

# 데이터구조설계

## 2차 프로젝트 보고서

제출 기한: 2022년 11월 20일(일)

과목: 데이터구조설계

담당 교수: 이기훈 교수님

학부: 컴퓨터정보공학부

학번: 2021202039

이름: 이소영

실습 분반: 금

## 1. Introduction

본 프로젝트는 FP-Growth와 B+ Tree를 구현하여 자료 구조에 대한 이해를 높이는 것을 목적으로 한다. 두 tree 구조를 이용하여 상품 추천 프로그램을 구현한다.

장바구니 데이터가 담긴 market.txt에서 함께 구매한 상품들을 줄 단위로 입력 받아 FP-Growth Algorithm을 구축한다. 이 알고리즘은 상품들의 연관성이 저장된 FP-Tree와 상품별 빈도수 및 정보, 해당 상품과 연결된 FP-Tree의 상품 노드들을 관리하는 Header Table로 구성된다.

위 FP-Growth에서 연관된 상품들 목록을 찾아 Frequent Pattern으로 result.txt에 저장한다. 이 result.txt에 적힌 정보들은 B+ Tree에 저장된다. B+-Tree는 IndexNode와 DataNode로 구성된다. IndexNode는 DataNode를 찾기 위한 Node이고 DataNode는 해당 빈도수를 가지는 Frequent Pattern들이 저장된 Node이다.

프로그램은 command.txt에서 받은 명령어에 따라 동작한다. 명령어의 종류와 대응하는 동작은 아래와 같다.

LOAD 명령은 market.txt 텍스트 파일의 데이터 정보를 불러오는 명령어로, 해당 텍스트 파일에 데이터가 존재하는 경우 텍스트 파일을 읽어 FP-Growth를 생성한다.

BTLOAD 명령은 market.txt 텍스트 파일의 데이터 정보를 불러오는 명령어로, 해당 텍스트 파일에 데이터 정보가 존재하는 경우 텍스트 파일을 읽어 B+-Tree에 저장한다.

PRINT\_ITEMLIST 명령은 FP-Growth의 Header Table에 저장된 상품들을 빈도수를 기준으로 내림차순으로 출력하는 명령어로 정해진 threshold보다 작은 빈도수를 가진 상품도 출력한다.

PRINT\_FPTREE 명령은 FP-Growth의 FP-Tree 정보를 출력하는 명령어이다. Header Table을 먼저 빈도수를 기준으로 오름차순으로 정렬하고 threshold 이상의 상품들을 출력한다. 출력하는 조건은 Header Table의 오름차순 순으로 FP-Tree의 path를 출력하며 threshold보다 작은 상품은 넘어간다. 해당 상품과 연결된 FP-Tree의 path들을 (상품명, 빈도수)로 root 노드 전까지 연결된 부모 노드를 출력한다. 해당 상품과 연결된 다음 노드들이 없을 때까지 출력하며 다음 노드로 이동하면 다음 줄로 이동한다.

PRINT\_BPTREE 명령은 B+ Tree에 저장된 Frequent Pattern 중 입력된 상품과 최소 빈도수 이상의 값을 가지는 Frequent Pattern을 출력하는 명령어이다. 명령어 입력 시 상품명과 최소 빈도수를 인자로 받는다. 명령이 실행되면 입력 받은 최소 빈도수를 기준으로 B+-Tree에서 탐색한다. 이후 B+ Tree에 저장된 Frequent Pattern 중 최소 빈도수를 만족하며, 입력된 상품을 포함하는 Frequent Pattern을 출력하며 이동한다.

PRINT\_CONFIDENCE 명령은 B+ Tree에 저장된 Frequent Pattern 중 입력된 상품과 연관율 이상의 confidence 값을 가지는 Frequent Pattern을 출력하는 명령어이다. 명령어 입력 시 상품명과 최소 연관율을 인자로 받는다. 명령이 실행되면 입력 받은 연관율과 해당 상품의 총 빈도수의 곱

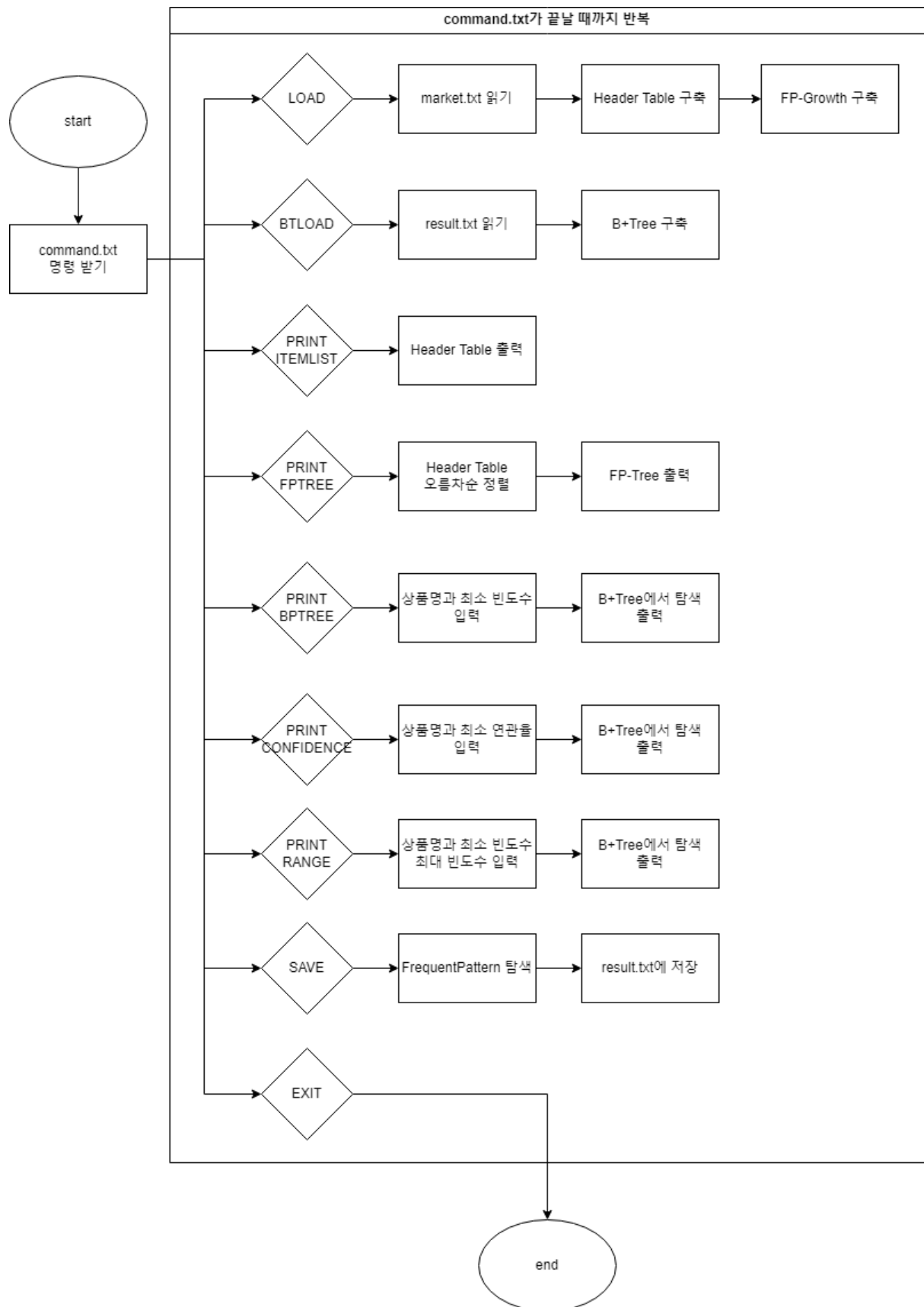
보다 큰 빈도수를 B+-Tree에서 탐색한다. 탐색이 끝나면 B+-Tree에 저장된 연관율 이상의 Frequent Pattern을 출력하며 이동한다.

