**2023 DGIST Summer Intern Report  
Quantum Many-body Theory Group. with professor Aram Kim**

202111178 조수호

목차

1. Pure Ising Model
2. Modeling
3. Advanced technic
4. Magnetization graph
5. Cluster Ising Model
6. Algorithm
7. Plot all according to variable lattice size
8. Fractal Structure
9. Magnetic Susceptibility
10. Time Complexity with vector and basic array
11. Disordered Ising Model
12. Basic concept
13. Advanced goal
14. Improvement
15. Reference and code
16. **Pure Ising Model**
17. **Modeling**

Ising Model은 통계역학적인 상전이 모델로서 물체의 자성을 나타내는 간단한 격자이다. 격자에는 -1 혹은 1로 spin값이 결정되며 이는 IsingModel의 알고리즘에 따라 결정된다. IsingModel이 흥미로운 점은 다른 상전이 모델에도 적용할 수 있다는 것이다. 예를 들면 물에서 수증기로 변하는 상전이(liquid to gas) 등이 있다.

Pure Ising모델에서는 아래와 같은 알고리즘을 이용하여 c++에 코딩하였다.

1. Lattice 위의 무작위 한 지점인 k를 선택한다.
2. k spin이 flip이 되었을 때의 hamiltonian의 변화량을 비교하여, k spin을 flip할 지 정한다. Pure IsingModel에서 Hamiltonian은 이다. 값이 1보다 크면 무조건 flip하고 1보다 작으면 의 확률로 k spin을 flip한다. 이때, 이고 J의 값은 변할 수 있으나 이번에는 J = -1로 고정한다. J = -1 일 때는 강자성체의 경우로 알려져 있다.
3. 위 1~2 의 과정을 매우 많이 반복한다. 이는 MonteCarlo algoritm의 원리에 따른 것으로, 무수히 많은 시행을 한다면 이 사건은 평균에 수렴하는 것이 된다. 예를 들어 한 변의 길이가 1인 정사각형과 지름의 길이가 1인 원을 중심이 일치하도록 그려둔다. 이후 다트를 무작위로 아주 많이 던진다. 이 경우 정사각형 내부에 꽃힌 다트의 개수와 원 내부에 꽃힌 다트의 개수를 비교하여 원주율을 구할 수 있다 (). 마찬가지로 IsingModel의 경우에도 아주 많은 횟수의 1~2과정을 시행한다면, 이는 가능한 경우가 각 확률에 따라 고려되기 때문에 실제로 lattice가 아주 큰 물체의 magnetization을 예상할 수 있다.
4. 의 식으로 평균 magnetization을 구할 수 있다. 이때 는 MonteCarlo number이고, 로 N은 lattice number이다.
5. **Advanced** **techenic**

위와 같은 알고리즘을 그대로 코드에 구현하면 한 MonteCarlo step에 따라 소요되는 연산시간이 중요하다. 왜냐하면 이 시간이 줄어들수록 더 많은 MonteCarlo step을 한정된 시간 안에 할 수 있기 때문이다.   
위의 알고리즘에서 연산시간을 결정하는 주요 요인은 random generate와 if문이 있다. Random generate는 k site를 결정하는 과정에서 한 번, 의 확률로 k spin을 flip할 지 결정하는 과정에서 한 번 쓰인다. 이 두 random generate를 하나로 합칠 수 있다. Random generate를 rand\_float(0, N)으로 호출하면 0에서 N(lattice size)사이의 실수를 저장할 수 있다. 이 값을 int로 바꿔주면 k site를 random하게 구할 수 있고, 이 값을 이 값을 int로 바꿔준 값으로 빼면 0~1사이의 실수를 저장할 수 있어서 두 번째 random generate에 사용할 수 있다. 두 번째 random generate에 대해 자세히 설명하면 다음과 같다. 값이 0~1 사이 실수보다 크면 k spin을 flip하면 된다. 이 1보다 큰 경우에는 0~1사이 실수 값보다 무조건 크기 때문에 알고리즘에 맞게 1의 확률로 flip하는 것이다. 이 1보다 작은 경우 0~1사이 실수 값보다 이 클 확률이 이기 때문에 알고리즘에 맞게 확률로 k spin이 flip한다.  
다음으로 if문의 수를 줄여야 한다. if문의 경우 이 random\_float(0,1)보다 작은지 확인하는 경우에는 무조건 사용되어야 한다. 그러므로 다른 경우에 if문 사용을 최소화해야 한다. 예를 들어 k site의 인접 site를 확인할 때 spin lattice의 한 변의 길이를 초과하는 인접 site의 경우 if문을 써야 한다고 생각할 수 있다. 그러나 나머지 연산자(c++의 경우 ‘%’)를 쓰면 불필요한 if문을 줄일 수 있다.

