

데이터 구조 설계 프로젝트2 보고서

제출일자: 2022년 11월 20일 (일)

학과: 컴퓨터정보공학부

담당교수: 이기훈교수님

학번: 2021202045

이름: 김예은

1. Introduction







이번 2차 프로젝트는 크게 Fptree와 B+tree 자료구조를 이용하여 상품 추천 프로그램을 구현한다. 먼저 market.txt에는 사람들이 구매한 거래내역이 한 줄씩 적혀있다. 각 상품마다 '\t'를 기준으로 구분되어있고, 사람들이 어떤 상품을 주로 같이 구매하는지 frequent pattern을 찾기 위해 fp-tree를 구현한다. 이는 먼저 header table을 구현한 뒤에 tree를 구현할 수 있다. header table은 상품명과, 그 상품이 전체 거래 내역 중에 얼마나 등장했는지 표시해주는 frequency부분으로 이루어진 indexTable과 해당 item들과 fp-tree 노드들을 연결해주는 FpNode*형이 짝지어진 dataTable로 구성되어있다. indexTable은 list<pair<int,string>>로 구현하여 알고리즘의 목적마다 알맞게 key값을 기준으로 내림차순이나 오름차순으로 바꿀 수 있게 하고, dataTable의 경우 map<string, FpNode*>을 이용하여 상품명들의 중복을 허용하지 않았다. indexTable을 내림차순(get smaller)으로 정렬한 뒤 이를 이용하여 각 거래내역에서 frequency순서가 큰 순서가 앞으로 상품들을 정렬 후 그 순서대로 fp-tree에 연결시켜준다. fp-tree의 상품명 및 그 순서가 중복된다면(단, 처음부터 같아야함) frequency값만 update해준다. fp-tree를 이용하여 찾은 frequent pattern들을 result.txt에 '\t'를 기준으로 출력하고 이를 이용하여 B+-tree를 구현한다. B+-tree는 기존 이진트리에 비해 degree값이 커지면서 더 많은 정보를 적은 level에 담을 수 있고, 각 노드들에 대한 정보를 리프노드에 2중 linked list로 정렬해주기 때문에 순회나 방문이 쉽다는 장점이 있다. b+tree는 indexNode와 dataNode로 나뉘는데 indexNode는 말그대로 root부터 해당 값이 어디쪽에 연결되어있는지 길잡이 역할을 해주고, dataNode는 key값에 따라 정렬된 상태로 이중 링크드 리스트에 연결되어 있다. b+tree의 특징은 order m이 만약 3이라 하면, m-1즉 2개의 요소만 가질 수 있고, degree는 3이다. 만약 요소의 수가 2보다 커지면 split을 통해 노드들을 나누어줘야 한다. 이렇게 구현한 두 자료구조는 command.txt에 연결된 명령어에 따라 목적에 맞게 print해준다. indexTable부분을 출력해주는 부분, confidence 범위에 맞게 출력해주는 부분, 원하는 frequency부분을 출력하는 부분 등 여러 가지 기능이 있다.

2. Algorithm

(1) Header Table

Header Table

list<pair<int, string>> indexTable
map<string, FpNode*> dataTable

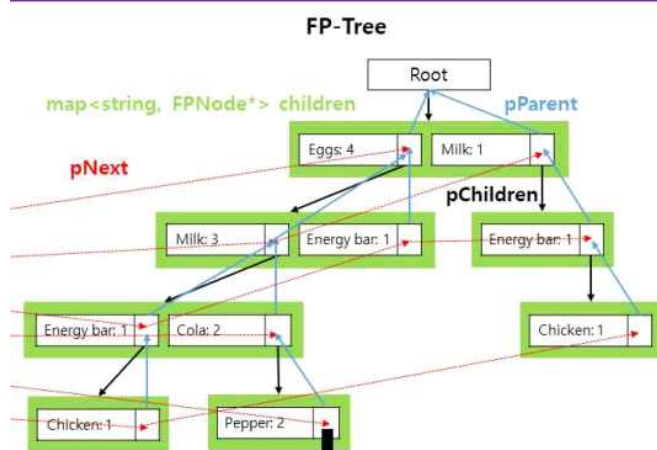
Frequency	Item	Pointer
4	Eggs	
4	Milk	
3	Energy Bar	
2	Cola	
2	Pepper	
2	Chicken	

FP-Tree를 create하기 전에 먼저 "market.txt"에서 모든 거래내역에 나와있는 상품과 각 상품의 총 빈도수를 정리하는 table을 구현해야한다. frequency와 item으로 이루어진 indexTable과 string, FpNode*로 이루어진 dataTable로 나뉜다. market.txt에서 line은 한 거래내역을 뜻하며 한 줄씩 getline을 통해 거래내역을 읽어오고 '\t'로 나뉜 상품들을 strtok

을 통해 추출하여 각 상품을 list<pair<int,string>>에 insert하였다. 맨처음에 변수 count=0을 선언 및 초기화 해준 뒤 table에 값을 입력할 때마다 count값을 증가시켜줬다. 즉, table인 empty인지 알기 위해 선언해준 변수이다. item을 table에 insert해주기 전에 이미 indexTable에 있는 item이라면, 새로 insert를 해주는 것이 아닌 해당 item과 연결된 frequency(pair로 연결되어있음) 값을 +1로 update만 해준다. dataTable은 map으로 item과 FPNode*로 이루어져있는데 새로 insert해줄때마다 new FPNode를 동적 할당해주어 연결해준다.