PRINT\_RANGE 명령은 B+ Tree에 저장된 Frequent Pattern을 출력하는 명령어로, 상품명과 최소 빈도수, 최대 빈도수를 입력 받는다. 명령이 실행되면 최소 빈도수를 가지고 B+ Tree에서 Frequent Pattern을 탐색한다. 탐색이 끝나면 최대 빈도수까지 B+ Tree에 저장된 입력된 상품을 포함한 Frequent Pattern을 출력하며 이동한다.

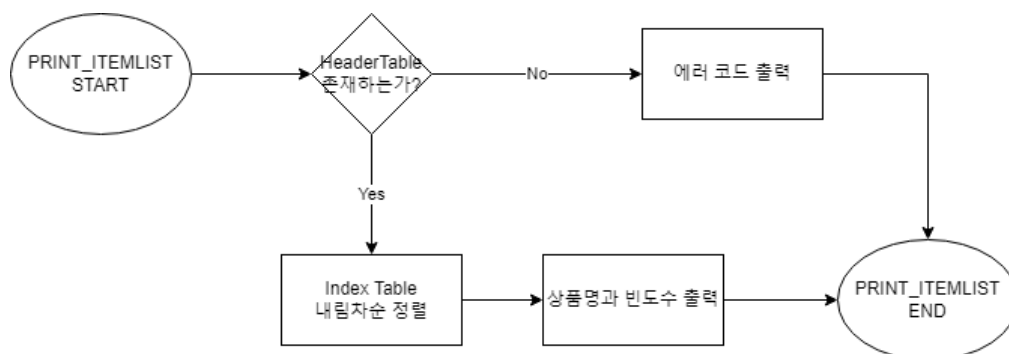
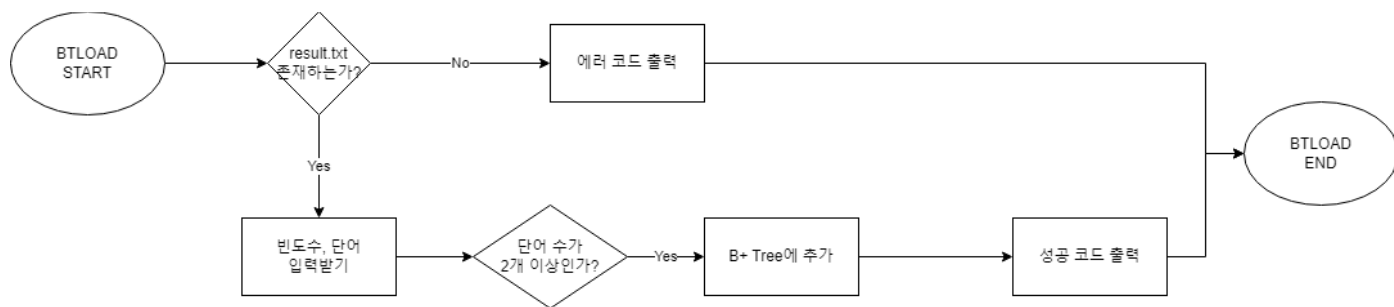
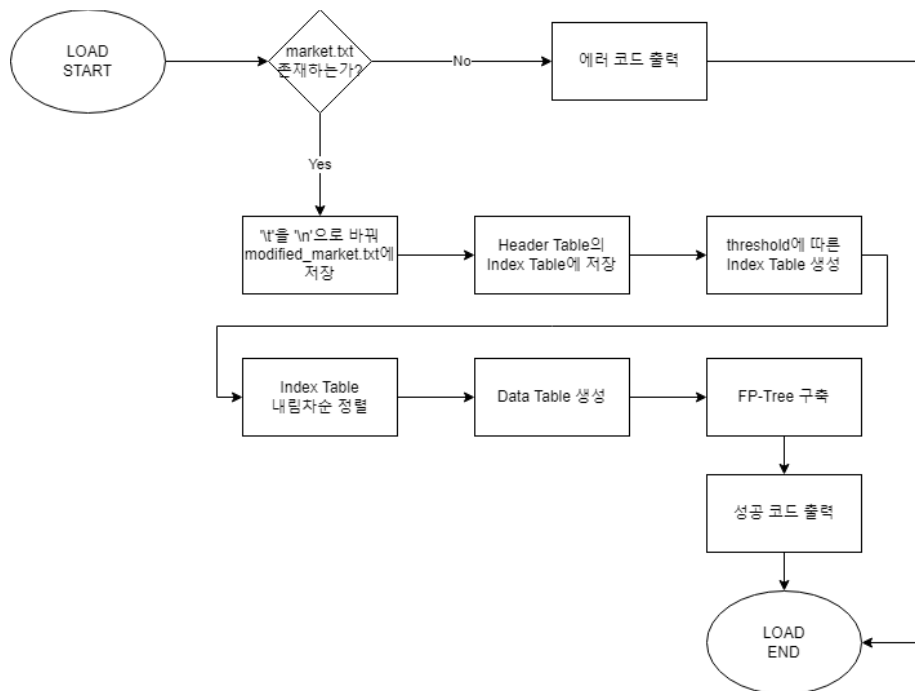
SAVE 명령어는 FP-Growth의 상품들의 연관성 결과를 저장하는 명령어로 생성된 Frequent Pattern들을 result.txt에 저장한다.