1. **Magnetization graph**

위와 같은 알고리즘으로 pureIsingModel을 c++에 구현하였을 때 결과 그래프는 아래와 같다. 빨간 점선은 데이터 10개의 표준편차를 나타낸 것이고, T값은 높은 값에서 낮은 값으로 내려가며 코드를 실행하였고, T가 최대값에서 MonteCarlo Step을 5000\*lattice\_size 만큼 실행하여 시작 T에서의 평균값에서 magnetization 계산을 시작하였다.

|  |  |
| --- | --- |
|  |  |
|  |  |

Pure IsingModel의 알려진 <m> - T graph는 아래와 같다.

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

Reference 추가

1. Cluster Algorithm
2. Algorithm  
   Cluster Algorithm을 Ising Model에 적용한 Cluster-Ising Model은 결과적으로 계산되는 결과는 유사하지만, 이 결과를 만드는 과정에서 차이가 있다. Pure Ising Model은 자연에서 발생하는 상황처럼 온도와 에너지에 대한 특정 확률로 각 site에 있는 spin들이 flip된다. 반면에, Cluster-Ising Model에서는 온도 조건에 따른 확률로 Cluster을 형성하고 Cluster의 원소 spin들의 flip이 한 번에 일어난다. Pure Ising Model에서는 온도가 높으면 spin이 flip될 확률이 높아서 spin들이 정렬되어 있지 않은데, Cluster-Ising Model에서는 Cluster에 포함된 spin들의 수가 작은 경우가 많아서 spin 방향의 무작위성이 나타난다. 온도가 낮은 경우에 Pure Ising Model에서는 spin이 flip될 확률이 낮아서 spin들이 한 방향으로 정렬되어 있고, Cluster-Ising Model에서는 Cluster에 포함되는 spin들이 수가 많을 확률이 높아서 대부분의 spin들이 같이 flip되서 spin들이 한 방향으로 정렬된다.

Cluster-Ising Model의 알고리즘은 아래와 같다.

1. Spin lattice의 무작위 site j를 고른다.
2. Pocket과 Cluster에 j를 넣고 Pocket이 공집합이 될 때까지 아래 3. ~ 6. 과정을 반복한다.
3. Pocket의 원소 중 하나인 k를 무작위로 선택한다.
4. k site의 인접한 4 방향의 site l 에 대해 아래 5. 과정을 시행한다.
5. l spin이 ‘k spin과 같은 방향을 가지고’, ‘Cluster의 원소로 없으며’, ‘의 확률을 성공’ 한다면, Pocket과 Cluster에 l 을 추가한다.
6. Pocket에서 k원소를 제거한다.

