2020-2

Data Structure Project #2

학번: 2019202045

이름: 박수연

담당교수: 이형근 교수님

제출일자: 2020-10-09

Data Structure Lab. Project #2 Report

A. Introduction

1. 제목

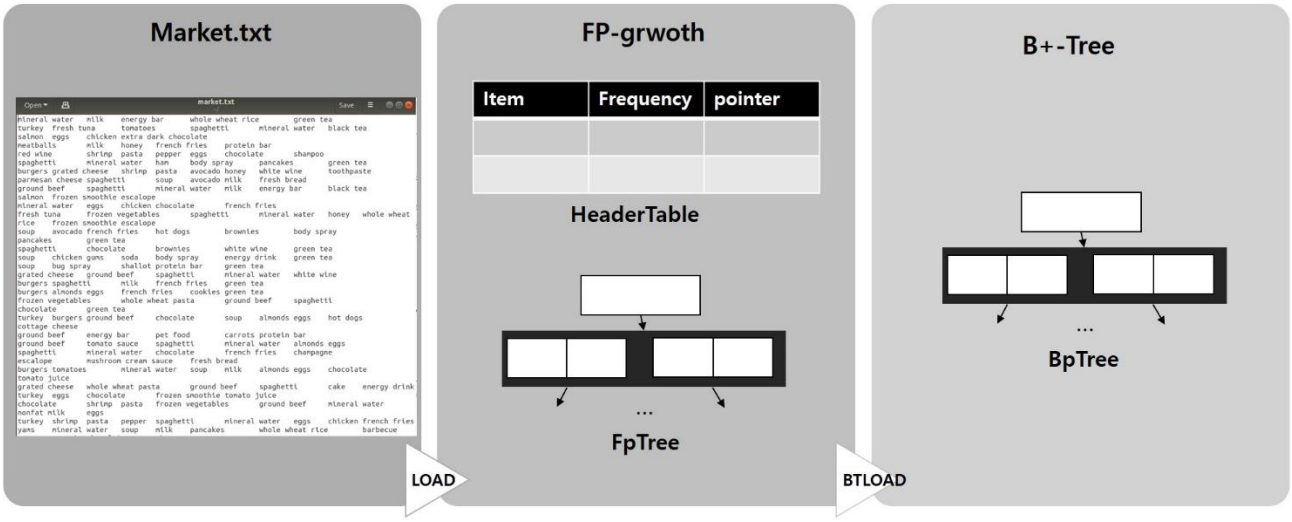
Fp Grwoth와 B+-Tree를 이용한 상품 추천 프로그램

2. 프로그램 설명

장바구니 데이터를 입력받아 같이 구매한 상품을 기준으로 Fp-Growth를 구축한다. 상품 구입의 연관성을 Tree 구조로 저장하고 있는 Fp-Tree, 상품별 빈도수 및 정보, 상품과 연결된 Fp-Tree의 상품 노드를 관리하는 Header Table 가 Fp-grwoth를 구성한다.

DataNode는 해당 빈도수를 가지는 Frequent Pattern이 저장된 노드, IndexNode는 DataNode를 찾기 위한 노드로, B+-Tree에서는 이 두 노드를 사용한다.

tructure implement

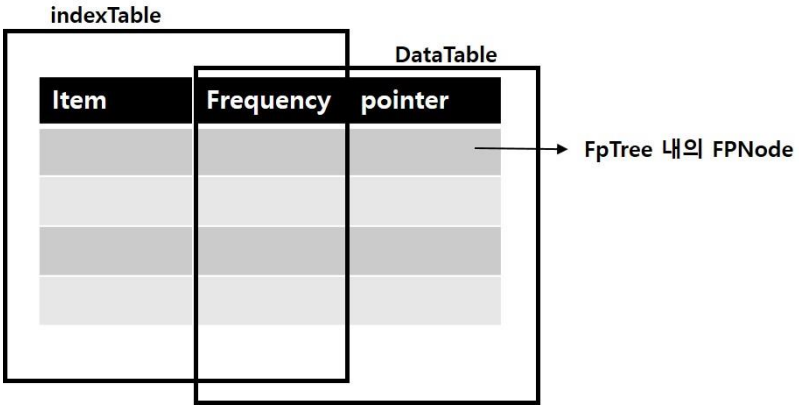


1) FP- Growth

market.txt에 저장된 데이터를 한 줄 단위로 불러와 상품 구입의 연관성을 저장한다.
FP-Tree는 FPNode 클래스를 통해, Header Table은 Header Table 클래스를 통해 구현한다.