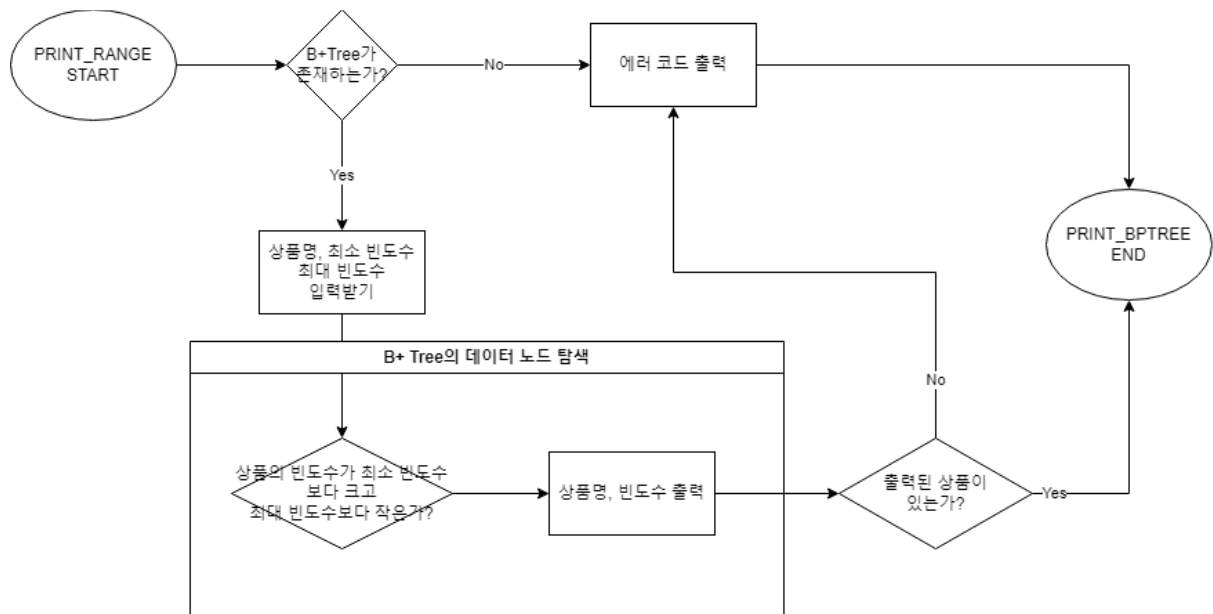
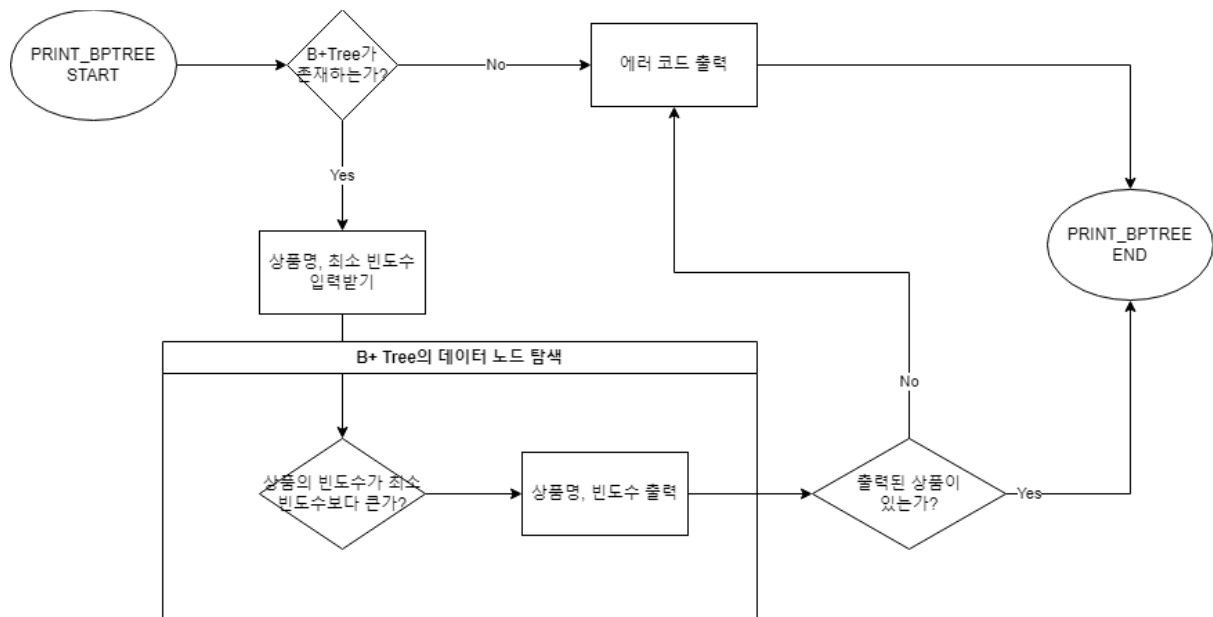
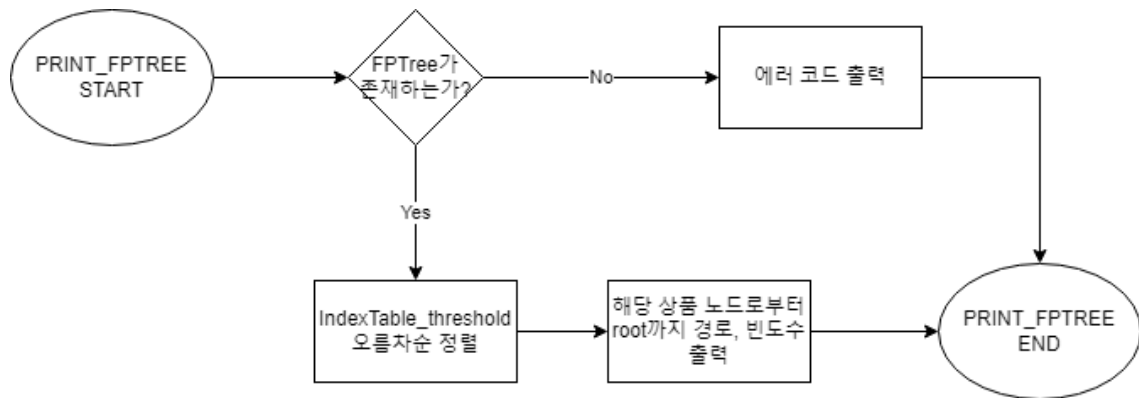
EXIT 명령어는 프로그램을 종료하는 명령어이다.

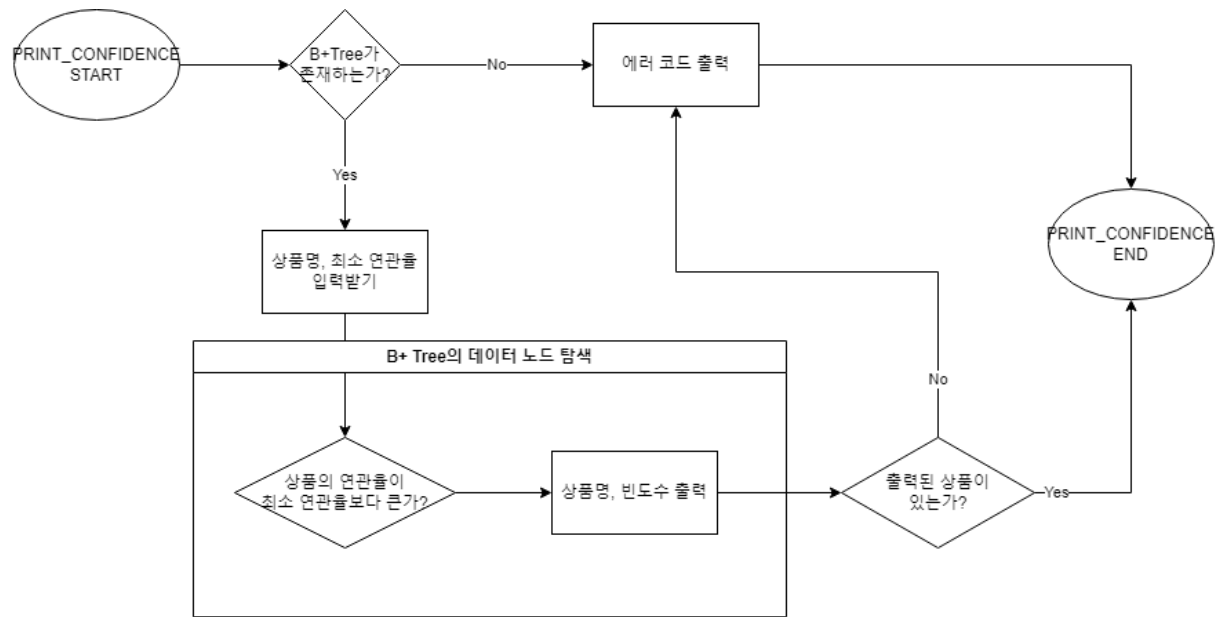
## 2. Flowchart



EXIT를 제외한 각 명령어 동작은 아래 flow chart에 따라 구현되었다.