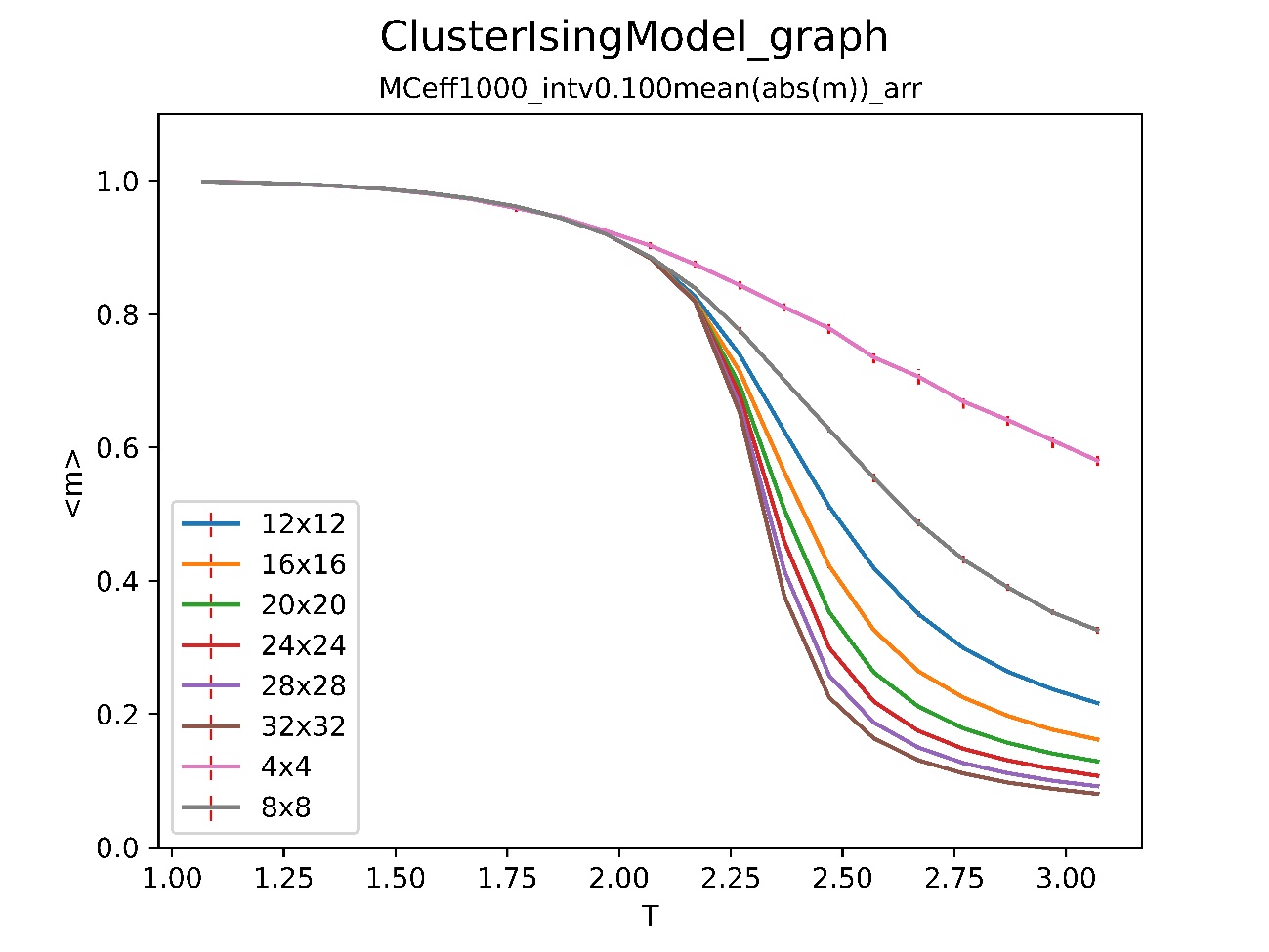
Magnetization 값을 구하는 방법 중 Pure Ising Model과 다른 점은 각 MonteCarlo Step에서 m값을 구했으면 이 값에 절대값을 씌워야 한다.

즉, 이다.

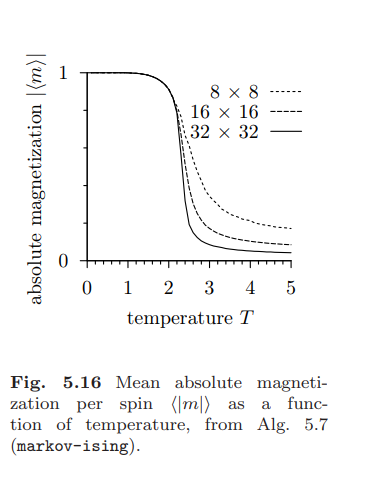
M에 절대값을 취해야 하는 이유는 Cluster의 원소로 있는 spin들이 무조건 flip되기 때문에 각 MC step의 lattice의 spin들이 얼마나 정렬되어 있는지를 봐야 하는 거지, 이를 평균 내어 버리면 온도가 낮은 경우에도 이 0에 수렴하기 때문이다.