2) Header Table

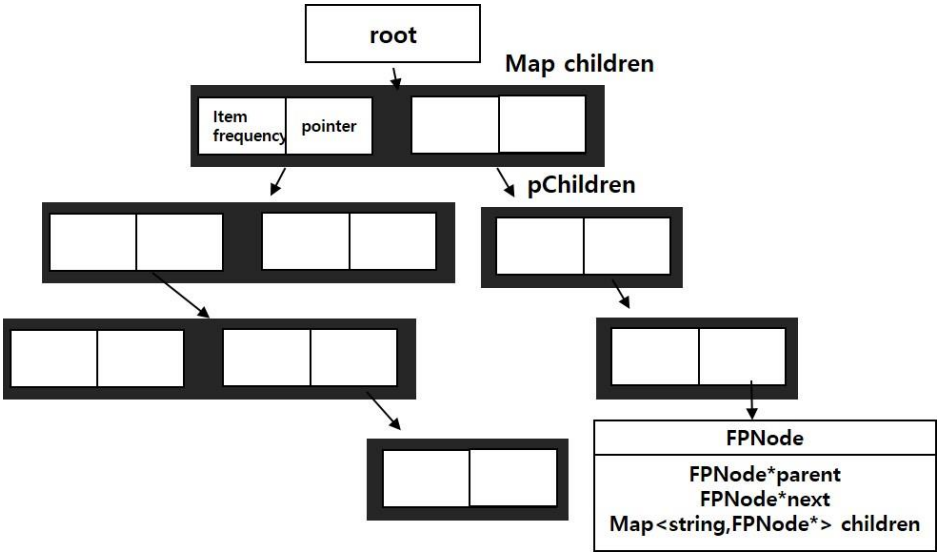
Index Table과 Data Table로 나누어 구성한다. Threshold보다 작은 상품도 포함하여 모든 상품의 정보를 저장하며, Index Table은 빈도수와 상품명이 list로 저장된 변수, DataTable은 상품명 string 과 상품과 연결되는 노드가 map으로 저장된 변수이다. 해당 노드에는 상품의 정보 및 FP-Tree로 연결되는 포인터를 갖고 있다.



3) Fp-Tree

자식 노드들은 map 형태로 저장하며, key는 string은 상품명을 저장하고 value의 FPNode*는 해당 상품의 빈도수 정보 및 연결된 Node 정보를 저장한다. FP-Tree의 노드는 Header Table에서 같은 상품의 노드에 연결되어야 한다.

FP-Tree에 연결되는 상품 순서는 빈도수 기존으로 내림차순으로 결정이 되며, 저장되는 데이터 상품명과 노드이며 상품 노드에는 빈도수, 부모 노드 포인터, 자식 노드 포인트가 저장된다.



4) B+-Tree

result.txt에 저장된 데이터를 불러와 구축하며 B+- Tree는 BpTreeIndexNode와 BpTreeDataNode로 구성되며 각 노드는 B+-Tree BpTreeNode를 상속받는다. 데이터 노드는 단말 노드로, 해당 빈도수의 Frequent Pattern이 저장된 FrequentPatternNode를 map 형태로 가지고 있으며, 가장 왼쪽에 있는 자식을 가리키는 포인터를 갖고 있다.

5) Function

LOAD : "marktet.txt" 텍스트 파일로부터 데이터를 불러와, HeaderTable, FPTree의 구축을 통해 FP-Growth를 생성한다. 텍스트 파일이 존재하지 않거나 이미 데이터가 LOAD된 경우 에러 코드를 출력한다.

BTLOAD : "result.txt" 텍스트 파일로부터 데이터를 불러와, B+Tree를 생성한다. 만약 텍스트 파일이 존재하지 않거나 이미 데이터가 BTLOAD 된 경우 에러 코드를 출력한다.

PRINT_ITEMLIST : FP-Growth의 Header Table에 저장된 상품의 Item 이름과 그 빈도수를 내림차순으로 출력한다. Header Table이 비어 있을 시 에러 코드를 출력한다.