### 3. Algorithm

각 명령어에 따라 FP-Growth 알고리즘과 B+ Tree가 구축되고, 활용되는 과정을 설명하고자 한다.

#### A. LOAD

LOAD에서는 Header Table을 생성하고 이에 따라 FP-Tree까지 생성하며 FP-Growth 알고리즘을 구축한다. market.txt에는 상품명이 탭('wt')을 기준으로 나누어져 있다. 단어를 더 편리하게 전달받기 위해 구분자 'wt'을 'wn'로 변경하여 modified\_market.txt에 저장한다. 이 동작은 Manager.h에 선언된 `void modify_market()` 함수에 의해 구현된다. 이제 어떤 상품명이 있는지, 각 상품의 빈도수는 몇인지 `vector<pair<int, string>> v`에 빈도수와 상품명 순으로 저장한다. 해당 벡터에 존재하지 않는 상품이 입력되면 벡터에 추가한다. 같은 상품명이 이미 벡터에 존재하면 빈도수에 1을 더한다. 이때 Manager.h에 선언, 정의된

```
bool checkExistinVector(vector<pair<int, string>> v, string ch)
```

```
int findElementinVector(vector<pair<int, string>> v, string ch)
```

두 함수로 각각 벡터에 해당 상품명이 존재하는지 여부와 몇 번째 원소인지 찾는다. 이렇게 상품명과 빈도수가 저장된 벡터를 FPGrowth.h, HeaderTable.h에 각각 선언된

```
void createTable(char* item, int frequency)
```

```
void HeaderTable::insertTable(char* item, int frequency)
```

위 함수들을 이용하여 Index Table을 만든다. 이제 modified\_market.txt에서 더 정보를 읽을 필요가 없으므로 파일을 닫는다.

FP-Tree를 만들기 위해 `list<string> l`을 선언한다. market.txt 텍스트 파일에서 한 줄 단위로 읽고, 그 안에서 'wt'을 기준으로 나누어 상품명을 위 리스트에 저장한다. 그리고 내림차순으로 정렬된 IndexTable의 빈도수에 따라 리스트를 정렬한다. FP-Tree에 연결되는 상품 순서는 빈도수를 기준으로 내림차순으로 결정되기 때문이다. 아래 함수로 정렬한다.

```
list<string> descendingList_IndexTable(list<string> l)
```

이제 리스트를 FP-Tree에 추가한다. 리스트의 각 원소의 정보를 담은 새로운 FPNode를 생성하여 만약 트리가 비어있다면 새로운 노드를 root로 지정하고, root에서부터 노드를 추가해간다. 새로운 노드와 부모 노드를 연결하고, 만약 이미 존재하는 상품명을 입력받았을 때는 해당 노드를 찾아 빈도수를 업데이트한다. 아래 함수로 FP-Tree에 리스트를 추가한다. market.txt에 있는 모든 줄을 리스트 형태로 FP-Tree에 추가하면 LOAD가 끝난다.

```
void createFPtree(FPNode* root, HeaderTable* table, list<string> item_array, int frequency)
```



## B. BTLOAD

result.txt에서 Frequent Pattern의 빈도수와 상품 집합을 전달받는다. 해당 텍스트 파일에서 한 줄 단위로 `set <string> s`으로 저장한다. Frequent Pattern 중 공집합 이거나 원소가 하나인 집합은 저장하지 않기 위하여 텍스트 파일에서 단어를 읽은 횟수를 `count` 변수에 저장하는데, 이 `count`에는 빈도수와 상품명 개수가 포함되기 때문에 `count`가 3 이상일 때만 B+-Tree에 추가한다.

빈도수를 `key`로, Frequent Pattern을 `set`으로 함수에 전달한다. 인자로 받은 정보에 따라 `FrequentPatternNode`를 생성한다. 만약 트리가 비어있다면 `root`에 이 `FrequentPatternNode`를 가진 `BpTreeNode`를 저장한다. 트리가 비어있지 않다면 새로운 정보는 트리의 데이터 노드에 삽입해야한다. 각 `BpTreeNode`에 있는 `map`은 빈도수의 크기에 따라 자식 노드를 가리키고 있다. `root`에서부터 출발하여 현재 노드의 `IndexMap`에 저장된 빈도수보다 `key`가 작다면 현재 노드의 가장 왼쪽 자식 노드로 이동하고, 크거나 같다면, `IndexMap`에 저장된 노드가 가리키는 곳으로 이동한다. 이렇게 이동하며 만약 현재 노드의 `pMostLeftChild=NULL`이라면 트리의 가장 하단부인 데이터 노드이며, 추가할 위치를 찾았다고 볼 수 있다.

만약 `key`가 이미 데이터 노드에 존재한다면 해당 데이터 노드의 `FrequentPatternNode`에 `set`을 추가해야 하므로 업데이트한다. 아니라면 데이터 노드의 `mapData`에 정보를 추가한다.

이렇게 노드를 생성하거나 추가한 후 만약 자식의 수가 정해진 양(`order`)보다 많다면 데이터 노드를 쪼개어야 한다. 필요한 경우 인덱스 노드를 쪼개야할 수 있다. 지금까지의 과정은 아래 함수에서 이루어진다.

```
bool BpTree::Insert(int key, set<string> input_set)
```

데이터 노드가 나누어지는 경우와 인덱스 노드가 나누어지는 경우를 분리해서 생각해야 한다. 데이터 노드가 나누어지는 경우, 나누는 기준은 현재 데이터 노드 내 `mapData`에서 중간에 해당하는 원소다. `mapData`의 `begin()`에서 `[order/2]`만큼 이동하여 기준이 될 빈도수를 구한다. 이 빈도수를 기준으로 작은 값들은 현재 데이터 노드에서 분리될 것이고, 큰 값들은 데이터 노드에 남아 있을 것이다. 분리된 작은 값들은 새로운 데이터 노드를 생성하여 기존의 데이터 노드와 `next`와 `prev`으로 연결된다. 그리고 기준이 된 빈도수는 새로운 인덱스 노드가 생성되어 이 안에 저장된다. 이 인덱스 노드는 두 데이터 노드의 부모가 된다. 그런데 만약 데이터 노드에게 부모 노드가 있었다면 새로운 인덱스 노드를 만들지 않고 위 인덱스 노드에 정보를 추가한다.

