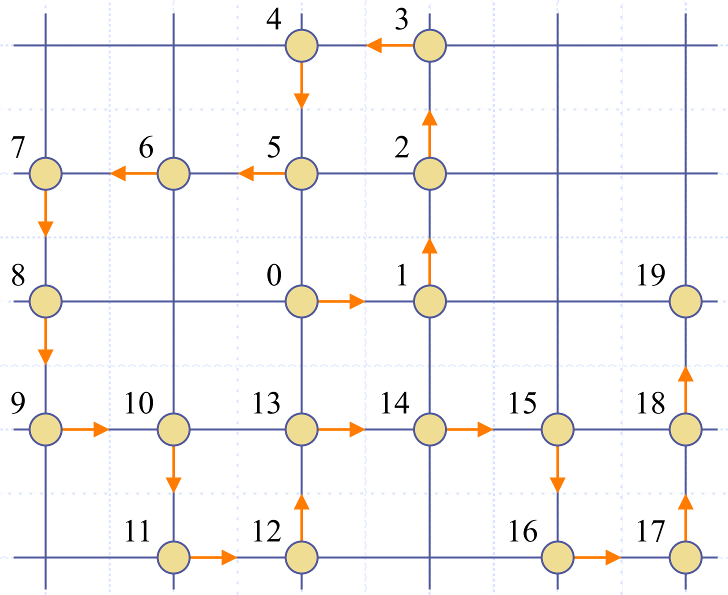
Unrestricted random walk



Non-reversal random walk

Examples of Self-avoiding Random walks





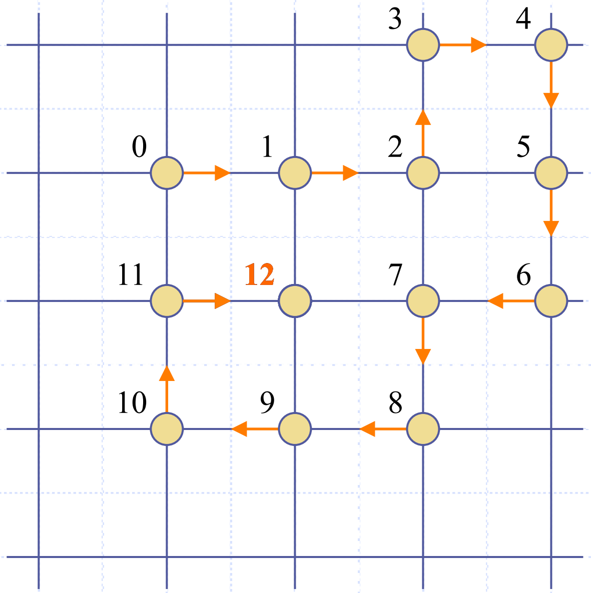
Seems rather straightforward

Another example

Starting from the origin, the random walker performed **12** steps. If the 13th step were to go to the right/up/down, or left then the walker would go to a previously visited site.

All next moves are NOT allowed.

In the first case the walker would visit a site twice. In the second case it would be an immediate reversal. As a consequence, the walk **terminates** with a length of 12 steps.



Asymptotic Scaling

$$\langle \Delta x^2(N) \rangle \sim N^{2\nu}$$

- For SAW $\nu = 3/4$ exactly in 2D.
- Approximately $\nu = 0.592$ in 3D.
- **Question in simulations:**
  For given D-dimensional lattice,
  - what is the **total number** of possible SAW?
  - how does the **simulation time** increase with system size?

Enumeration of All Possible SAW

This is a challenging problem!

Steps	**Number of walks on 2D Square Lattice**	
n	**Non-reversal**	**Self-avoiding**
0	1	1
1	4	4
2	12	12
3	36	36
4	108	100
5 ⋮ 34 (1989)	324 ⋮ 4×3^{33}	284 ⋮ 845 279 074 648 708
39 (1993)	4×3^{38}	113 101 676 587 853 932
51 (1996)	4×3^{50}	14 059 415 980 606 050 644 844

Simulation: Simple SAW Algorithm

Start with Simple RW algorithm (…), in which one must generate a random number as usual, even though one of more of the steps might lead to a self-intersection. If the next step does lead to a self-intersection, the walk must be terminated to keep the statistics correct!