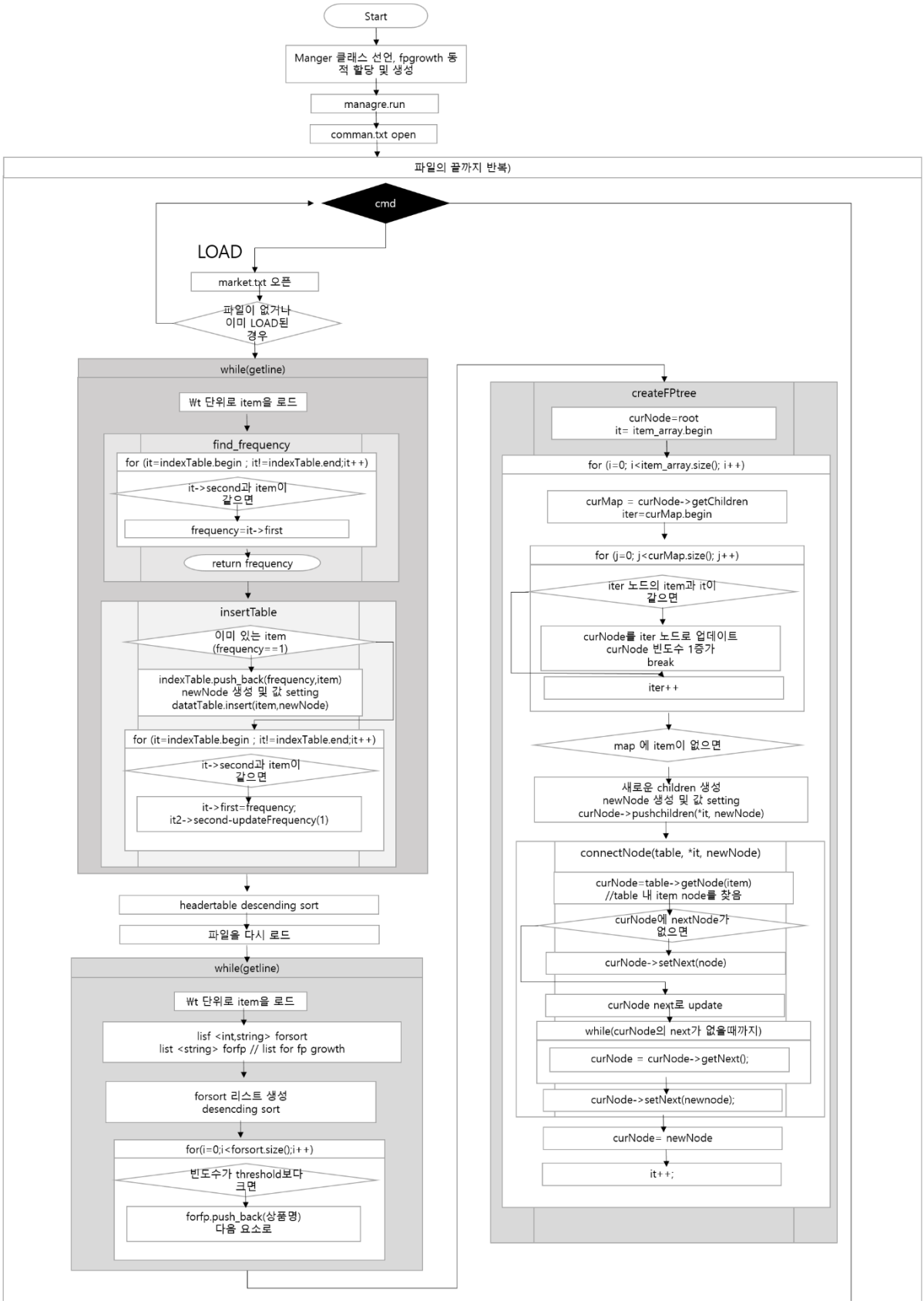
PRINT_FPTREE : FP-Growth의 FP-Tree 정보를 출력하는 명령어로, Header Table을 오름차순으로 정렬 후 FP-Tree에 존재하는 노드만을 출력한다. 이 때 Header Table의 상품을 { 상품명, 빈도수 }의 포맷으로 출력후, 해당 상품과 연결된 FP-Tree의 path를 (상품명, 빈도수) 의 포맷으로 root를 만날 때까지 연결된 부모 노드를 따라 출력한다. 해당 상품과 연결된 모든 path를 출력해야 한다.