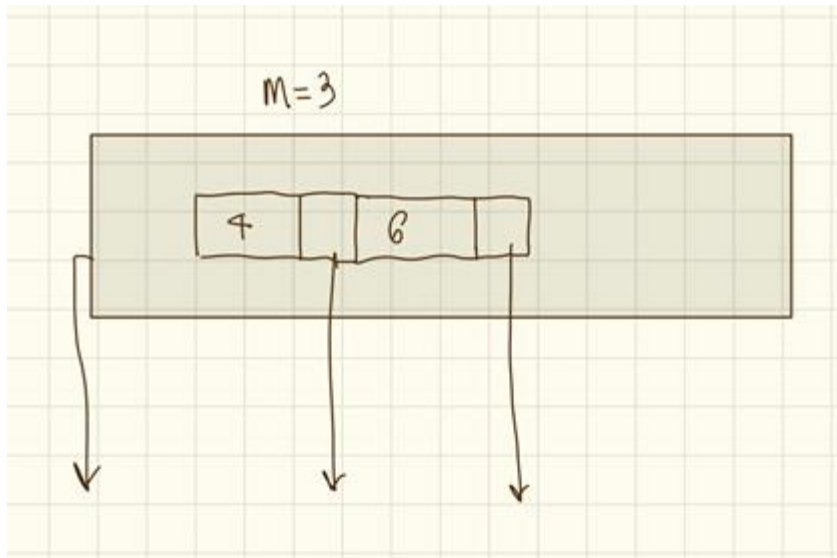
(2) FP-TREE

LOAD 명령어를 입력받을 때, 위에서 설명한 알고리즘을 통해 먼저 table을 생성해준 후, table 값에 맞게 fp-tree를 구현한다. 동일하게 “market.txt”에서 line by line으로 거래내역을 받아와서 strtok을 통해 상품들을 list<string> trans_list에 insert한다. 거래내역에 하나에 부합하는 상품들을 인자로 가지는 createFPtree함수에 넣으면 multimap<int, string, greater<int>> m을 선언해주어 table에 저장된 각 item에 대한 총 frequency의 크기가 큰 순서대로 item들을 정리하여 insert해준다. 여기서 frequent를 key값으로 자동으로 sorting해주는데, key값이 같을 때 value값도 정리를 해주고 싶었기 때문에 key값에 따라 정돈된 multimap m을 vector v로 바꿔준 뒤, 변수 count를 사용하여 vecotr v를 처음부터 끝까지 순회하며 key값이 같은 요소가 몇 개 있는지 센다. count값을 구했으면 count값동안 vector v를 순회하여 key값이 같다면->string끼리 비교하고 swap을 해줘야 하면 swap을 해줌으로써 string끼리도 sorting되도록 해주었다. 이렇게 정돈된 vecotr v에는 거래내역 한 줄에 대한 정보가 담겨있다. 이제 fptree를 구현할 때 frequency가 높은 순서대로 tree에 넣을 것이다. 이때 check라는 변수를 통해 fptree가 비었는지 알 수 있다. fptree는 다음 사진과 같은 구조를 가지고 있다.



map<string, FPNode*>를 children으로 가지고 있고, 안에 여러개의 <string, FPNode*>요소들이 들어있다. 상품 하나 하나를 children의 요소와 비교하면서 insert해줘야 하는데, 예를 들어 위의 그래프가 이미 구현된 상태에서 Eggs, Milk, Cider을 넣고싶다면 Eggs의 경우 Root의 children 노드에 이미 있음을 확인할 수 있다.(getChildrenNode라는 함수를 이용해 구현) 따라서 그냥 frequency만 update해주면 된다. Milk또한 Eggs의 children 노드에 있기 때문에 frequency만 update해준다. 하지만 그 다음 Cider의 경우 Milk의 children노드에 없기 때문에 FPNode를 새로 하나 동적 할당해준 후, 새 노드의 frequency, item을 set해준 후, 새 노드의 부모노드를 Milk랑 연결해주면 된다. 그 다음 같은 item 노드끼리 연결해주면 된다. 위의 사진에서도 빨간 선으로 같은 item끼리 연결리스트 형태로 연결되어있음을 볼 수 있다. 이는 connectNode 함수를 통해 구현하였다. 같은 item 노드끼리도 연결해줘야하지만 dataTable과 fp tree도 연결해줘

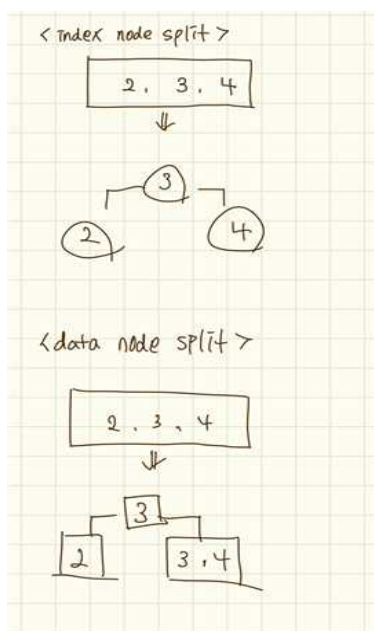
root부터 index node들을 타고 내려와야한다. 위의 설명에서 말했듯이 order가 m 인 b+ tree는 components를 $m-1$ 만 가질 수 있다. 만약 $m=3$ 이라면 아래와 같이 갈 수 있는 경로가 나뉜다.



