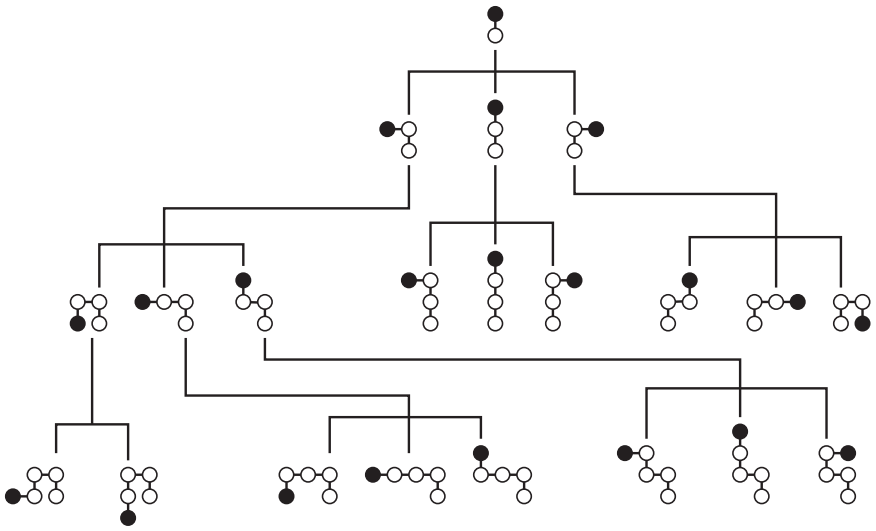
Physics 750: Exercise 15

Thursday, November 9, 2017

In this exercise, we undertake an *enumeration* of the self-avoiding walk (SAW) on the square and cubic lattices. Recall that enumeration is the strategy of exhaustively generating all SAW configurations up to some cutoff length—as opposed to *sampling* them stochastically, which is difficult to do efficiently in an unbaised way given the excluded volume constraint.

The configurations can be organized into a branching diagram with SAWs of equal length at the same level. Each node at level N has children corresponding to all the allowed walks of length $N+1$. For the square or cubic lattices, with coordination numbers $z = 4$ and $z = 6$, we can take advantage of the z-fold rotation symmetry by assuming that the single-step walk starts in a particular direction; the number of children at each node is between 0 and $z - 1$.



The number of configurations generated by random walks with no immediate backtracking (but no memory beyond the last step) forms an upper bound for C_N, the total SAW count at level N:

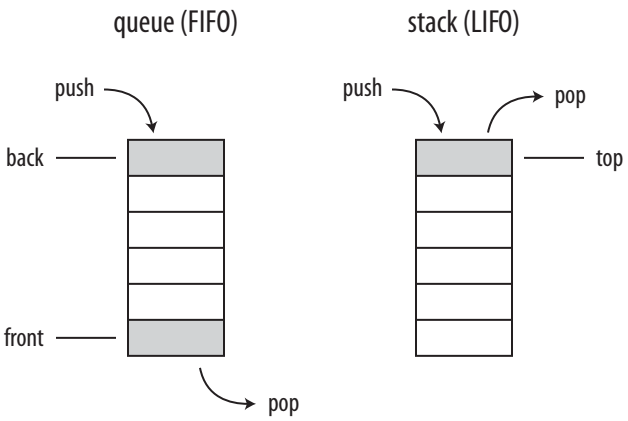
$$C_N < z(z-1)^{N-1}$$

A likely functional form for C_N is

$$C_N \sim \mu^{N-1} N^{\delta}$$

with $0 < \mu < z - 1$ and $|\delta| \ll 1$.

In class, we discussed two algorithms designed to systematically traverse the tree. These are referred to as *depth-first* search (DFS) and *breadth-first* search (BFS). The first goes as deep as possible down one branch before backtracking. The second generates all configurations at the highest incomplete level before moving on. We found that the two algorithms could be implemented in such a way that their code structure is almost identical, except for the choice of data structure. In one case, the intermediate configurations are stored in a first-in first-out (FIFO) list or *queue*. In the other a last-in first-out (LIFO) list or *stack* is used.

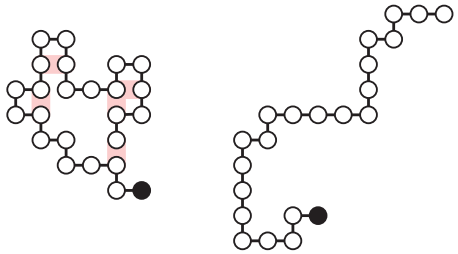


For a conventional random walk, the average end-to-end distance of the walk is characterized by $\langle r_N^2 \rangle \sim N^{2\nu}$ with the exponent taking the diffusive value $\nu = 1/2$. After the imposition of the excluded volume constraint, the behaviour is still powerlaw but with a modified exponent $\nu > 1/2$ (that depends on the spatial dimension). In the limit of many walk steps, the ratio of successive squared displacements behaves as

$$\frac{\langle r_{N+1}^2 \rangle}{\langle r_N^2 \rangle} \sim \left(1 + \frac{1}{N}\right)^{2\nu} \sim 1 + \frac{2\nu}{N}.$$

We can use this property to extract the value of ν from numerical data.