Sample program (next page)

```
integer lattice(-N:N, -N:N)   ! 2D Lattice
integer xadd(1:4), yadd(1:4), rev(1:4)
```

SOME INITIAL SETUP

```
Do sample=1, No_of_samples
99 step=0
    x=0
    y=0
    lattice(x,y)=sample

    get random number
    idir = 4*rand + 1
    x = x + xadd(idir)
    y = y + yadd(idir)
    lattice(x,y) = sample
    step = step + 1
    reversal = rev(idir)        ! Make up rev(1:4)
```

```
199 get random number
        idir = 4*rand + 1
        if(idir .eq. reversal) goto 199
        x = x + xadd(idir)
        y = y + yadd(idir)
        if(lattice(x,y) .eq. sample) goto 99
        lattice(x,y) = sample
        step = step +1
        reversal = rev(idir)

if(step .lt. N) goto  199

      DO STATISTICS HERE

End Do
```

Method of Rosenbluth and Rosenbluth

Assign a weighting function $W_i(N)$ for the ith walk of N steps. We have $W_i(1) = 1$ and the following:

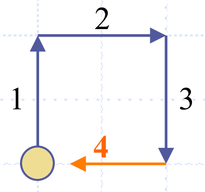
- If all three possible steps violate the self-intersection constraint, $W_i(N) = 0$ and a new walk is generated at the origin.
- If three steps are possible, $W_i(N) = W_i(N - 1)$.
- If only $m \leq 3$ are possible, $W_i(N) = (m/3)\, W_i(N - 1)$, and a random number is generated to choose one of the m possible steps.

$$\left\langle R^2(N) \right\rangle = \sum_i W_i(N) R_i^2(N) \Big/ \sum_i W_i(N).$$

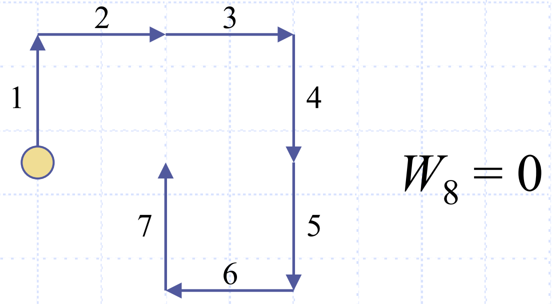
Examples of self-avoiding walks on square lattice

The origin is denoted by a filled circle.

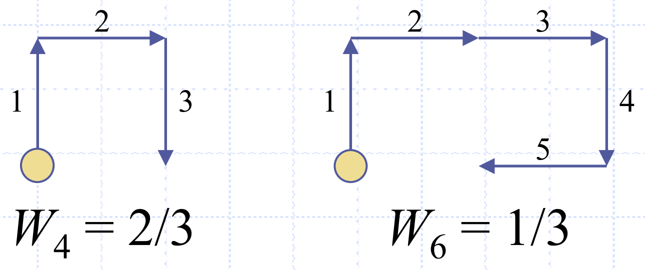
- A $N = 3$ walk. The **4th** step shown is forbidden.



- A $N = 7$ walk that leads to a self-intersection at the next step; the weight of the $N = 8$ walk is zero.



- Examples of the weights of walk in the enrichment method.