count = 1으로 선언 및 초기화를 해준다. count가 0이고, key(삽입할 frequency)값이 요소값보다 작다면 맨 왼쪽으로 가고, key값이 요소값보다 작지만 count가 1이 아닐 땐 그 전의 요소값의 아래로 내려간다. 만약 key값과 요소값이 같다면 그 요소의 아래로 내려가고, 맨끝으로 갔는데도 key값이 더 크다면 그 맨 끝의 요소값의 아래로 내려간다. 즉 총 case를 4개로 나누어줬고 아무 case에도 해당되지 않으면 다음 요소로 이동해서 다시 비교하도록 해주었다. 그렇게 알맞은 위치에 data node를 넣었는데 요소가 $m-1$ 보다 커져서 초과된다면 splitDataNode함수를 통해 data node를 split 해준다.

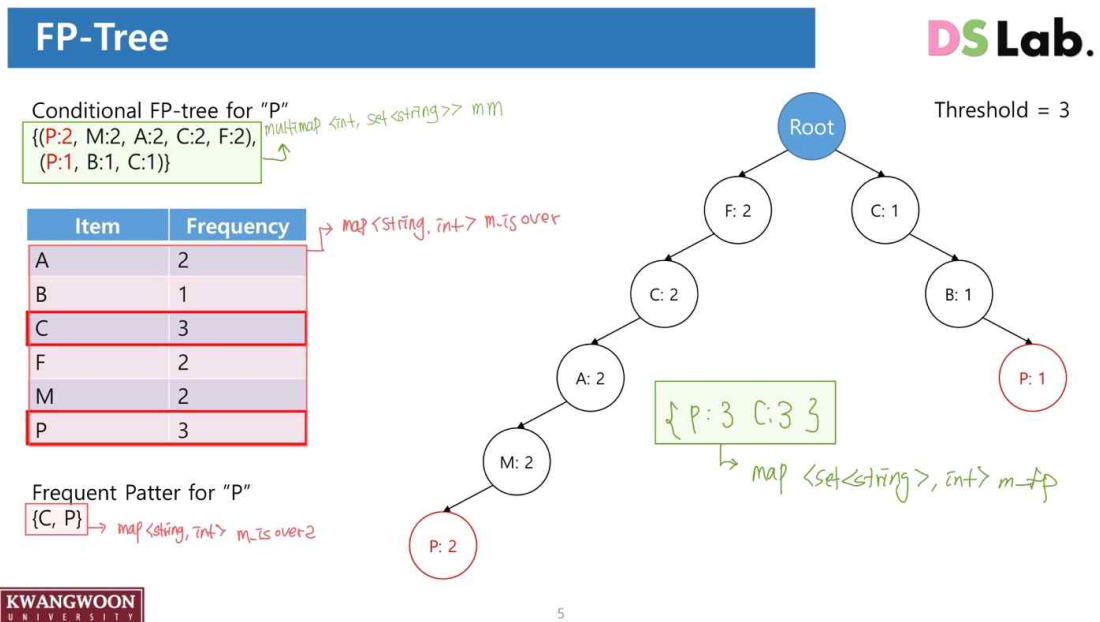
(4) split

data node 부분에서 요소가 초과되어 split을 해줄 때는 split의 기준이 되는 key값이 index node로 올라감과 동시에 해당 split key가 data node에 남아서 같이 split된다는 점이 index node에서의 split과 차이점이다. 밑의 그림을 참고하면 된다.



split_key는 $\text{ceil}((\text{order}-1)/2.0)+1$ 식을 이용하여 구해주고, split_key보다 큰 key값을 가진 요소를 옮겨줄 data 또는 index node를 새로 할당해주어 옮겨준다. 그리고 기존의 split해줘야하는 노드에서 옮긴 key들은 erase를 통해 지워준다. index node를 split할 경우 새 노드를 만들어서 옮긴 후 그 노드의 자식노드를 새 노드로 부모 포인터를 가리키게 고쳐주었다. split한 뒤 그 노드의 부모노드가 원래 없었다면 부모노드를 새로 만들어주어 그 노드가 새로 만들어준 노드가 되게 해주었고, 아닐 때는 부모노드에 옮겨준 부분을 부모에 추가해주었다. 모든 과정이 끝난 다음 기존 root의 부모가 NULL이 아닐 시 `root->getParent`가 NULL일때까지 반복문을 돌려주어 root를 바꾸어 준다.