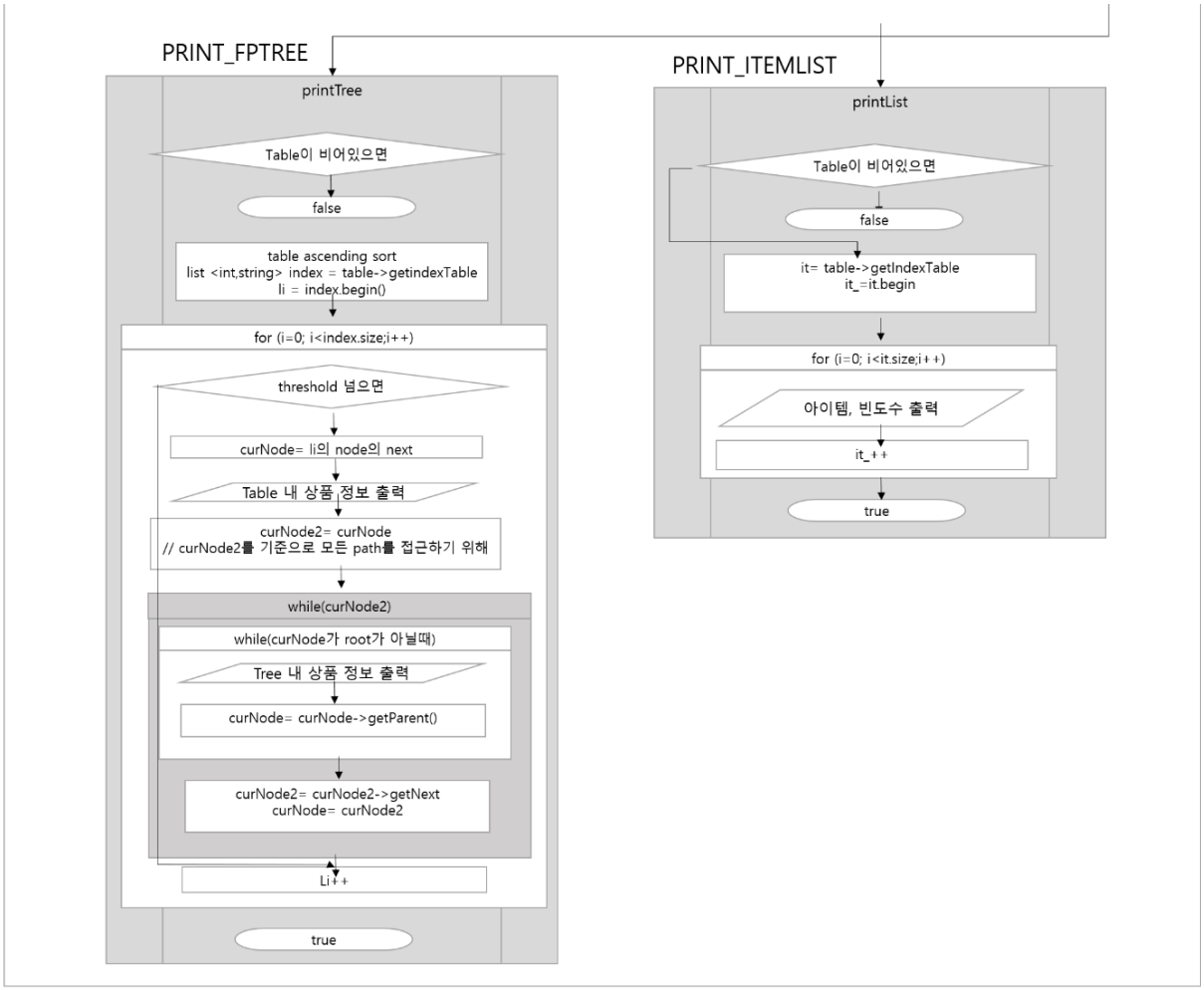
PRINT_MIN : B+-Tree에 저장된 Frequent Pattern 중 첫 번째 인자로 입력받는 상품과, 두 번째 인자로 입력받는 최소 빈도수 이상의 값을 가지는 Frequent Pattern을 출력한다. 최소 빈도수를 기준으로 B+-Tree에서 탐색 후, B+-Tree에 저장된 Frequent Pattern를 출력하며 이동한다.

PRINT_CONFIDIENC : B+-Tree에 저장된 Frequent Pattern 중 첫 번째 인자로 입력받는 상품과, 두 번째 인자로 입력받는 연관율 이상의 값을 갖는 Frequent Pattern을 출력한다. 연관율은 (부분 집합의 빈도 수)/(해당 상품의 총 빈도 수) 이다. 입력받는 연관율과 해당 상품의 총 빈도 수의 곱보다 큰 빈도수를 B+-Tree에서 탐색 후, 탐색이 끝나면 B+-Tree에 저장된 Frequent Pattern을 출력하며 이동한다.

PRINT_RANGE : B+-Tree에 저장된 Frequent Patten을 출력하는 명령어로, 첫 번째 인자로 상품명, 두 번째 인자로 최소 빈도수, 세 번째 인자로 최대 빈도수를 입력받는다. 최소 빈도수를 통해 B+-Tree에서 Frequent pattern을 탐핵 후 최대 빈도수까지 B+-Tree에 저장된 Frequent Pattern을 출력하며 이동한다.

B. Flow chart


Start
Manger 클래스 선언, fpgrowth 동적 할당 및 생성
managre.run
comman.txt open
파일의 끝까지 반복)
cmd
LOAD
market.txt 오픈
파일이 없거나 이미 LOAD된 경우
while(getline)
Wt 단위로 item을 로드
find_frequency
for (it=indexTable.begin ; it!=indexTable.end;it++)
it->second과 item이 같으면
frequency=it->first
return frequency
insertTable
이미 있는 item (frequency==1)
indexTable.push_back(frequency,item)
newNode 생성 및 값 setting
datatTable.insert(item,newNode)
for (it=indexTable.begin ; it!=indexTable.end;it++)
it->second과 item이 같으면
it->first=frequency;
it2->second-updateFrequency(1)
headertable descending sort
파일을 다시 로드
while(getline)
Wt 단위로 item을 로드
lisf <int,string> forsort
list <string> forfp // list for fp growth
forsort 리스트 생성
desencding sort
for(i=0;i<forsort.size();i++)
빈도수가 threshold보다 크면
forfp.push_back(상품명)
다음 요소로
createFPtree
curNode=root
it= item_array.begin
for (i=0; i<item_array.size(); i++)
curMap = curNode->getChildren
iter=curMap.begin
for (j=0; j<curMap.size(); j++)
iter 노드의 item과 it이 같으면
curNode를 iter 노드로 업데이트
curNode 빈도수 1증가
break
iter++
map 에 item이 없으면
새로운 children 생성
newNode 생성 및 값 setting
curNode->pushchildren(*it, newNode)
connectNode(table, *it, newNode)
curNode=table->getNode(item)
//table 내 item node를 찾음
curNode에 nextNode가 없으면
curNode->setNext(node)
curNode next로 update
while(curNode의 next가 없을때까지)
curNode = curNode->getNext();
curNode->setNext(newnode);
curNode= newNode
it++;