이렇게 데이터 노드들에서 정보가 올라오면 인덱스 노드에 저장된 정보 수가 `order`를 넘기는 경우가 생길 것이다. 이때 인덱스 노드를 쪼개야 한다. 데이터 노드를 쪼갤 때처럼 인덱스 노드의 `mapIndex`에서 가운데에 해당하는 빈도수를 기준으로 노드를 쪼개고, 새로운 인덱스 노드를 생성하고, `pParent`와 `pMostLeftChild`로 노드들을 연결한다. 만약 기존 인덱스 노드에게 부모가 없었다면, 새롭게 생성된 인덱스 노드가 부모이자 `root`가 된다.

데이터 노드와 인덱스 노드를 쪼개는(`split`) 동작은 아래 함수들로 구현된다.

```
void BpTree::splitDataNode(BpTreeNode* pDataNode)
```

```
void BpTree::splitIndexNode(BpTreeNode* pIndexNode)
```

result.txt에 있는 필요한 정보들을 모두 저장하면 BTLOAD가 완료된다.

### C. PRINT\_ITEMLIST

FP-Growth의 Index Table을 내림차순으로 정렬하고 출력한다. Header Table 클래스에서 출력하는 것이지만 이 클래스에는 log.txt인 fout 변수가 없어, print\_table.txt에 출력하고 FPGrowth.h에서 아래 함수를 호출하여 구현한다. 이 함수는 print\_table.txt에 적힌 내용을 log.txt에 복사하여 출력한다.

```
void Print_ItemList()
```

### D. PRINT\_FPTREE

FP-Tree 정보를 출력해야 한다. Header Table의 오름차순 순으로 FP-Tree의 path를 출력하고, threshold보다 작은 상품은 넘어간다는 조건에 따라 threshold보다 큰 상품만 저장한 IndexTable\_threshold를 만든다. 이는 LOAD 동작 때 void createIndexTable\_threshold()로 만들어져 있다. 이 Index Table을 오름차순으로 정렬하여 빈도수와 상품명에 따라 FP-Tree에서 노드를 찾아가 그 노드에서부터 root까지의 경로를 출력한다. 각 노드를 출력하고 parent로 이동하는 것을 반복한다. 위 과정은 아래 함수들로 구현한다.

```
void printFPTree()
```

```
void printPathToRoot(FILE*file, FPNODE* node)
```

### E. PRINT\_BPTREE & PRINT\_RANGE

두 명령은 동작이 유사하다. PRINT\_BPTREE는 명령 시 상품과 최소 빈도수를 입력 받는다. B+-Tree의 데이터 노드들을 순회하며 해당 상품이 존재하고 빈도수가 최소 빈도수 이상이면 출력한다. PRINT\_RANGE는 상품명, 최소 빈도수에 더해 최대 빈도수를 입력 받는다. B+-Tree의 데이터 노드들을 순회하며 해당 상품이 존재하고 빈도수가 최소 빈도수 이상이고, 최대 빈도수 이하인 경우에만 출력한다. 두 동작은 아래 함수들로 각각 실행된다.

```
bool BpTree::printFrequency(string item, int min_frequency)
```

```
bool BpTree::printRange(string item, int min, int max)
```

#### D. PRINT\_CONFIDENCE

명령 시 상품명과 최소 연관율을 입력 받는다. 연관율은  $\frac{\text{부분 집합의 빈도수}}{\text{해당 상품의 총 빈도수}}$ 로 구할 수 있다. 해당 상품의 총 빈도수를 구하기 위해 Header Table의 Index Table에서 해당 상품의 총 빈도수를 가져온다. 데이터 노드들을 순회하며 해당 상품이 있는 mapData를 찾는다. 만약 방문한 빈도수가 인자로 전달 받은 최소 빈도수와 총 빈도수의 곱을 총 빈도수와 최소 연관율의 곱으로 나눈 값 이상이면 출력한다. 연관율은 현재 빈도수를 총 빈도수로 나눈 값으로 출력한다.

```
bool BpTree::printConfidence(string item, double item_frequency, double min_confidence)
```

#### E. EXIT

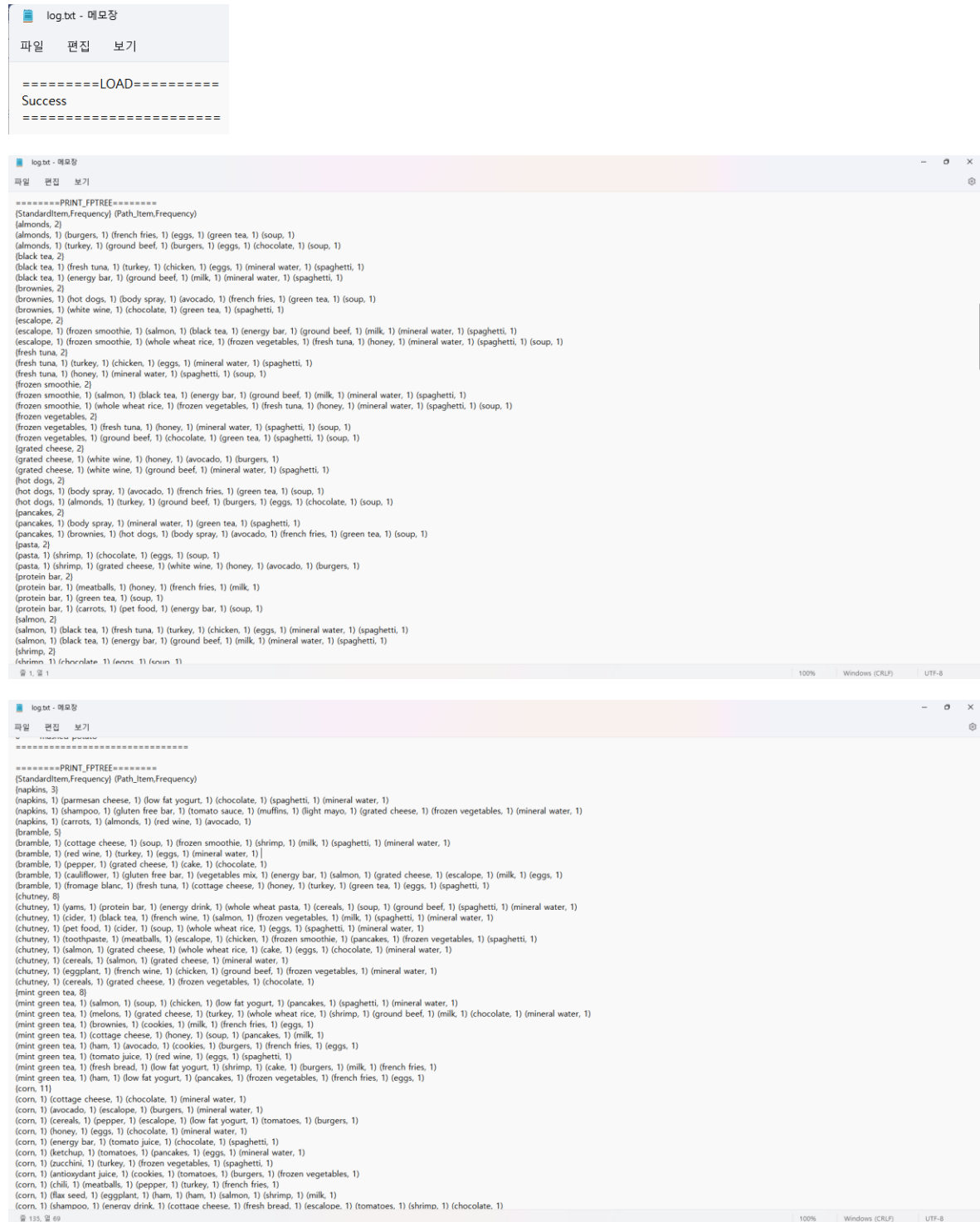
SAVE는 구현하지 않았으며 EXIT를 명령 받으면 프로그램을 종료한다.