1. Plot all according to variable lattice size

아래는 lattice를 다르게 하여 Cluster-Ising Model의 - T 그래프를 그린 것이다. Pure Ising Model에 비해 smooth한 curve를 볼 수 있다.

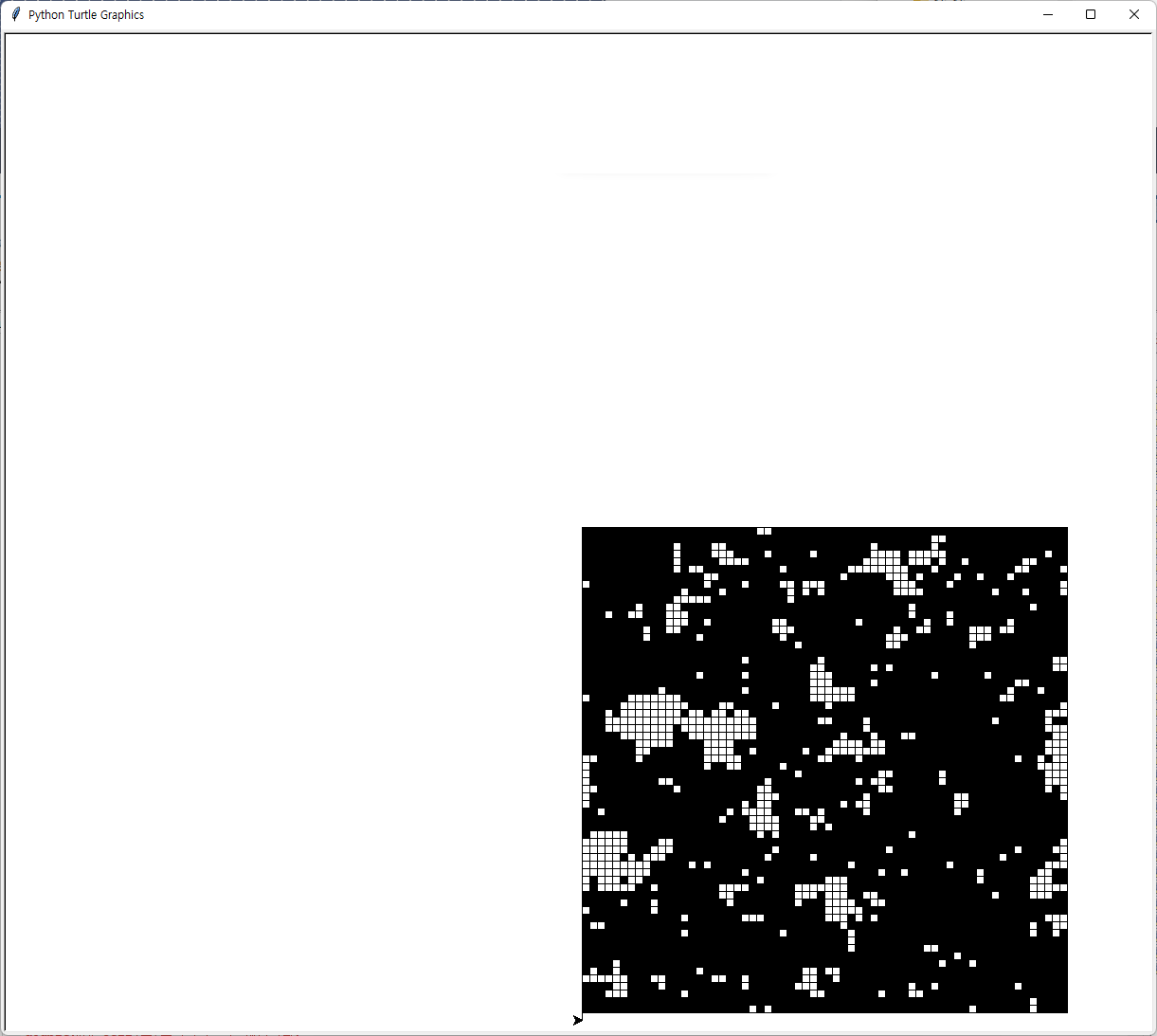
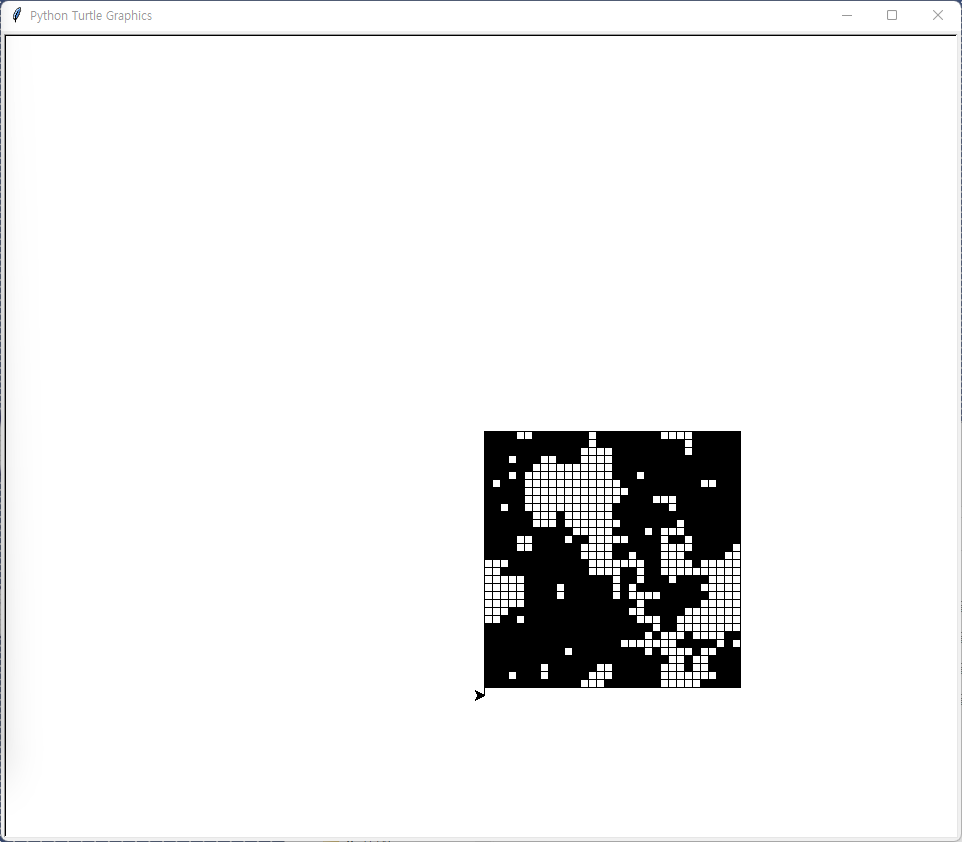
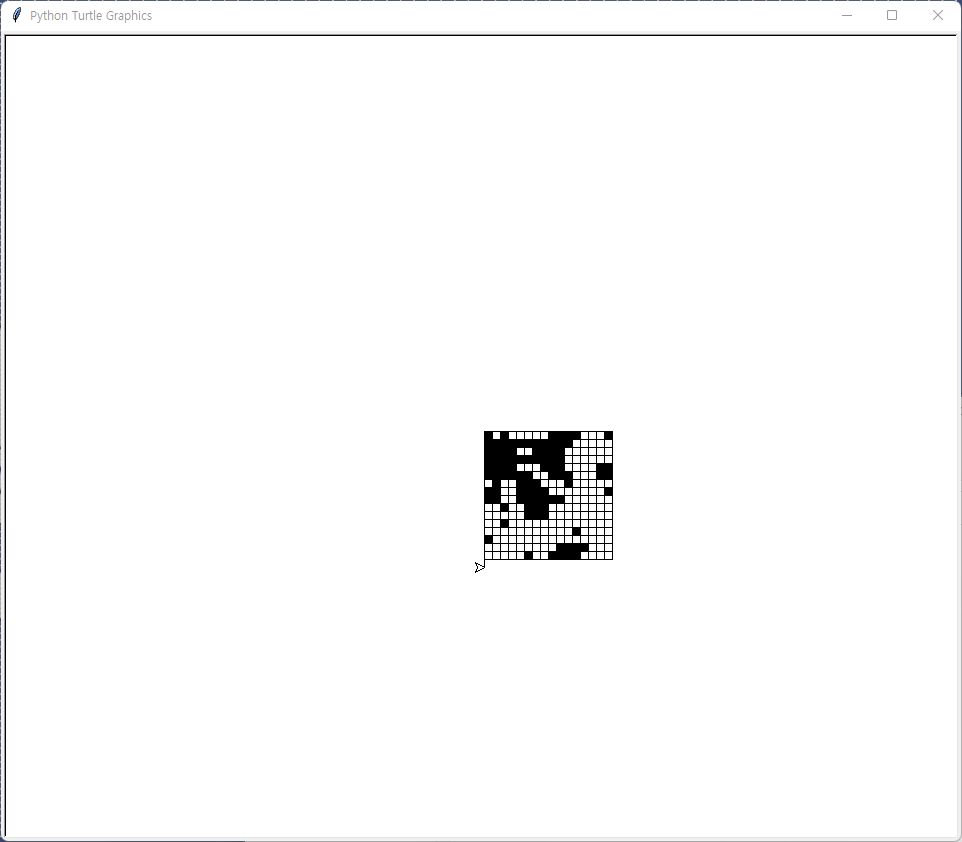