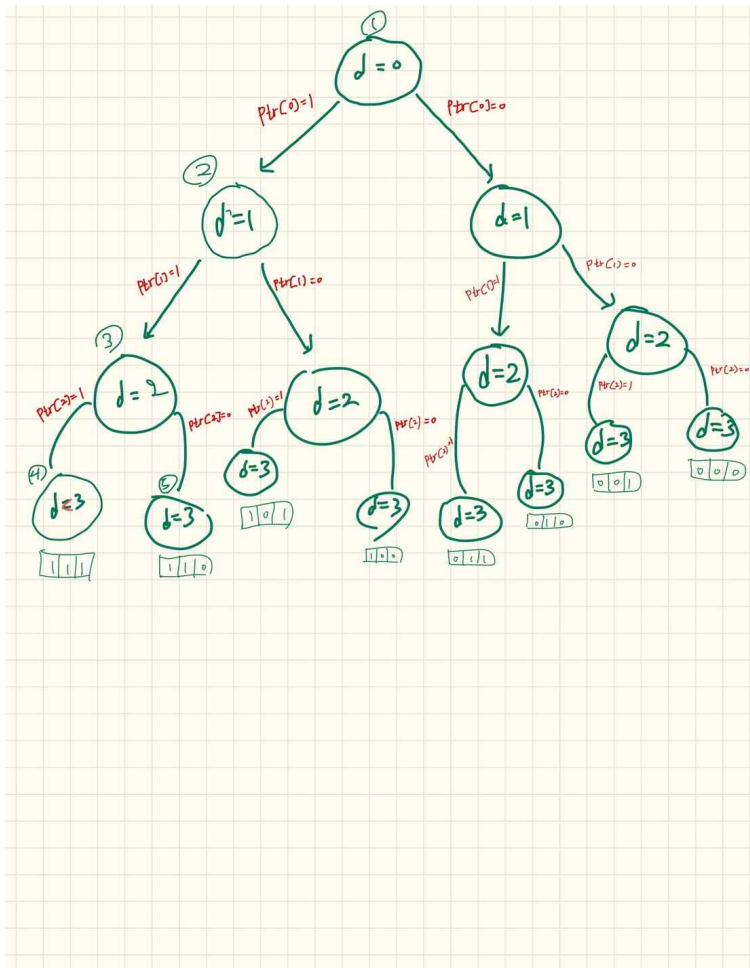
(5) save



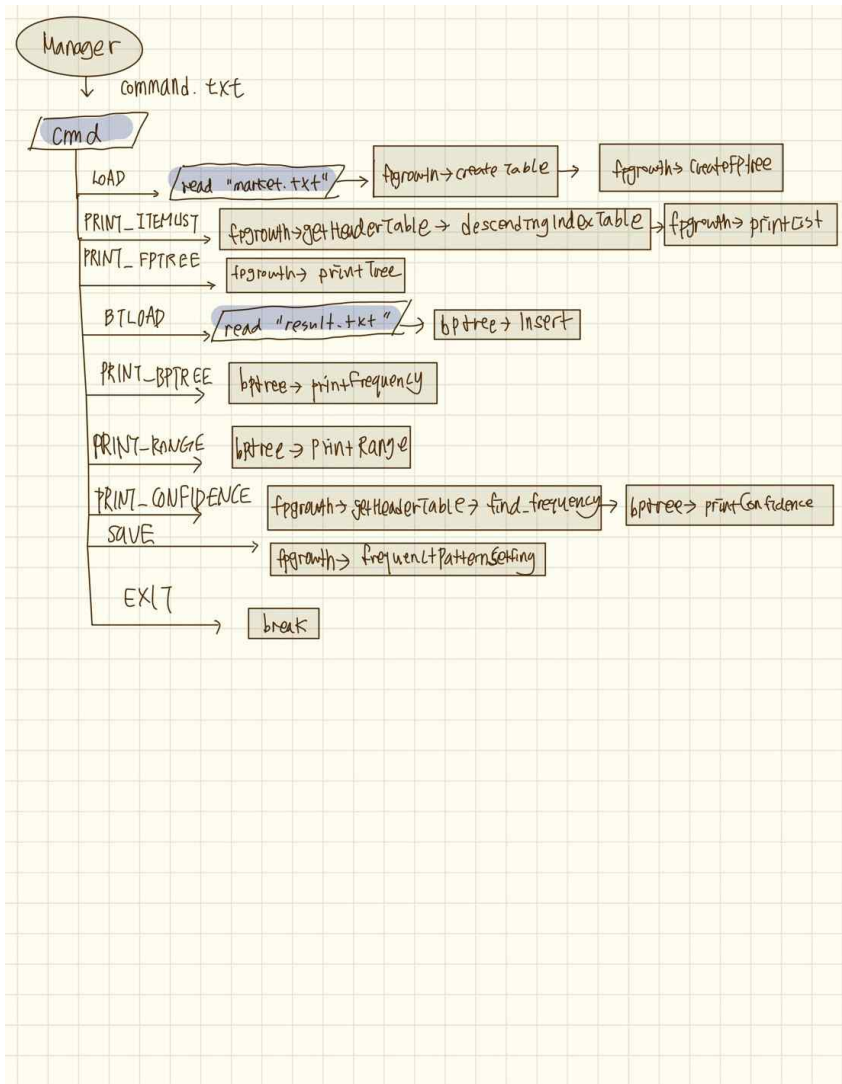
위 그림은 save부분에서 내가 짠 코드를 표시한 그림이다.

mm에 conditional FP-tree for 특정 상품에 대한 집합을 저장하고, m_isover에 같은 item끼리 frequency를 더해준다. 이중에서 threshold보다 큰 frequency를 갖는 item만 m_isover2에 저장하고 다시 mm으로 가서 m_isover2에 저장된 item이 mm에 있으면 그 부분만 추출하여 `set<string>fp`에 넣는다. 만약 m_fp에 `set<string>fp`가 없다면 insert해주고 있다면 해당 frequency만 update해준다. 이 m_fp를 powerSet 알고리즘에 넣어주어 멍집합에 대한 부분집합을 구해준다. 단, 원소가 1이하일때는 구하지않는다. 만약 이미 있는 부분집합이라면 frequency만 update하고 다시 추가하지는 않는다.