## 4. Result Screen

아래 결과 화면들은 윈도우, visual studio 에서 실행한 화면이다.

### A. LOAD & PRINT\_FPTREE

LOAD는 PRINT\_FPTREE 출력이 제대로 되면 성공했음을 확실히 확인할 수 있다. 아래는 제공받은 testcase1, testcase2에 대한 LOAD 결과와 PRINT\_FPTREE 결과 일부이다.



```
=====LOAD=====
Success
=====

=====PRINT_FPTREE=====
[StandardItem.Frequency] (Path_Item.Frequency)
(almonds, 2)
(almonds, 1) (burgers, 1) (french fries, 1) (eggs, 1) (green tea, 1) (soup, 1)
(almonds, 1) (turkey, 1) (ground beef, 1) (burgers, 1) (eggs, 1) (chocolate, 1) (soup, 1)
(black tea, 2)
(black tea, 1) (fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(brownies, 2)
(brownies, 1) (hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(brownies, 1) (white wine, 1) (chocolate, 1) (green tea, 1) (spaghetti, 1)
(escalope, 2)
(escalope, 1) (frozen smoothie, 1) (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(escalope, 1) (frozen smoothie, 1) (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(fresh tuna, 2)
(fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen smoothie, 2)
(frozen smoothie, 1) (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(frozen smoothie, 1) (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen vegetables, 2)
(frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen vegetables, 1) (ground beef, 1) (chocolate, 1) (green tea, 1) (spaghetti, 1) (soup, 1)
(grated cheese, 2)
(grated cheese, 1) (white wine, 1) (honey, 1) (avocado, 1) (burgers, 1)
(grated cheese, 1) (white wine, 1) (ground beef, 1) (mineral water, 1) (spaghetti, 1)
(hot dogs, 2)
(hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(hot dogs, 1) (almonds, 1) (turkey, 1) (ground beef, 1) (burgers, 1) (eggs, 1) (chocolate, 1) (soup, 1)
(pancakes, 2)
(pancakes, 1) (body spray, 1) (mineral water, 1) (green tea, 1) (spaghetti, 1)
(pancakes, 1) (brownies, 1) (hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(pasta, 2)
(pasta, 1) (shrimp, 1) (chocolate, 1) (eggs, 1) (soup, 1)
(pasta, 1) (shrimp, 1) (grated cheese, 1) (white wine, 1) (honey, 1) (avocado, 1) (burgers, 1)
(protein bar, 2)
(protein bar, 1) (meatballs, 1) (honey, 1) (french fries, 1) (milk, 1)
(protein bar, 1) (green tea, 1) (soup, 1)
(protein bar, 1) (carrots, 1) (pet food, 1) (energy bar, 1) (soup, 1)
(salmon, 2)
(salmon, 1) (black tea, 1) (fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(shrimp, 2)
(shrimp, 1) (chocolate, 1) (eggs, 1) (soup, 1)

=====PRINT_FPTREE=====
[StandardItem.Frequency] (Path_Item.Frequency)
(napkins, 3)
(napkins, 1) (parmesan cheese, 1) (low fat yogurt, 1) (chocolate, 1) (spaghetti, 1) (mineral water, 1)
(napkins, 1) (shampoo, 1) (gluten free bar, 1) (tomato sauce, 1) (muffins, 1) (light mayo, 1) (grated cheese, 1) (frozen vegetables, 1) (mineral water, 1)
(napkins, 1) (carrots, 1) (almonds, 1) (red wine, 1) (avocado, 1)
(ramble, 5)
(ramble, 1) (cottage cheese, 1) (soup, 1) (frozen smoothie, 1) (shrimp, 1) (milk, 1) (spaghetti, 1) (mineral water, 1)
(ramble, 1) (red wine, 1) (turkey, 1) (eggs, 1) (mineral water, 1)
(ramble, 1) (pepper, 1) (grated cheese, 1) (cake, 1) (chocolate, 1)
(ramble, 1) (cauliflower, 1) (gluten free bar, 1) (vegetables mix, 1) (energy bar, 1) (salmon, 1) (grated cheese, 1) (escalope, 1) (milk, 1) (eggs, 1)
(ramble, 1) (fromage blanc, 1) (fresh tuna, 1) (cottage cheese, 1) (honey, 1) (turkey, 1) (green tea, 1) (eggs, 1) (spaghetti, 1)
(chutney, 8)
(chutney, 1) (yams, 1) (protein bar, 1) (energy drink, 1) (whole wheat pasta, 1) (cereals, 1) (soup, 1) (ground beef, 1) (spaghetti, 1) (mineral water, 1)
(chutney, 1) (cider, 1) (black tea, 1) (french wine, 1) (salmon, 1) (frozen vegetables, 1) (milk, 1) (spaghetti, 1) (mineral water, 1)
(chutney, 1) (pet food, 1) (cider, 1) (soup, 1) (whole wheat rice, 1) (eggs, 1) (spaghetti, 1) (mineral water, 1)
(chutney, 1) (toothpaste, 1) (meatballs, 1) (escalope, 1) (chicken, 1) (frozen smoothie, 1) (pancakes, 1) (frozen vegetables, 1) (spaghetti, 1)
(chutney, 1) (salmon, 1) (grated cheese, 1) (whole wheat rice, 1) (cake, 1) (eggs, 1) (chocolate, 1) (mineral water, 1)
(chutney, 1) (cereals, 1) (salmon, 1) (grated cheese, 1) (mineral water, 1)
(chutney, 1) (eggplant, 1) (french wine, 1) (chicken, 1) (ground beef, 1) (frozen vegetables, 1) (mineral water, 1)
(chutney, 1) (cereals, 1) (grated cheese, 1) (frozen vegetables, 1) (chocolate, 1)
(mint green tea, 8)
(mint green tea, 1) (salmon, 1) (soup, 1) (chicken, 1) (low fat yogurt, 1) (pancakes, 1) (spaghetti, 1) (mineral water, 1)
(mint green tea, 1) (melons, 1) (grated cheese, 1) (turkey, 1) (whole wheat rice, 1) (shrimp, 1) (ground beef, 1) (milk, 1) (chocolate, 1) (mineral water, 1)
(mint green tea, 1) (brownies, 1) (cookies, 1) (milk, 1) (french fries, 1) (eggs, 1)
(mint green tea, 1) (cottage cheese, 1) (honey, 1) (soup, 1) (pancakes, 1) (milk, 1)
(mint green tea, 1) (ham, 1) (avocado, 1) (cookies, 1) (burgers, 1) (french fries, 1) (eggs, 1)
(mint green tea, 1) (tomato juice, 1) (red wine, 1) (eggs, 1) (spaghetti, 1)
(mint green tea, 1) (fresh bread, 1) (low fat yogurt, 1) (shrimp, 1) (cake, 1) (burgers, 1) (milk, 1) (french fries, 1)
(mint green tea, 1) (ham, 1) (low fat yogurt, 1) (pancakes, 1) (frozen vegetables, 1) (french fries, 1) (eggs, 1)
(corn, 11)
(corn, 1) (cottage cheese, 1) (chocolate, 1) (mineral water, 1)
(corn, 1) (avocado, 1) (escalope, 1) (burgers, 1) (mineral water, 1)
(corn, 1) (cereals, 1) (pepper, 1) (escalope, 1) (low fat yogurt, 1) (tomatoes, 1) (burgers, 1)
(corn, 1) (honey, 1) (eggs, 1) (chocolate, 1) (mineral water, 1)
(corn, 1) (energy bar, 1) (tomato juice, 1) (chocolate, 1) (spaghetti, 1)
(corn, 1) (ketchup, 1) (tomatoes, 1) (pancakes, 1) (eggs, 1) (mineral water, 1)
(corn, 1) (zucchini, 1) (turkey, 1) (frozen vegetables, 1) (spaghetti, 1)
(corn, 1) (antioxidant juice, 1) (cookies, 1) (tomatoes, 1) (burgers, 1) (frozen vegetables, 1)
(corn, 1) (chili, 1) (meatballs, 1) (pepper, 1) (turkey, 1) (french fries, 1)
(corn, 1) (flax seed, 1) (eggplant, 1) (ham, 1) (ham, 1) (salmon, 1) (shrimp, 1) (milk, 1)
(corn, 1) (shampoo, 1) (energy drink, 1) (cottage cheese, 1) (fresh bread, 1) (escalope, 1) (tomatoes, 1) (shrimp, 1) (chocolate, 1)
```