Simulation: Pivot Algorithm

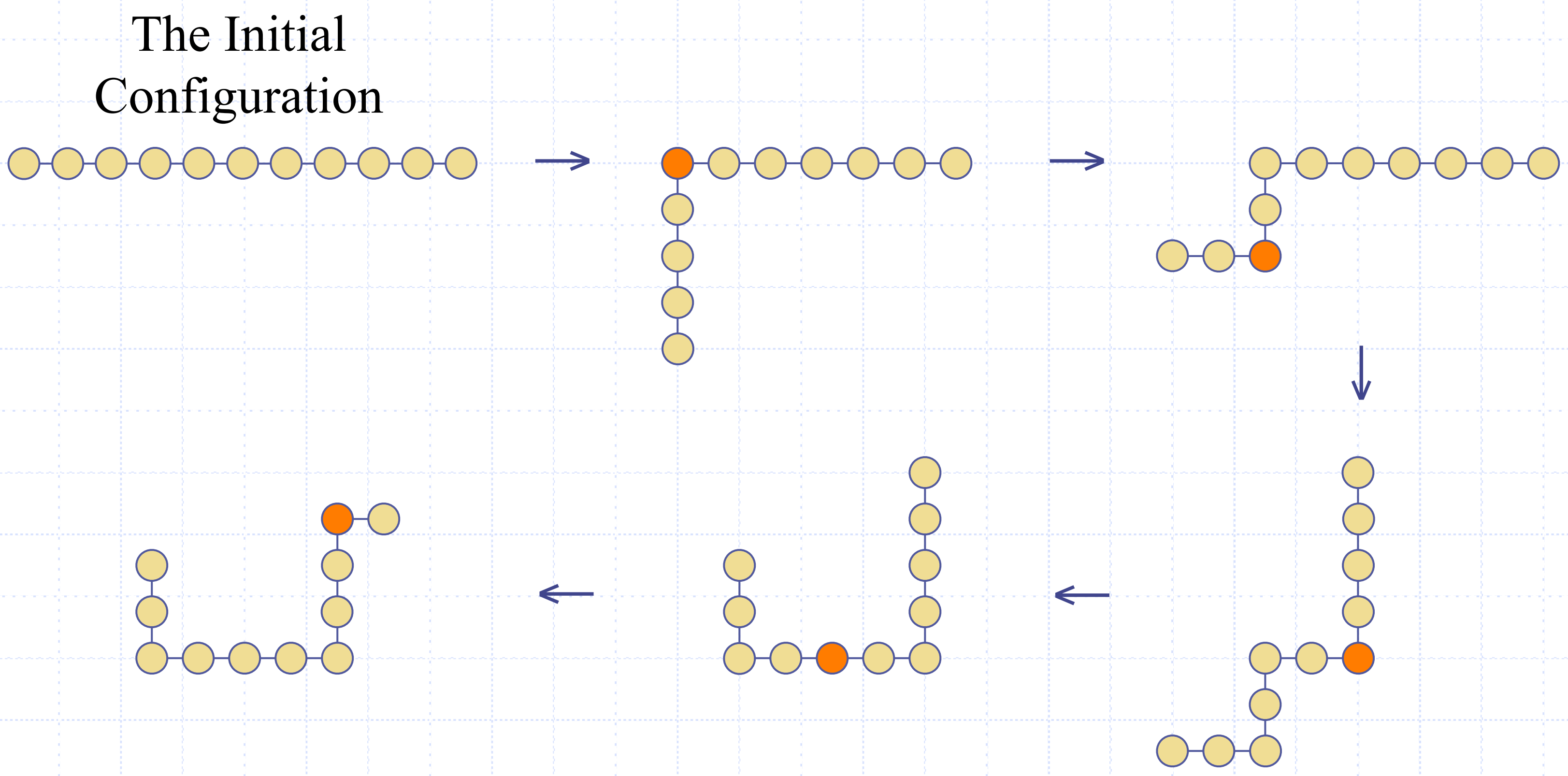
- The algorithms that we have discussed for generating self-avoiding random walks are all based on making ***local*** deformation of the walk (chain) for a given value of N, the number of bonds. The time $\boldsymbol{\tau_c}$ between statistically independent configuration is nonzero.
- The problem is that $\boldsymbol{\tau_c}$ increase with N as some power, e.g. $\tau_c \sim N^3$. This power law dependence of τ_c on N is called ***critical slowing down*** and implies that it becomes increasingly more time consuming to generate long walks.
- Need a ***global algorithm***, reduces ***correlation time*** τ_c.

Pivot Algorithm (e.g., square lattice)

- Choose an initial configuration.
- Select a site and one of the four direction at random.
- The shorter portion of the walk is rotated (pivoted) to this new direction by treating the walk as a rigid structure.
- If an end point is chosen, the previous walk is retained.
- This rotation is accepted only if the new walk is self-avoiding. Otherwise the old walk is retained.
- The shorter portion of the walk is chosen to save computer time.

Example of the first several changes generated by the pivot algorithm for a self-avoiding walk of $N = 10$ steps (11 sites).

The **red circle** denotes the pivot point.



Lecture 11: Review & Required

- Various random walks, **self-avoiding**, asymptotic behaviour.
- Application to physical systems.
- **Simulation techniques**.
- **Generating RWs and enumerations**.
- Global algorithm, etc.