PRINT_FPTREE
printTree
Table이 비어있으면
false
table ascending sort
list <int,string> index = table->getindexTable
li = index.begin()
for (i=0; i<index.size;i++)
threshold 넘으면
curNode= li의 node의 next
Table 내 상품 정보 출력
curNode2= curNode
// curNode2를 기준으로 모든 path를 접근하기 위해
while(curNode2)
while(curNode가 root가 아닐때)
Tree 내 상품 정보 출력
curNode= curNode->getParent()
curNode2= curNode2->getNext
curNode= curNode2
Li++
true
PRINT_ITEMLIST
printList
Table이 비어있으면
false
it= table->getIndexTable
it_=it.begin
for (i=0; i<it.size;i++)
아이템, 빈도수 출력
it_++
true

C. 알고리즘

Manager

command.txt로부터 cmd를 불러온다. 파일 오픈 실패 시 에러 문구 출력한다.
fin에 cmd를 저장해, 값을 LOAD PRINT_ITMELIST PRINT_FPTREE 의 문자열과 비교해 strcmp==0이 될 때 해당 함수로 진입하며, 각 함수는 return값이 true일 때 성공 문구를 그렇지 않을 때 에러 문구를 출력한다. 이 과정을 파일이 끝날 때까지 반복 수행한다.
해당 과제에서는 HeaderTable, FP-Growth tree의 구현까지밖에 완성하지 못하였으므로, SAVE PRINT_MIN PRINT_RAGNE PRINT_CONFIDENCE 의 경우 기능을 수행하지 않고 다음 cmd로 넘어갈 수 있도록 했다.

LOAD

Headertable 생성

data.txt로부터 데이터를 불러오는데 파일 오픈 실패 시, 혹은 이미 Headertable이 완성되어 있을 시 에러코드를 출력한다. 파일을 한 줄씩 읽어들일 변수를 선언해 getline 함수를 통해 문서의 끝까지 정보를 읽기를 반복한다. strtok 함수를 이용해 탭단위로 저장되어 있는 정보를 읽어와 item에 상품명을 저장, item_frequency 함수를 호출해 얻어낸 빈도수를 변수 f에 저장한다. 이 두 변수를 이용해 createTabel(item, f+1)의 형식으로 함수를 호출해 Headertable을 생성한 후, descendingTable함수를 통해 이를 내림차순으로 정렬하고 파일을 닫는다.

- **HeaderTable::find_frequency**

  초기 frequency 값을 0으로 초기화한 후 indexTable의 첫 요소부터 마지막 요소까지를 for문으로 반복해 전달받은 item과 같은 이름의 요소가 있을 경우 그 빈도수를 frequency에 저장하여 리턴한다.

- **FPGrowth::createTable=> HeaderTable::insertTable**

  만약 frequency 의 값이 1일 경우 해당 item이 header table에 존재하지 않는 새로운 item이기 때문에 indexTable.push_back을 통해 indextable에 item을 추가한다. DataTable에 item을 추가하기 위해 새로운 newNode를 동적할당 및 그 값을 설정해주고, dataTable.insert을 통해 item을 추가한다.

  만약 item이 이미 headerTable에 존재하는 경우, indexTable에서 해당 item의 이름과 같은 명의 상품명을 가지는 요소를 if문과 for문의 반복 수행을 통해 찾은 후 , 해당 요소의 빈도수를 재설정, dataTable의 노드 속의 빈도수도 함께 재설정한다.

fp tree 생성

data.txt로부터 데이터를 다시 읽어와 전과 같은 방식으로 item을 불러온다. 이때 한 줄이 fp tree 내에서의 한 경로가 되기 때문에 한줄 단위로 정렬을 위한 list <pair<int,string> forsort와 fp growth로 넘겨줄 item_array를 위한 list<string>forfp를 선언한다.

한 줄 씩 forsort list를 headertable과 같은 방식으로 생성 후, 이를 내림차순 정렬한다. for 문을 통해 forsort의 모든 요소를 돌면서 그 빈도수가 threshold 보다 크거나 같은 경우의 상품명을 통해 list fp를 생성한다. list fp가 완성되면 이를 createFPTree를 통해 넘겨 FPTree를 생성한다.