이는 알려진 그래프와 유사한 모습을 띈다.



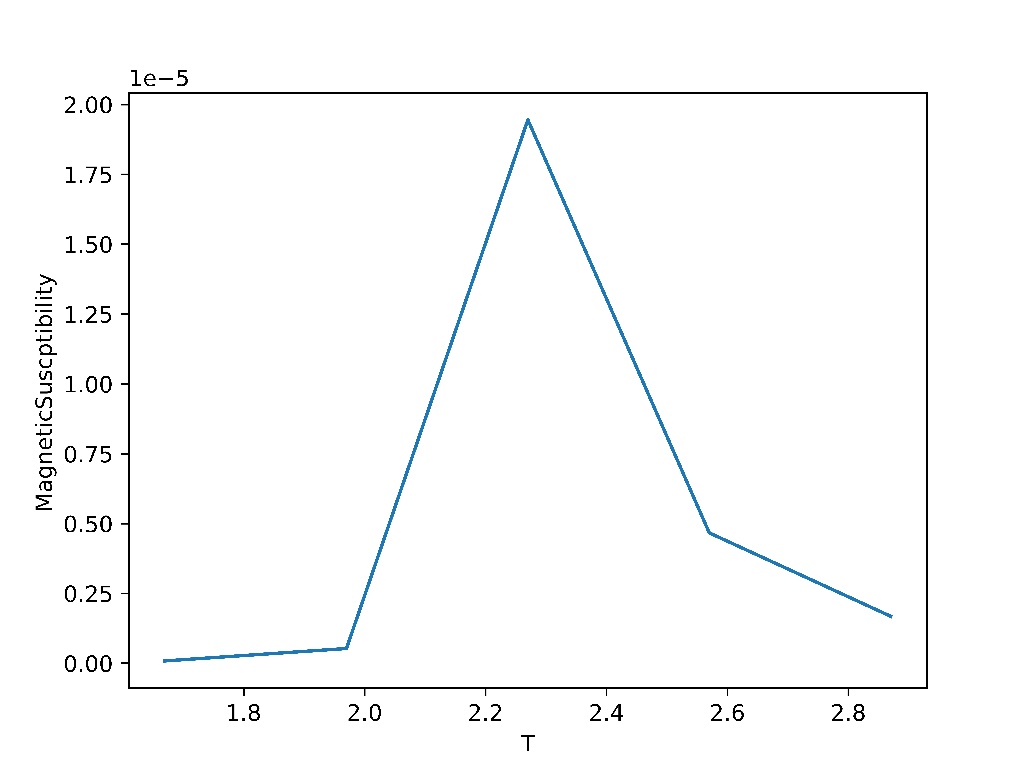
1. Fractal structure

Ising Model에는 critical point를 정의할 수 있다. Critical point는 T에 따라 <m>값값 변화시킬 때 <m>값이 급변하는 T point이다. 2차원 Ising Model에서 critical point는 약 2.27로 계산된다. Ising Model의 한 가지 흥미로운 점은 critical point부근에서 spin lattice는 Fractal structure를 띈다는 것이다. 이는 correlation length가 발산하기 때문으로, lattice의 length scale이 급변하면서 작은 구조와 큰 구조가 공존하게 되어 Fractal Structure을 볼 수 있다. 아래는 Cluster-Ising Model을 연산 중에, T = 2.27에서 spin lattice를 출력한 그림이다. Spin 값이 1이면 검은색, -1이면 하얀색으로 격자를 채웠다. 순서대로 16x16 , 32x32, 64x64 spin lattice이다. 작은 lattice에서 보이는 구조가 큰 구조에서 보이는 구조를 구성한다.



1. Magnetic susceptibility

Ising Model로 Magnetization만 계산할 수 있는 건 아니다. 다양한 물리량 중 하나로 Magnetic Susceptibility가 있다. 이는 로 나타낼 수 있다. Magnetic Susceptibility는 에서 가장 큰 값을 가지고, 와 멀어질수록 작은 값을 가진다. 아래는 Cluster-Ising Model로 계산한 Magnetic Susceptibility – T 그래프이다.



1. Time complexity with vector and basic array

Ising Model에서는 MonteCarlo step을 높일수록 표준편차가 줄어들기에, 해당하는 값이 평균에 가까울 가능성이 커진다. 따라서 많은 양의 연산을 해야 할 필요성이 있고 한정된 resource를 가지고 계산을 하려면 코드의 최적화가 필요하다. 앞서 pure Ising Model에서는 if문과 random함수의 최소화를 통해 이를 실행하였다. Cluster Ising Model에서는 Pocket size가 0이 될 때까지 반복하는 반복문이 있어서, 반복문 안의 코드가 얼마나 큰 시간복잡도로 시행되는지가 중요한 요소이다. Whille문 안의 코드들은 주로 Pocket과 Cluster의 업데이트, Pocket과 Cluster에 원소를 추가할 건지 결정 등의 경우가 있다. 