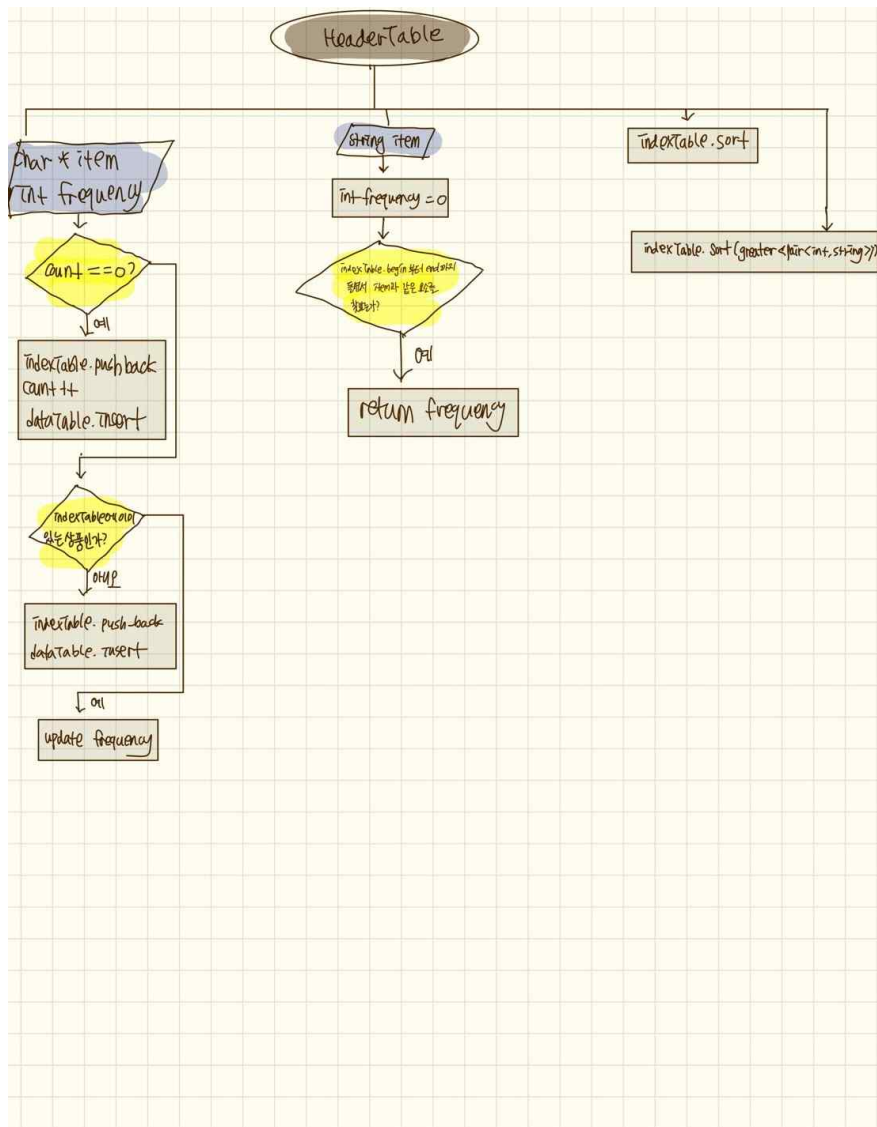
powerSet의 알고리즘은 재귀함수를 이용하여 멍집합을 구하는 것으로 다음과 같다.



