**Programming Assignment #1**

**학번: 2017041930  
이름: 김선동**

Index  
1. Environment

2. Instructions for compiling my source codes

3. Summary of my implementation

4. Testing

**1. Environment**

OS는 linux 18.04, language는 c++를 사용하였다.

주석을 전부 한글로 달아 놓았기 때문에, 주석을 보려면 한글팩이 정상적으로 깔려 있어야 합니다.

**2. Instructions for compiling my source codes**

내가 사용한 컴파일러는 다음과 같다.



아래와 같이 컴파일하면 apriori.exe 실행파일을 얻을 수 있다.

  
또는 makefile을 첨부하였으니 make 명령어만으로 실행 파일을 생성할 수 있다.  
텍스트이(가) 표시된 사진

자동 생성된 설명

아래와 같이 실행하면 output.txt에 정상적으로 결과물을 얻을 수 있다.



**3. Summary of my implementation**

텍스트이(가) 표시된 사진

자동 생성된 설명

첫째로, main문에서 요구사항에 맞게 3개의 인자들을 입력 받는다. 3 개의 인자가 아니라면 에러 문구를 출력하고 종료한다. Minimum support와 input file name, output file name을 차례로 입력 받는다. Minimum support는 단위가 %이므로 100으로 나누어 준다. printArgv함수는 프로그램의 시작을 알리는 함수로 입력 받은 인자들을 출력해준다.  
텍스트이(가) 표시된 사진

자동 생성된 설명  
이제 입력 받은 input file name으로 파일을 열어서 eof까지 읽어주면서 정보를 저장해준다. 혹시 transaction에 중복된 item이 있을 수도 있으니 자료구조로 set을 선택했다. 이 set을 vector에 저장해준 것이 DB이고 말 그대로 input.txt의 모든 정보를 담고 있다. 이제 입력 파일의 모든 정보를 저장해주었으니 수도 코드를 따라 코딩해주면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 수도 코드는 강의 자료에도 나와 있는 코드로 length 별로 frequent itemset을 구하는 것이 목표이다. 우선 우리는 L1에 해당하는 1개짜리 itemset의 set을 구해야 한다. 이를 수행하는 것이 initialScan이라는 함수이다.