- **FPGrowth::createFPTreee**

curNode를 root로 설정하고, it 변수를 전달 받은 item_array의 첫번째 요소로 설정한다. 모든 item 요소에 접근하기 위한 for문을 사용하고, curMap 변수에는 curNode의 children을 저장 후 반복자 선언을 통해 curMap의 첫번째 요소에 접근한다.

curMap 내의 모든 요소에 접근하기 위한 for문을 또 하나 사용하여, 만약 item의 상품명과 같은 요소가 curMap 내에 존재한다면 curNode를 해당 노드로 변경, curNode의 빈도수를 증가한 후 반복문을 빠져나온다.

만약 item의 상품이 curMap 내에 없다면, 새로운 children을 생성한다. 새로운 FPNode인 newNode를 동적할당 후 이의 부모노드를 curNode로 설정한다. 또한 상품명과 빈도 정보를 설정해주고, curNode->pushchildren을 통해 해당 children을 curNode에 연결 생성한다. 생성된 노드는 같은 상품의 노드와 연결되어야 하므로 connecNode함수를 호출한 후 curNode를 newNode로 변경한다.

- **FPGrowth::connectNode**

전달 받은 item 인자의 노드를 table 내에서 getNode함수를 통해 찾아 이를 curNode로 설정한다. 만약 curNode의 next 노드가 없다면 curNode의 next로 전달받은 node를 설정한다. 만약 curNode의 next 노드가 있다면, 이를 curNode로 재설정해주고 curNode의 next 노드가 없을 때까지 반복하여 curNode가 마지막으로 연결된 node로 이동할 수 있도록 한다. 마지막 노드에 접근하면 curNode의 next로 인자로 전달받은 node를 설정한다.

PRITNT_ITEMLIST

printlist 함수를 호출해 이에 성공할 시 true를 실패할 시 false를 반환한다.

- **FPGrowth::printList**

  만약 headertable이 비어있을 시 false를 반환한다.

  그렇지 않을 시, indextTable의 첫번째 요소를 시작으로 for문을 통해 indexTable의 모든 요소에 접근해 그 상품명과 빈도수를 출력한다.

PRITNT_FPTREE

printTree 함수를 호출해 이에 성공할 시 true를 실패할 시 false를 반환한다.

- **FPGrowth::printTree**

  만약 fpTree가 비어있을 시 false를 반환한다.

  그렇지 않을 시, indextTable의 첫번째 요소를 시작으로 for문을 통해 indexTable의 모든 요소에 접근한다.

  이때 indexTable의 요소 중 그 빈도수가 threshold와 같은 값인지 확인하여 그런 경우에만 print를 수행한다. Table 내 해당 요소의 이름과 빈도수를 {} 형식으로 출력한다. curNode를 해당 요소와 연결된 Node의 Next 노드로 설정하고 모든 경로의 체크를 위하여 하나의 curNode2를 더 설정해 curNode값을 복사한다.

  curNode2가 존재하는 동안 반복하는 while문 내에 curNode가 root가 아닌 동안 반복하는 while문을 하나 더 이용한다. curNode의 정보를 () 형식으로 출력하여 path 내 노드를 출력한 후 , curNode를 parent 노드로 옮기는 과정을 반복해 root에 도달할 때까지 path 내 모든 노드를 출력한다. 한 path의 출력이 완료되면 curNode2를 curNode2의 next노드로 변경해 다음 path를 출력할 수 있도록한다.

해당 과제에서 HeaderTable, fpTree의 구현까지밖에 완성하지 못했기 때문에, BTLOAD SAVE PRINT_RANGE PRINT_MIN PRINT_CONFIDENCE의 기능은 구현하지 못했다.

D. 실행결과

1) test case2

Data.txt

mineral water	milk	energy bar	whole wheat rice	green tea					
turkey	fresh tuna	tomatoes	spaghetti	mineral water	black tea	salmon	eggs	chicken	extra dark chocolate
meatballs	milk	honey	french fries	protein bar					
red wine	shrimp	pasta	pepper	eggs	chocolate	shampoo	soup		
spaghetti	mineral water	ham	body spray	pancakes	green tea				
burgers	grated cheese	shrimp	pasta	avocado	honey	white wine	toothpaste		
parmesan cheese	spaghetti	soup	avocado	milk	fresh bread				
ground beef	spaghetti	mineral water	milk	energy bar	black tea	salmon	frozen smoothie	escalope	
mineral water	eggs	chicken	chocolate	french fries	soup				
fresh tuna	frozen vegetables	spaghetti	mineral water	honey	whole wheat rice	frozen smoothie	escalope	soup	
soup	avocado	french fries	hot dogs	brownies	body spray	pancakes	green tea		
spaghetti	chocolate	brownies	white wine	green tea					
soup	chicken	gums	soda	body spray	energy drink	green tea			
soup	bug spray	shallot	protein bar	green tea					
grated cheese	ground beef	spaghetti	mineral water	white wine					
burgers	spaghetti	milk	french fries	green tea	soup				
burgers	almonds	eggs	french fries	cookies	green tea	soup			
frozen vegetables	whole wheat pasta	ground beef	spaghetti	chocolate	green tea	soup			
turkey	burgers	ground beef	chocolate	soup	almonds	eggs	hot dogs	cottage cheese	
soup	energy bar	pet food	carrots	protein bar					