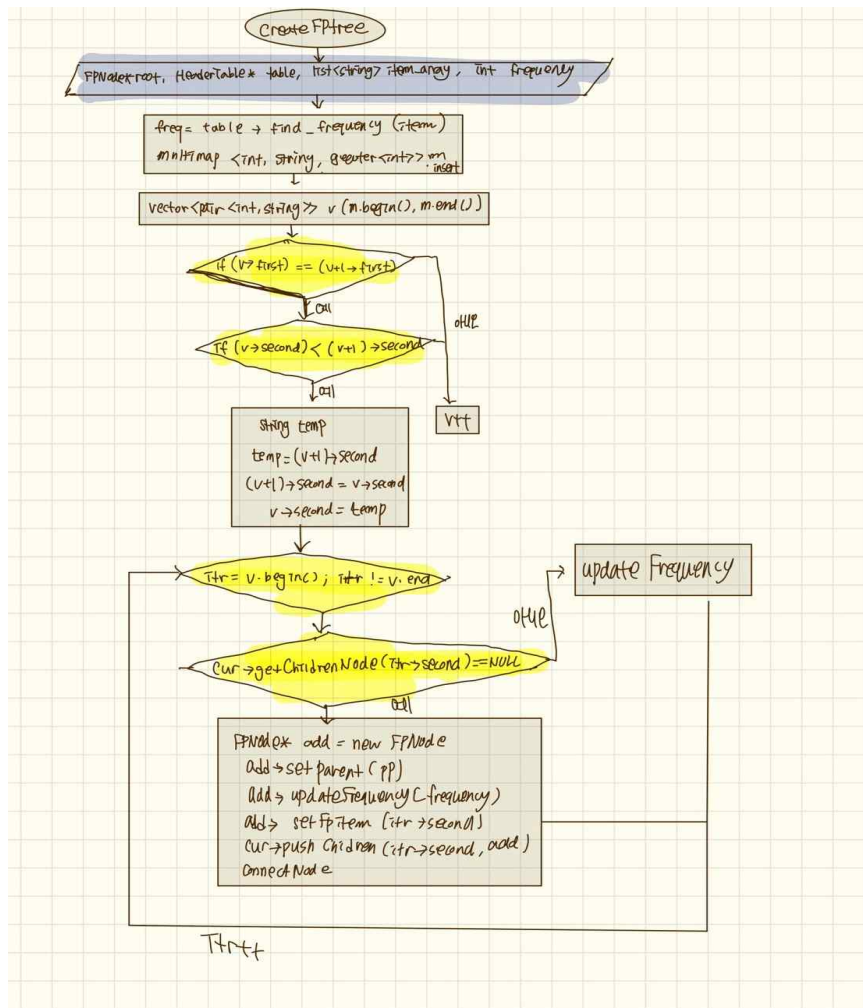
3. Flowchart



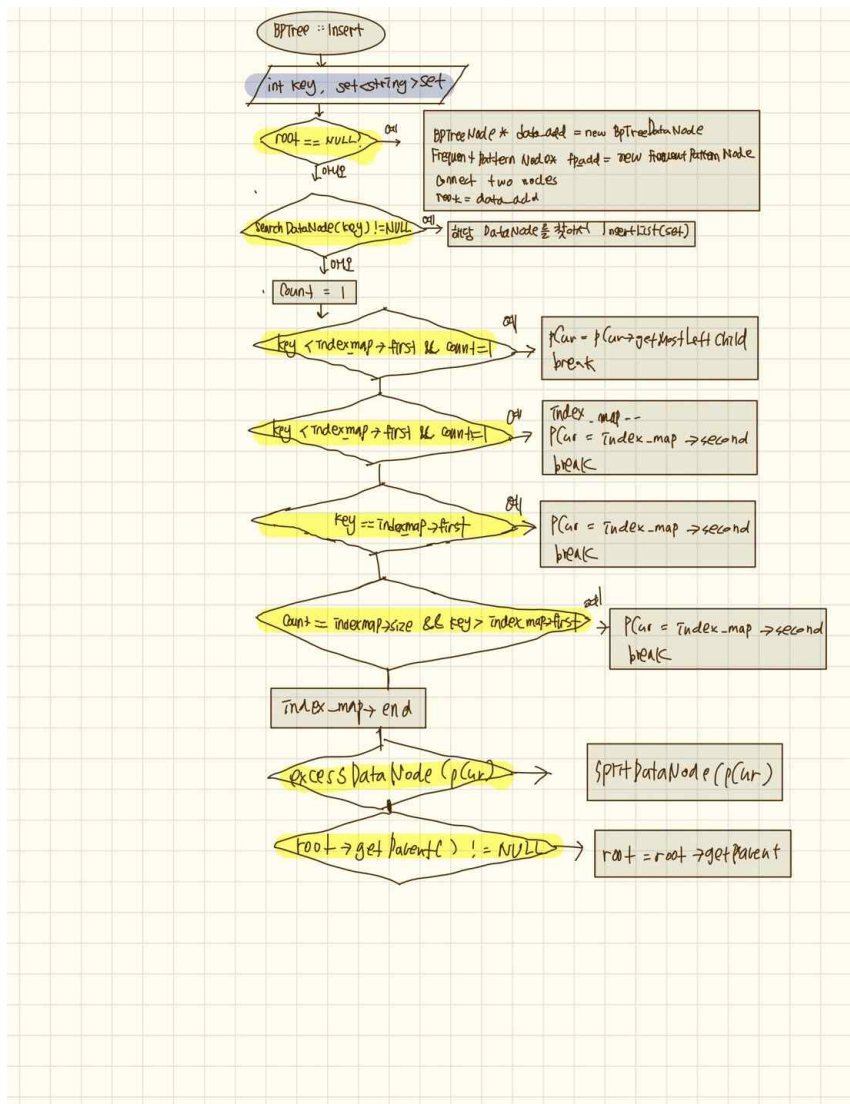
위는 "command.txt"를 읽고 각 command에 따라 동작하는 과정을 간단하게 그린 것이다. 명령어에 따라 각각의 함수로 들어가서 동작한다.



위는 맨 처음에 item과 frequency를 인자로 받으면 count가 0일 때와 아닐 때로 나뉘어 indexTable과 dataTable을 각각 만들어준다. indexTable의 경우 list형으로 push_back을 통해 인자를 삽입하고 dataTable의 경우 map형식이므로, insert를 통해 인자를 삽입한다.