텍스트이(가) 표시된 사진

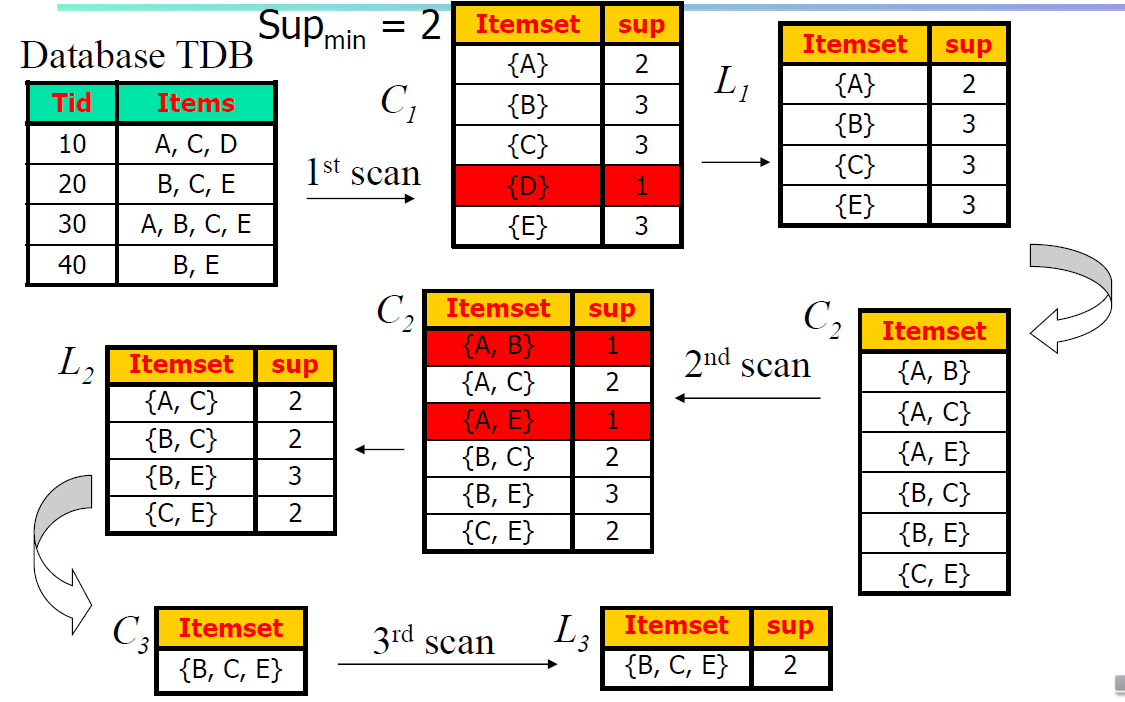
자동 생성된 설명  
텍스트이(가) 표시된 사진

자동 생성된 설명

L은 itemset과 sup을 map으로 저장한 것을 원소로 갖는 벡터이다. 즉 L[1]은 length가 1이면서 frequent한 itemset을 가리키고 L[k]는 length가 k이면서 frequent한 itemset을 가리킨다. DB를 순회하면서 1개짜리 아이템셋을 찾고, 해당 아이템 셋의 support가 minimum support이상이라면 L[1]에 추가해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이제 L[k]가 더 이상 생성되지 않을 때까지 k를 늘려주면서 frequent한 itemset을 구해주어야 한다.   


위와 같은 순서대로, candidate를 생성하고 sup을 검토해서 기준치 미달이면 drop하고 나머지는 L[k]에 추가해주면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 함수는 앞서 이야기한 candidate를 생성하는 함수이다. Candidates는 set안에 set을 갖는 자료구조이다. L[k]를 돌면서 self joining을 해주고, k+1의 길이가 완성되었다면 frequent한 지 검사한다. Frequent한 지 검사하는 함수는 chkItemset으로 검사하는 itemset의 부분집합(길이 k의 itemset만 해당)이 L[k]에 존재한다면 true를 반환하고 존재하지 않는다면 false를 반환한다. 이 과정이 pruning에 해당된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 생성된 candidates의 sup을 계산해주어서 minimum support를 충족하는 지 검사해야 한다. 다시 DB를 순회하면서 생성한 candidate가 속해 있는 지 확인해야 한다. 이를 확인하는 함수가 isSubset이라는 함수이다. DB의 itemset이 parent가 되고, candidates들 중 각각이 pattern이 되어서 안에 속하는 지 확인한다. 일치한다면 true를 반환하고, pattern이 parent보다 길다면 false, 짧다면 일일이 검사해서 확인해준다. 이를 통해서 true인 것들 만을 저장해준다. 이제 적합한 candidate와 그에 따른 sup을 저장해주었으니, minimum support를 충족하는 지 확인해주면 된다. 충족하지 않는다면 drop해주고 남은 것들을 L에 추가해준다. 각 for문마다 length를 늘려주어서 더 이상 생성되지 않았다면 반복문을 종료해주면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Apriori algorithm을 통해 구한 모든 frequent한 itemset을 바탕으로 association rule을 구해야 한다. 나중에 confidence를 구하기 위해 L을 하나의 map에 모두 저장해준다. 주어진 요구사항에 맞게 소수점 2자리까지 반올림해서 표시하게 한다. 또한 output을 length 오름차순으로 출력해주기 위해서 L의 itemset마다 계산해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

모든 Association rule을 구해주기 위해서는 하나의 아이템 셋에 대해서 모든 부분집합이 필요하다. 단, 자기 자신과 공집합은 제외한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

추후에 write\_Output에서 사용되기 좋게 따로 자료형을 선언하고 이를 원소로 하는 vector를 만들어서 push\_back 해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

getPowerset은 앞서 말한 집합들을 구하는 함수이다. 새로 만든 부분 집합과 새로운 원소를 사용해서 또 새로운 부분집합을 생성하고, 1 개짜리 원소는 따로 부분집합에 추가해준다. 이렇게 해서 구한 association rule들을 write\_Output 함수에 넘겨준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

요구사항의 출력 제한에 맞추어서 갖고 있는 정보들을 넣어준다. Sup은 어차피 같이 저장해주었기 때문에 전체 DB의 size로 나눠주면 되고, confidence의 경우 조건부 확률이기 때문에, 아까 전체 L을 저장해준 all\_frequent\_set에서 찾아서 나눠준다.

**4. Testing**

강의 자료의 예제로 테스트 해본 결과 값이 알맞게 잘 나왔다.



테이블이(가) 표시된 사진

자동 생성된 설명