## B. BTLOAD

```
log.txt - 메모장
파일 편집 보기

=====LOAD=====
Success
=====

=====BTLOAD=====
Success
=====
```

## C. PRINT\_ITEMLIST

```
log.txt - 메모장
파일 편집 보기

=====
=====PRINT_ITEMLIST=====
Item Frequency
12 soup
9 spaghetti
9 green tea
7 mineral water
5 milk
5 french fries
5 eggs
5 chocolate
4 ground beef
4 burgers
3 white wine
3 honey
3 energy bar
3 chicken
3 body spray
3 avocado
2 whole wheat rice
2 turkey
2 shrimp
2 salmon
2 protein bar
2 pasta
2 pancakes
2 hot dogs
2 grated cheese
2 frozen vegetables
2 frozen smoothie
2 fresh tuna
2 escalope
2 brownies
2 black tea
2 almonds
1 whole wheat pasta
1 toothpaste
1 tomatoes
1 soda
1 shampoo
1 shallot
1 red wine
1 protein bar
.
```

## D. PRINT\_FPTREE

```
log.txt - 메모장

파일 편집 보기

=====PRINT_FPTREE=====
(StandardItem.Frequency) (Path,Item.Frequency)
(almonds, 2)
(almonds, 1) (burgers, 1) (french fries, 1) (eggs, 1) (green tea, 1) (soup, 1)
(almonds, 1) (turkey, 1) (ground beef, 1) (burgers, 1) (eggs, 1) (chocolate, 1) (soup, 1)
(black tea, 2)
(black tea, 1) (fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(brownies, 2)
(brownies, 1) (hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(brownies, 1) (white wine, 1) (chocolate, 1) (green tea, 1) (spaghetti, 1)
(escalope, 2)
(escalope, 1) (frozen smoothie, 1) (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(escalope, 1) (frozen smoothie, 1) (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(fresh tuna, 2)
(fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen smoothie, 2)
(frozen smoothie, 1) (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(frozen smoothie, 1) (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen vegetables, 2)
(frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
(frozen vegetables, 1) (ground beef, 1) (chocolate, 1) (green tea, 1) (spaghetti, 1) (soup, 1)
(grated cheese, 2)
(grated cheese, 1) (white wine, 1) (honey, 1) (avocado, 1) (burgers, 1)
(grated cheese, 1) (white wine, 1) (ground beef, 1) (mineral water, 1) (spaghetti, 1)
(hot dogs, 2)
(hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(hot dogs, 1) (almonds, 1) (turkey, 1) (ground beef, 1) (burgers, 1) (eggs, 1) (chocolate, 1) (soup, 1)
(pancakes, 2)
(pancakes, 1) (body spray, 1) (mineral water, 1) (green tea, 1) (spaghetti, 1)
(pancakes, 1) (brownies, 1) (hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
(pasta, 2)
(pasta, 1) (shrimp, 1) (chocolate, 1) (eggs, 1) (soup, 1)
(pasta, 1) (shrimp, 1) (grated cheese, 1) (white wine, 1) (honey, 1) (avocado, 1) (burgers, 1)
(protein bar, 2)
(protein bar, 1) (meatballs, 1) (honey, 1) (french fries, 1) (milk, 1)
(protein bar, 1) (green tea, 1) (soup, 1)
(protein bar, 1) (carrots, 1) (pet food, 1) (energy bar, 1) (soup, 1)
(salmon, 2)
(salmon, 1) (black tea, 1) (fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
(salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
(shrimp, 2)
(shrimp, 1) (chocolate, 1) (eggs, 1) (soup, 1)

줄 17, 열 7 100% Windows (CRLF) UTF-8
```

## E. PRINT\_BPTREE

```
log.txt - 메모장

파일 편집 보기

=====

=====PRINT_BPTREE=====
FrequentPattern Frequency
{almonds, eggs} 2
{burgers, eggs} 2
{chicken, eggs} 2
{eggs, french fries} 2
{eggs, mineral water} 2
{eggs, turkey} 2
{almonds, burgers, eggs} 2
{almonds, eggs, soup} 2
{burgers, eggs, soup} 2
{chicken, eggs, mineral water} 2
{eggs, french fries, soup} 2
{almonds, burgers, eggs, soup} 2
{chocolate, eggs} 3
{chocolate, eggs, soup} 3
{eggs, soup} 4
=====
```

## F. PRINT\_CONFIDENCE

```
log.txt - 메모장
파일 편집 보기
{eggs, soup} 4
=====

=====PRINT_CONFIDENCE=====
FrequentPattern Frequency Confidence
{almonds, eggs} 2 0.4
{burgers, eggs} 2 0.4
{chicken, eggs} 2 0.4
{eggs, french fries} 2 0.4
{eggs, mineral water} 2 0.4
{eggs, turkey} 2 0.4
{almonds, burgers, eggs} 2 0.4
{almonds, eggs, soup} 2 0.4
{burgers, eggs, soup} 2 0.4
{chicken, eggs, mineral water} 2 0.4
{eggs, french fries, soup} 2 0.4
{almonds, burgers, eggs, soup} 2 0.4
{chocolate, eggs} 3 0.6
{chocolate, eggs, soup} 3 0.6
{eggs, soup} 4 0.8
=====

=====PRINT_CONFIDENCE=====
FrequentPattern Frequency Confidence
{chocolate, eggs} 3 0.6
{chocolate, eggs, soup} 3 0.6
{eggs, soup} 4 0.8
=====
```