위는 FPtree를 구현하는 flow chart로 인자값으로 받은 item_array를 indexTable의 frequency 값에 따라 정돈하고 이를 또 value값인 string에 따라 정돈한 vector v를 iterator를 통해 순회하면서 각 item들을 fp tree에 넣는다. 만약 item을 넣기 전 childrenNode에 item이 있다면 update Frequency만 진행하고 아닐 시에는 새로 노드를 할당하여 알맞게 setting 후 기존 fp tree에 삽입해준다.



위는 b+-tree를 구현하는 flow chart로, root가 null일 때와 null이 아닐 때로 맨처음에 나누어 준 다음, root가 아닐 시에 해당 frequency가 이미 data node에 연결되어 있는지 확인해준다. 있을 경우 frequent pattern node만 update해주고, 없을 시에는 위의 알고리즘 설명에서 적었듯이 네가지의 경우에 따라 알맞은 위치로 가서 data node를 삽입해준다. 만약 datanode가 초과되었다면 split을 해주고 그 다음 root를 이동해줘야 할 때 root를 change해준다.

4. Result Screen ->testcase1

(1)LOAD 및 PRINT_ITEMLIST

```
Open ▾ log.txt
~/project2

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

=====PRINT_ITEMLIST=====
Item      Frequency
soup      12
spaghetti  9
green tea  9
mineral water  7
milk      5
french fries  5
eggs      5
chocolate  5
ground beef  4
burgers    4
white wine  3
protein bar  3
honey      3
energy bar  3
chicken    3
body spray  3
avocado    3
whole wheat rice  2
turkey     2
shrimp     2
salmon     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    2
whole wheat pasta  1
toothpaste  1
tomatoes   1
soda       1
shampoo    1
shallot    1
red wine   1
pet food   1
pepper     1
parmesan cheese  1
meatballs  1
ham        1
gums       1
fresh bread  1
extra dark chocolate  1
energy drink  1
cottage cheese  1
cookies    1
carrots    1
bug spray  1
=====
-----
```

LOAD가 성공한 success코드 및 ITEM_LIST가 내림차순으로 잘 출력되는 것을 볼 수 있다.

(2) PRINT_FPTREE

```

=====PRINT_FPTREE=====
{StandardItem.Frequency}      (Path_Item.Frequency)
{bug spray, 1}
(bug spray, 1) (shallot, 1) (protein bar, 1) (green tea, 4) (soup, 12)
{carrots, 1}
(carrots, 1) (pet food, 1) (energy bar, 1) (protein bar, 1) (soup, 12)
{cookies, 1}
(cookies, 1) (almonds, 1) (burgers, 1) (eggs, 1) (french fries, 2) (green tea, 4) (soup, 12)
{cottage cheese, 1}
(cottage cheese, 1) (almonds, 1) (hot dogs, 1) (turkey, 1) (burgers, 1) (ground beef, 1)
(chocolate, 2) (eggs, 2) (soup, 12)
{energy drink, 1}
(energy drink, 1) (gums, 1) (soda, 1) (body spray, 1) (chicken, 1) (green tea, 4) (soup, 12)
{extra dark chocolate, 1}
(extra dark chocolate, 1) (tomatoes, 1) (black tea, 1) (fresh tuna, 1) (salmon, 1) (turkey, 1)
(chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
{fresh bread, 1}
(fresh bread, 1) (parmesan cheese, 1) (avocado, 1) (milk, 1) (spaghetti, 4) (soup, 12)
{gums, 1}
(gums, 1) (soda, 1) (body spray, 1) (chicken, 1) (green tea, 4) (soup, 12)
{ham, 1}
(ham, 1) (pancakes, 1) (body spray, 1) (mineral water, 1) (green tea, 2) (spaghetti, 5)
{meatballs, 1}
(meatballs, 1) (honey, 1) (protein bar, 1) (french fries, 1) (milk, 1)
{parmesan cheese, 1}
(parmesan cheese, 1) (avocado, 1) (milk, 1) (spaghetti, 4) (soup, 12)
{pepper, 1}
(pepper, 1) (red wine, 1) (shampoo, 1) (pasta, 1) (shrimp, 1) (chocolate, 2) (eggs, 2) (soup, 12)
{pet food, 1}
(pet food, 1) (energy bar, 1) (protein bar, 1) (soup, 12)
{red wine, 1}
(red wine, 1) (shampoo, 1) (pasta, 1) (shrimp, 1) (chocolate, 2) (eggs, 2) (soup, 12)
{shallot, 1}
(shallot, 1) (protein bar, 1) (green tea, 4) (soup, 12)
{shampoo, 1}
(shampoo, 1) (pasta, 1) (shrimp, 1) (chocolate, 2) (eggs, 2) (soup, 12)
{soda, 1}
(soda, 1) (body spray, 1) (chicken, 1) (green tea, 4) (soup, 12)
{tomatoes, 1}
(tomatoes, 1) (black tea, 1) (fresh tuna, 1) (salmon, 1) (turkey, 1) (chicken, 1) (eggs, 1)
(mineral water, 3) (spaghetti, 5)
{toothpaste, 1}
(toothpaste, 1) (grated cheese, 1) (pasta, 1) (shrimp, 1) (avocado, 1) (honey, 1) (white wine, 1)
(burgers, 1)
{whole wheat pasta, 1}
(whole wheat pasta, 1) (frozen vegetables, 1) (ground beef, 1) (chocolate, 1) (green tea, 2)
(spaghetti, 4) (soup, 12)
{almonds, 2}
(almonds, 1) (burgers, 1) (eggs, 1) (french fries, 2) (green tea, 4) (soup, 12)
(almonds, 1) (hot dogs, 1) (turkey, 1) (burgers, 1) (ground beef, 1) (chocolate, 2) (eggs, 2)
(soup, 12)
{black tea, 2}
(black tea, 1) (fresh tuna, 1) (salmon, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 3)
(spaghetti, 5)
(black tea, 1) (escalope, 1) (frozen smoothie, 1) (salmon, 1) (energy bar, 1) (ground beef, 1)
(milk, 1) (mineral water, 3) (spaghetti, 5)
{brownies, 2}
(brownies, 1) (hot dogs, 1) (pancakes, 1) (avocado, 1) (body spray, 1) (french fries, 2) (green
tea, 4) (soup, 12)
(brownies, 1) (white wine, 1) (chocolate, 1) (green tea, 2) (spaghetti, 5)
{escalope, 2}
(escalope, 1) (frozen smoothie, 1) (salmon, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral
water, 3) (spaghetti, 5)
(escalope, 1) (fresh tuna, 1) (frozen smoothie, 1) (frozen vegetables, 1) (whole wheat rice, 1)
(honey, 1) (mineral water, 1) (spaghetti, 4) (soup, 12)
{fresh tuna, 2}
(fresh tuna, 1) (salmon, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
(fresh tuna, 1) (frozen smoothie, 1) (frozen vegetables, 1) (whole wheat rice, 1) (honey, 1)

