

1. **연결리스트 구현을 위한 구조체를 생성한다. 시행착오를 거쳐 데이터를 담을 어레이, 원본 데이터 참조를 위한 인덱스, 연결리스트를 위한 링크로 구성했다.**
2. **각 medoids의 가장 첫 노드들은 헤드 노드로 작동한다. 헤드의 어레이는 NULL이고, 인덱스는 해당 클러스터에 속하는 데이터들의 개수를 나타낸다.**
3. **전체 데이터 개수, 특성 개수, 클러스터 개수, 원본 데이터를 담을 포인터, 클러스터들을 담을 포인터, 각 클러스터의 마지막 노드를 가리키는 라스트를 담을 포인터, medoids 중 변화의 개수를 세는 체인지, 반복횟수를 세는 이터레이션은 자주 쓰이므로 전역변수로 선언한다.**
4. **데이터를 읽어온다. 파일을 잘 닫아준다. 이 때 얻어온 클러스터 개수를 이용하여 클러스터와 라스트에 메모리를 할당한다.**
5. **임의의 데이터를 뽑아서 medoid로 설정한다. 이 때 한 번 뽑았던 데이터가 중복되어 설정되지 않도록 설계한다. 헤드들의 인덱스는 1로 설정하고 설정된 medoid들을 헤드에 연결시킨다. 편의를 위해 설정이 끝난 medoids를 출력한다.**
6. **각 클러스터에 데이터를 할당한다. medoids로 사용된 데이터는 건너뛴다. 각 데이터를 가지고 모든 medoids와의 거리를 계산한 뒤 그 중 가장 가까운 클러스터에 추가한다. 이 때도 편의를 위해 거리와 할당 결과를 출력한다.**
7. **각 클러스터 내의 데이터들을 가지고 클러스터 내 다른 데이터들과의 거리를 모두 더한 값을 계산한 뒤, 그 값이 가장 작은 데이터를 새로운 medoids로 설정하고 나머지 데이터들은 모두 메모리를 해제하여 반복을 수행할 준비를 한다. 모두 끝난 뒤, medoids의 변동이 0보다 크다면 다시 6번으로 돌아가 반복을 수행한다. 편의를 위해 새롭게 할당된medoids를 출력한다.**
8. **변동이 없어질 때까지 반복한다.**
9. **이해를 원활히 하기 위해 각 클러스터를 출력한다. 최종 medoids를 출력한다. 그리고 클러스터별 데이터를 출력한다.**
10. **아웃풋 파일을 생성한 뒤 분류가 끝난 데이터를 출력한다.**
11. **각 클러스터 별 메모리를 모두 해제한다. 클러스터를 해제한다. 라스트를 해제한다. 데이터를 해제한다.**