Command.txt

```
LOAD
PRINT_ITEMLIST
PRINT_FPTREE
EXIT
```

Log.txt

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

======= PRINT_ITEMLIST =======
Item    Frequency
mineral water 625
spaghetti 480
chocolate 447
eggs 436
french fries 432
green tea 409
milk 378
frozen vegetables 317
ground beef 278
pancakes 275
burgers 261
cake 235
shrimp 210
tomatoes 204
low fat yogurt 189
frozen smoothie 189
chicken 180
whole wheat rice 175
turkey 173
escalope 160
fresh bread 143
soup 141
honey 134
grated cheese 129
salmon 128
cookies 123
champagne 118
cottage cheese 98
butter 98
hot dogs 95
brownies 94
avocado 92
ham 88
red wine 87
pepper 86
tomato juice 82
energy bar 77
light mayo 76
vegetables mix 74
french wine 73
cereals 67
almonds 67
whole wheat pasta 66
fresh tuna 66
energy drink 64
meatballs 60
yogurt cake 57
parmesan cheese 57
carrots 56
strawberries 54
rice 54
protein bar 50
mushroom cream sauce 48
muffins 47
fromage blanc 47
body spray 47
white wine 45
mint 45
eggplant 45
barbecue sauce 41
black tea 39
pasta 34
magazines 34
light cream 34
gums 33
cider 33
yams 32
tomato sauce 32
nonfat milk 32
zucchini 29
whole weat flour 29
flax seed 29
extra dark chocolate 28
green grapes 27
melons 26
bug spray 26
pet food 25
toothpaste 24
green beans 24
shallot 23
salt 23
clothes accessories 23
bacon 23
sparkling water 22
blueberries 22
strong cheese 20
```

(이후 생략)

```
======= PRINT_FPTREE =======
 {StandaraItem.Frequency} (Path_Itme.Frequency)
{napkins.3}
(napkins.1) (parmesan cheese.1) (low fat yogurt.1) (chocolate.52) (spaghetti.193) (mineral
water.625)
(napkins.1) (shampoo.1) (gluten free bar.1) (tomato sauce.1) (muffins.1) (light mayo.1)
(grated cheese.1) (frozen vegetables.24) (mineral water.625)
(napkins.1) (carrots.1) (almonds.1) (red wine.1) (avocado.1)
{bramble.5}
(bramble.1) (cottage cheese.1) (soup.1) (frozen smoothie.2) (shrimp.4) (milk.20) (spaghetti
193) (mineral water.625)
(bramble.1) (red wine.1) (turkey.1) (eggs.90) (mineral water.625)
(bramble.1) (pepper.1) (grated cheese.1) (cake.4) (chocolate.187)
(bramble.1) (cauliflower.1) (gluten free bar.1) (vegetables mix.1) (energy bar.1) (salmon.
1) (grated cheese.1) (escalope.1) (milk.12) (eggs.140)
(bramble.1) (fromage blanc.1) (fresh tuna.1) (cottage cheese.1) (honey.1) (escalope.1)
(turkey.1) (green tea.11) (eggs.55) (spaghetti.287)
{chutney.8}
(chutney.1) (yams.1) (protein bar.1) (energy drink.1) (whole wheat pasta.1) (cereals.1)
(soup.1) (ground beef.15) (spaghetti.193) (mineral water.625)
(chutney.1) (cider.1) (black tea.1) (french wine.1) (salmon.1) (frozen vegetables.2) (milk.
20) (spaghetti.193) (mineral water.625)
(chutney.1) (pet food.1) (cider.1) (soup.1) (whole wheat rice.1) (green tea.5) (eggs.35)
(spaghetti.193) (mineral water.625)
(chutney.1) (toothpaste.1) (meatballs.1) (escalope.1) (chicken.1) (frozen smoothie.1)
(pancakes.4) (frozen vegetables.15) (spaghetti.287)
(chutney.1) (salmon.1) (grated cheese.1) (whole wheat rice.1) (cake.3) (eggs.35) (chocolate
128) (mineral water.625)
(chutney.1) (cereals.1) (salmon.1) (grated cheese.1) (mineral water.625)
(chutney.1) (eggplant.1) (french wine.1) (chicken.2) (ground beef.7) (frozen vegetables.24)
(mineral water.625)
(chutney.1) (cereals.1) (grated cheese.1) (frozen vegetables.14) (chocolate.187)
{mint green tea.8}
(mint green tea.1) (salmon.1) (soup.1) (chicken.1) (low fat yogurt.1) (pancakes.5)
(spaghetti.193) (mineral water.625)
(mint green tea.1) (melons.1) (grated cheese.1) (turkey.1) (whole wheat rice.1) (shrimp.1)
(ground beef.2) (milk.16) (chocolate.128) (mineral water.625)
(mint green tea.1) (brownies.1) (cookies.1) (milk.8) (french fries.52) (eggs.140)
(mint green tea.1) (cottage cheese.1) (honey.1) (soup.1) (pancakes.3) (milk.35)
(mint green tea.1) (ham.1) (avocado.1) (cookies.1) (burgers.6) (french fries.52) (eggs.140)
(mint green tea.1) (tomato juice.1) (red wine.1) (eggs.55) (spaghetti.287)
(mint green tea.1) (fresh bread.1) (low fat yogurt.1) (shrimp.1) (cake.1) (burgers.3) (milk
20) (french fries.117)
(mint green tea.1) (ham.1) (low fat yogurt.1) (pancakes.2) (frozen vegetables.4) (french
fries.52) (eggs.140)
{mashed potato.10}
(mashed potato.1) (tomato juice.1) (pepper.1) (grated cheese.1) (cake.3) (eggs.35)
(chocolate.128) (mineral water.625)
```

(중략)