The standard SAWs we have encountered so far are athermal, and each configuration carries the same weight as all the others. If instead we assume that the walk is coupled to a heat bath at temperature kT and that there is an interaction of strength $-V$ between adjacent walks, then the weight associated with each configuration is e^{nx}, where n is the nearest-neighbour count and $x = V/kT$ is the dimensionless inverse temperature. A collapsed walk with $n = 5$ and a swollen walk with $n = 0$ are shown below.



1. Use the `curl` command to download from the class website everything you'll need for this exercise.

```
$ WEBPATH=http://www.phy.olemiss.edu/~kbeach/
$ curl $WEBPATH/courses/fall2017/phys750/src/exercise15.tgz -O
$ tar xzf exercise15.tgz
$ cd exercise15
```

   The `make` command creates an executable `saw2d_bfs`, which enumerates the SAWs of increasing length (up to some maximum length set by `size_t N = 14;` in `saw2d_bfs.cpp`) using the BFS algorithm.

```
$ make
$ ./saw2d_bfs
1 4
2 12
```

```
3 36
4 100
5 284
...
$ ./saw2d_bfs > saw.dat
$ gnuplot
gnuplot> set logscale y
gnuplot> plot "saw.dat" using 1:2 with linespoints, 4*3**(x-1)
gnuplot> unset logscale
gnuplot> plot "saw.dat" using 1:($2/(4*3**($1-1))) with linespoints
```

You can try to fit the number of configurations to the suggested functional form:

```
gnuplot> f(x) = C*a**(x-1)*x**b
gnuplot> C = 4; a = 3; b = 0.1
gnuplot> fit[6:] f(x) "saw.dat" using 1:2 via a,b,C
gnuplot> plot "saw.dat" using 1:2 with points, f(x), 4*3**(x-1)
```

2. So far, the program only counts the number of SAWs of each given length. Modify the code so that it also computes $\langle r_N^2 \rangle$, the mean squared displacement of the walk. Include this measurement as a third column of output. Your results should look like this.

```
$ ./saw_bfs
1        4     1
2       12     2.66667
3       36     4.55556
4      100     7.04
5      284     9.56338
...
```

A data file containing the ratios $\langle r_{N+1}^2 \rangle / \langle r_N^2 \rangle$ can be constructed using the `awk` utility. An estimate of ν can be determined by `gnuplot`.

```
$ ./saw_bfs | awk 'BEGIN { r2 = 0.0 } { if (r2 != 0.0) \
    print $1-1,$3/r2; r2 = $3 }' > ratios.dat
$ cat ratios.dat
1 2.66667
2 1.70833
3 1.54536
4 1.35843
5 1.31485
...
$ gnuplot
gnuplot> plot[0:][1:] "ratios.dat" using (1.0/$1):2
gnuplot> f(x) = 1.0 + 2.0*nu*x + c*x**2
gnuplot> fit[0:0.2] f(x) "ratios.dat" using (1.0/$1):2 via nu,c
gnuplot> replot f(x)
```

3. The BFS algorithm is implemented using a `queue` data structure. Create a nearly identical program `saw2d_dfs.cpp` that implements the DFS algorithm by switching out the `queue` for a `stack`. (Remember that the `stack` also has `push` and `pop` methods, but rather than `front` and `back`, it has only `top`.)

   The correct header is

   ```
   #include <stack>
   using std::stack;
   ```

   Check that your old and new programs produce identical output.

   ```
   $ ./saw2d_bfs > bfs.dat
   $ ./saw2d_dfs > dfs.dat
   $ diff bfs.dat dfs.dat
   ```

   A cute trick is that the DFS program can be generated automatically using the search and replace functionality of the `sed` utility.

   ```
   $ sed -e 's/queue/stack/' -e 's/.front()/.top()/' \
         -e 's/.back()/.top()/' saw2d_bfs.cpp > saw2d_dfs.cpp
   ```

4. Write a program `saw3d_bfs.cpp` that computes all SAWs on a three-dimensional cubic lattice up to length $N = 8$. You should be able to reproduce the following results.

   ```
   $ ./saw3d_bfs
   1        6    1
   2       30    2.4
   3      150    3.88
   4      726    5.55372
   5     3534    7.2343
   6    16926    9.07054
   7    81390   10.8972
   8   387966   12.8451
   ```

   Can you estimate the exponent ν? It should be $\nu \approx 7/12$ in the large N limit.

5. Try implementing the interaction scheme discussed in the introduction. This involves replacing

$$\langle r_N^2 \rangle = \frac{1}{C_N} \sum_{\sigma_N} r^2(\sigma_N)$$

   with the Boltzmann-weighted average

$$\langle r_N^2 \rangle = \Big(\sum_{\sigma_N} r^2(\sigma_N)\, e^{xn(\sigma_N)} \Big) \Big/ \Big(\sum_{\sigma_N} e^{xn(\sigma_N)} \Big).$$

   Here, x is the reduced inverse temperature, and σ_N ranges over all SAWs of length N. $r^2(\sigma_N)$ and $n(\sigma_N)$ are the squared displacement and number of nearest neighbours for a particular walk. Plot $\langle r_N^2 \rangle$ for various N over the range $-10 < x < 10$.