## G. PRINT\_RANGE & EXIT (SAVE는 구현하지 않는다.)

```
=====PRINT_RANGE=====
FrequentPattern Frequency
{almonds, eggs} 2
{burgers, eggs} 2
{chicken, eggs} 2
{eggs, french fries} 2
{eggs, mineral water} 2
{eggs, turkey} 2
{almonds, burgers, eggs} 2
{almonds, eggs, soup} 2
{burgers, eggs, soup} 2
{chicken, eggs, mineral water} 2
{eggs, french fries, soup} 2
{almonds, burgers, eggs, soup} 2
{chocolate, eggs} 3
{chocolate, eggs, soup} 3
=====

=====EXIT=====
Success
=====
```

## 리눅스에서의 실행 결과

```
=====LOAD=====
```

```
Success
```

```
=====
```

```
=====BTLOAD=====
```

```
Success
```

```
=====
```

```
=====PRINT_ITEMLIST=====
```

Item	Frequency
12	soup
9	spaghetti
9	green tea
7	mineral water
5	milk
5	french fries
5	eggs
5	chocolate
4	ground beef
4	burgers
3	white wine
3	honey
3	energy bar
3	chicken
3	body spray
3	avocado
2	whole wheat rice
2	turkey
2	shrimp
2	salmon
2	protein bar
2	pasta
2	pancakes
2	hot dogs
2	grated cheese
2	frozen vegetables
2	frozen smoothie
2	fresh tuna
2	escalope
2	brownies
2	black tea
2	almonds
1	whole wheat pasta
1	toothpaste
1	tomatoes
1	soda

```
=====PRINT_FPTREE=====
```

```
{StandardItem,Frequency} (Path_Item,Frequency)
{almonds, 2}
{almonds, 1} (burgers, 1) (french fries, 1) (eggs, 1) (green tea, 1) (soup, 1)
{almonds, 1} (turkey, 1) (ground beef, 1) (burgers, 1) (eggs, 1) (chocolate, 1) (soup, 1)
{black tea, 2}
{black tea, 1} (fresh tuna, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
{black tea, 1} (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
{brownies, 2}
{brownies, 1} (hot dogs, 1) (body spray, 1) (avocado, 1) (french fries, 1) (green tea, 1) (soup, 1)
{brownies, 1} (white wine, 1) (chocolate, 1) (green tea, 1) (spaghetti, 1)
{escalope, 2}
{escalope, 1} (frozen smoothie, 1) (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
{escalope, 1} (frozen smoothie, 1) (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
{fresh tuna, 2}
{fresh tuna, 1} (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 1) (spaghetti, 1)
{fresh tuna, 1} (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
{frozen smoothie, 2}
{frozen smoothie, 1} (salmon, 1) (black tea, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 1) (spaghetti, 1)
{frozen smoothie, 1} (whole wheat rice, 1) (frozen vegetables, 1) (fresh tuna, 1) (honey, 1) (mineral water, 1) (spaghetti, 1) (soup, 1)
{frozen vegetables, 2}
```



=====PRINT\_BPTREE=====

FrequentPattern Frequency

```
{almonds, eggs} 2
{burgers, eggs} 2
{chicken, eggs} 2
{eggs, french fries} 2
{eggs, mineral water} 2
{eggs, turkey} 2
{almonds, burgers, eggs} 2
{almonds, eggs, soup} 2
{burgers, eggs, soup} 2
{chicken, eggs, mineral water} 2
{eggs, french fries, soup} 2
{almonds, burgers, eggs, soup} 2
{chocolate, eggs} 3
{chocolate, eggs, soup} 3
{eggs, soup} 4
```

=====

=====PRINT\_CONFIDENCE=====

FrequentPattern Frequency Confidence

```
{almonds, eggs} 2 0.4
{burgers, eggs} 2 0.4
{chicken, eggs} 2 0.4
{eggs, french fries} 2 0.4
{eggs, mineral water} 2 0.4
{eggs, turkey} 2 0.4
{almonds, burgers, eggs} 2 0.4
{almonds, eggs, soup} 2 0.4
{burgers, eggs, soup} 2 0.4
{chicken, eggs, mineral water} 2 0.4
{eggs, french fries, soup} 2 0.4
{almonds, burgers, eggs, soup} 2 0.4
{chocolate, eggs} 3 0.6
{chocolate, eggs, soup} 3 0.6
{eggs, soup} 4 0.8
```

=====

=====PRINT\_CONFIDENCE=====

FrequentPattern Frequency Confidence

```
{chocolate, eggs} 3 0.6
{chocolate, eggs, soup} 3 0.6
{eggs, soup} 4 0.8
```

=====

```
=====PRINT_RANGE=====
FrequentPattern Frequency
{almonds, eggs} 2
{burgers, eggs} 2
{chicken, eggs} 2
{eggs, french fries} 2
{eggs, mineral water} 2
{eggs, turkey} 2
{almonds, burgers, eggs} 2
{almonds, eggs, soup} 2
{burgers, eggs, soup} 2
{chicken, eggs, mineral water} 2
{eggs, french fries, soup} 2
{almonds, burgers, eggs, soup} 2
{chocolate, eggs} 3
{chocolate, eggs, soup} 3
=====

=====EXIT=====
Success
=====
```

## 5. Consideration

FPGrowth.h에서는 log.txt를 ofstream\* fout, result.txt를 ofstrea flog로 표현한다. 똑같이 파일에 출력하는 것인데 왜 fout은 포인터고 flog는 아닌지 이해를 하지 못하여 사용에 어려움을 겪었다. 그러나 log.txt는 Manager.h에서 지정하는 것이고, 이를 FPGrowth.h에서 접근하려면 log.txt의 위치를 알아야 하므로 포인터로 전달하는 것이 효율적이라는 것을 이해하여 성공적으로 출력할 수 있었다.

몇몇 함수에 대해 헤더 파일을 포함하는 cpp 파일에서 정의를 하였으나 외부에서 참조할 수 없다는 오류가 떠 컴파일이 불가능했다. 이런 경우 헤더 파일에서 정의와 선언을 모두 하면 오류가 뜨지 않았다. 원인을 확실히 알 수 없으나 헤더 파일이 제대로 포함되지 않았다고 추측한다.

노드 추가나 PRINT 시 데이터 노드에 접근해야 했는데 처음에는 가장 왼쪽 데이터 노드로 접근하여 각 노드의 mapData를 비교하며 next로 이동하여 적합한 위치를 찾으려고 하였다. 그러나 B+Tree의 특성 상 빈도수를 기준으로 큰 것은 오른쪽으로 작은 것은 왼쪽으로 가는 BinarySearchTree와 비슷하다고 느껴 root에서부터 mapData나 mapIndex를 비교하여 자식 노드로 내려가도록 수정하였다. 이 방법이 트리의 특성을 이해하고 더 잘 활용한 방법이라고 판단하였다.

본 프로젝트로 FP-Growth 알고리즘과 B+-Tree를 구현하며 효과적으로 패턴을 분석하는 법과 Multiple ways Tree 구조에 대해 이해하고 활용할 수 있었다.