이 두 경우의 시간복잡도는 Pocket과 Cluster의 자료형에 밀접한 관련이 있다. 이 글에서 제시하는 자료형은 c++에서 사용하는 ‘vector’ 자료형과 기본 array 자료형이 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pseudo Code | Vector | | Array | |
| is\_empty(Pocket) | O(1) | Pocket.empty(); | O(1) | Pocket\_size > 0; |
| k <- Random\_choice(Pocket) | O(1) + Rand\_time | k = Random\_choice(Pocket.size());  Random\_choice는 1~n 을 무작위 정수를 리턴 | O(1) + Rand\_time | int k = distribution3(engine3) \* Pocket\_size; distribution3(engine3)은 float type의 0~1 generate |
| l in cluster l은 k주변 인덱스 | O(n) | find(Cluster.begin(), Cluster.end(), l ) | O(1) | Cluster[l] == 1 |
| Pocket, Cluster + {j} | O(1) | Pocket.push\_back(j);  Cluster.push\_back(j); | O(1) | Pocket[l] = -1; Pocket\_size += 1;  Cluster[l] = -1; Cluster\_size += 1; |
| Pocket.remove({k}) | O(n) | Pocket.erase( remove(Pocket.begin(), Pocket.end(), k), Pocket.end() ) | O(1) | Pocket[k] = 1; Pocket\_size -= 1; |
| SpinMatrix.flip(Clister) | O(n) | for ( i : Cluster){      SpinMatrix[i] \*= -1;  } | O(n) | for(int i = 0 ; i < N ; i++){      SpinMatrix[i] \*= Cluster[i];  } |