Testbed for the 1d hungry walker model. Map the process onto a Markov process with state space (i,r), where i is the position of the walker, r is the position of the food front. (Such that i=r means the walker is just at the food front and will have jump rates that are biased.) We write the bias towards food as 1-ϵ.

In[39]:=
```
maxSteps = 12;
```

In[40]:=
```
lattice = Range[-maxSteps, maxSteps];
offset = 1 - Min[lattice];
dim = Length[lattice];
```

The trick to deal with this: sparse arrays. Otherwise, the matrices become too big to handle, and we're storing zeros mostly for nothing. But this means we need to map (i,r) to a onedimensional index. Helper functions:

In[43]:=
```
idx[i_, r_] := (i + offset - 1) dim + (r + offset);
rx[idx_] := Mod[idx - 1, dim] - offset + 1;
ix[idx_] := Quotient[idx - 1, dim] - offset + 1;
```

Helpers to map back and forth between a matrix and its functional representation: if T is a function taking four arguments (i,r,id,rd), then mat[T] makes a matrix out of it. Reversely, ind[Tmat] takes such a matrix and gives a function object that we can acces with four arguments.

In[46]:=
```
mat[T_] := Table[T[ix[idx], rx[idx], ix[idxd], rx[idxd]], {idx, dim ^ 2}, {idxd, dim ^ 2}]
```

In[47]:=
```
ind[Tmat_] := Function[{i, r, id, rd}, Tmat[[idx[i, r], idx[id, rd]]]]
```

We need n-th powers of the T matrix of the stochastic process. Trick: break down $T^{2n} = T^n.T^n$ for integer n and $T^{2n+1} = T^n.T^n.T$, and perform the exponentiation recursively, storing all intermediate results. This way, we will save ourselves repeated reevaluation of the matrix powers.
The first power is set tothe T matrix itself.
(mat here is the bottleneck: should the initial matrix already be too big to story as a non-sparse array, we have to think about creating it as a sparse array right in the beginning.)

In[60]:=
```
ClearAll[TtoN];
TtoN[n_] := With[{n2 = Quotient[n, 2]},
   If[2 n2 == n, (TtoN[n] = TtoN[n2].TtoN[n2]), (TtoN[n] = TtoN[n2].TtoN[n2].TtoN[1])]];
TtoN[0] = SparseArray[mat[Function[{i, r, id, rd},
      KroneckerDelta[i, id] KroneckerDelta[r, rd]]]];
TtoN[1] = SparseArray[mat[Function[{i, r, id, rd},
      1 / 2 KroneckerDelta[id, i - 1] KroneckerDelta[rd, r] +
       (1 - ϵ) KroneckerDelta[id, i - 1] KroneckerDelta[rd, r - 1] KroneckerDelta[id, rd] +
       ϵ KroneckerDelta[id, i + 1] KroneckerDelta[rd, r] KroneckerDelta[id, rd] +
       1 / 2 KroneckerDelta[id, i + 1] KroneckerDelta[rd, r] UnitStep[r - 2 - i]]]];
```

Test: probability to be at i=n with r=n, after t=n steps. This should be $(1-\epsilon)^n$.
Note our finite lattice: since the walker in n steps can at most go from i=0 to i=n or i=-n, we are good if n is less or equal the lattice extension, if the process starts at i=0.

In[52]:=
```
Table[ind[TtoN[n]][n, n, 0, 0], {n, maxSteps}]
```

Out[52]= $\{1-\epsilon, (1-\epsilon)^2, (1-\epsilon)^3, (1-\epsilon)^4, (1-\epsilon)^5, (1-\epsilon)^6, (1-\epsilon)^7, (1-\epsilon)^8, (1-\epsilon)^9, (1-\epsilon)^{10}, (1-\epsilon)^{11}, (1-\epsilon)^{12}\}$

This is the formula for which Sebastian has derived a closed form, it would be nice to check!

In[57]:=
```
With[{n = maxSteps / 2}, Table[ind[TtoN[2 n]][2 (n - k), 2 (n - k), 0, 0], {k, 0, n}]] // Simplify
```

Out[57]= $$\left\{(-1+\epsilon)^{12}, \frac{11}{2}(-1+\epsilon)^{10}\epsilon, \frac{9}{8}(-1+\epsilon)^8\epsilon(1+10\epsilon), \frac{7}{16}(-1+\epsilon)^6\epsilon\left(1+8\epsilon+24\epsilon^2\right), \frac{5}{128}(-1+\epsilon)^4\epsilon\left(5+30\epsilon+84\epsilon^2+112\epsilon^3\right), \frac{3}{256}(-1+\epsilon)^2\epsilon\left(7+28\epsilon+60\epsilon^2+80\epsilon^3+56\epsilon^4\right), \frac{\epsilon\left(21+42\epsilon+56\epsilon^2+56\epsilon^3+40\epsilon^4+16\epsilon^5\right)}{1024}\right\}$$

In[59]:=
```
Table[ind[TtoN[2 n]][0, n, 0, 0], {n, maxSteps / 2}]
```

Out[59]= $$\left\{(1-\epsilon)\epsilon, \frac{1}{2}(1-\epsilon)^2\epsilon, \frac{1}{4}(1-\epsilon)^3\epsilon, \frac{1}{8}(1-\epsilon)^4\epsilon, \frac{1}{16}(1-\epsilon)^5\epsilon, \frac{1}{32}(1-\epsilon)^6\epsilon\right\}$$