```
(green tea.1) (french fries.7) (eggs.35) (spaghetti.193) (mineral water.625)
(green tea.1) (eggs.12) (chocolate.52) (spaghetti.193) (mineral water.625)
{french fries.432}
(french fries.117)
(french fries.5) (eggs.35) (chocolate.128) (mineral water.625)
(french fries.31) (spaghetti.287)
(french fries.52) (eggs.140)
(french fries.8) (chocolate.52) (spaghetti.193) (mineral water.625)
(french fries.7) (eggs.35) (spaghetti.193) (mineral water.625)
(french fries.23) (eggs.45) (chocolate.187)
(french fries.53) (mineral water.625)
(french fries.10) (chocolate.79) (spaghetti.287)
(french fries.52) (chocolate.187)
(french fries.22) (chocolate.128) (mineral water.625)
(french fries.6) (eggs.24) (chocolate.79) (spaghetti.287)
(french fries.12) (eggs.55) (spaghetti.287)
(french fries.20) (spaghetti.193) (mineral water.625)
(french fries.2) (eggs.12) (chocolate.52) (spaghetti.193) (mineral water.625)
(french fries.12) (eggs.90) (mineral water.625)
{eggs.436}
(eggs.35) (spaghetti.193) (mineral water.625)
(eggs.45) (chocolate.187)
(eggs.35) (chocolate.128) (mineral water.625)
(eggs.140)
(eggs.90) (mineral water.625)
(eggs.55) (spaghetti.287)
(eggs.24) (chocolate.79) (spaghetti.287)
(eggs.12) (chocolate.52) (spaghetti.193) (mineral water.625)
{chocolate.447}
(chocolate.187)
(chocolate.128) (mineral water.625)
(chocolate.79) (spaghetti.287)
(chocolate.52) (spaghetti.193) (mineral water.625)
(chocolate.1) (chocolate.52) (spaghetti.193) (mineral water.625)
{spaghetti.480}
(spaghetti.193) (mineral water.625)
(spaghetti.287)
{mineral water.625}
(mineral water.625)
==========================
======== EXIT ========
Success
======================
```

E. 고찰

FP growth의 자료구조에 대해 이해하고 이를 직접 설계해보는 프로젝트였다. BST 나 linked list가 아닌 다양한 자료구조를 접한 것이 이번 학기가 처음이라, 그 개념과 새로운 노드 추가 등의 작동 원리를 이해하는 것이 쉽지만은 않았다. fp growth의 경우 개념 자체는 그렇게 어렵지 않은 내용이었지만 이를 포인터를 통해 직접 클래스로 구현하여 사용하려고 하니 고려해야할 부분이 많아 프로젝트를 수행하는데 많은 시간과 노력이 들었다.

먼저 HeaderTable을 list와 map으로 나누어 저장하다보니 하나의 개념으로 생각하고 있었던 headertable도 코드내에서 둘과의 관계를 모두 고려하여 작성해야 한다는 점이 실수를 만들기 쉬웠다. 또한 fp growth 내의 chlidren들의 상관 관계가 잘 와닿지 않아, 한 chidlren인 하나의 map 내부에 같은 level의 노드가 저장되어 있는 구조로 구현해야 하는 부분을 이해하기 어려운 점이 있었다.

fp growth를 모두 구현한 후에도 각각의 노드를 같은 상품의 노드끼리 link해줘야 했었는데, 그때 하나의 table 요소에서 나온 노드의 모든 path와 이를 연결해야하고 이를 출력해야한다는 점이 어려운 점이었다.

Bptree 구조를 구현하고자 노력해보았으나, 노드를 삽입할 때의 split 기능의 포인터 구조가 바뀌는 부분을 구현하는 것이 쉽지 않았다. 이번 과제에서는 Btree 구조를 구현하는 것을 완성하지 못했지만, 앞으로 자료구조에 대해 더 많은 노력을 들여 다음 프로젝틑 꼭 완성해보고 싶다.