```

(fresh tuna, 1) (salmon, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
 (fresh tuna, 1) (frozen smoothie, 1) (frozen vegetables, 1) (whole wheat rice, 1) (honey, 1)
 (mineral water, 1) (spaghetti, 4) (soup, 12)
 {frozen smoothie, 2}
 (frozen smoothie, 1) (salmon, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 3)
 (spaghetti, 5)
 (frozen smoothie, 1) (frozen vegetables, 1) (whole wheat rice, 1) (honey, 1) (mineral water, 1)
 (spaghetti, 4) (soup, 12)
 {frozen vegetables, 2}
 (frozen vegetables, 1) (whole wheat rice, 1) (honey, 1) (mineral water, 1) (spaghetti, 4) (soup,
 12)
 (frozen vegetables, 1) (ground beef, 1) (chocolate, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
 {grated cheese, 2}
 (grated cheese, 1) (pasta, 1) (shrimp, 1) (avocado, 1) (honey, 1) (white wine, 1) (burgers, 1)
 (grated cheese, 1) (white wine, 1) (ground beef, 1) (mineral water, 3) (spaghetti, 5)
 {hot dogs, 2}
 (hot dogs, 1) (pancakes, 1) (avocado, 1) (body spray, 1) (french fries, 2) (green tea, 4) (soup,
 12)
 (hot dogs, 1) (turkey, 1) (burgers, 1) (ground beef, 1) (chocolate, 2) (eggs, 2) (soup, 12)
 {pancakes, 2}
 (pancakes, 1) (body spray, 1) (mineral water, 1) (green tea, 2) (spaghetti, 5)
 (pancakes, 1) (avocado, 1) (body spray, 1) (french fries, 2) (green tea, 4) (soup, 12)
 {pasta, 2}
 (pasta, 1) (shrimp, 1) (chocolate, 2) (eggs, 2) (soup, 12)
 (pasta, 1) (shrimp, 1) (avocado, 1) (honey, 1) (white wine, 1) (burgers, 1)
 {salmon, 2}
 (salmon, 1) (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
 (salmon, 1) (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 3) (spaghetti, 5)
 {shrimp, 2}
 (shrimp, 1) (chocolate, 2) (eggs, 2) (soup, 12)
 (shrimp, 1) (avocado, 1) (honey, 1) (white wine, 1) (burgers, 1)
 {turkey, 2}
 (turkey, 1) (chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
 (turkey, 1) (burgers, 1) (ground beef, 1) (chocolate, 2) (eggs, 2) (soup, 12)
 {whole wheat rice, 2}
 (whole wheat rice, 1) (energy bar, 1) (milk, 1) (mineral water, 1) (green tea, 1)
 (whole wheat rice, 1) (honey, 1) (mineral water, 1) (spaghetti, 4) (soup, 12)
 {avocado, 3}
 (avocado, 1) (honey, 1) (white wine, 1) (burgers, 1)
 (avocado, 1) (milk, 1) (spaghetti, 4) (soup, 12)
 (avocado, 1) (body spray, 1) (french fries, 2) (green tea, 4) (soup, 12)
 {body spray, 3}
 (body spray, 1) (mineral water, 1) (green tea, 2) (spaghetti, 5)
 (body spray, 1) (french fries, 2) (green tea, 4) (soup, 12)
 (body spray, 1) (chicken, 1) (green tea, 4) (soup, 12)
 {chicken, 3}
 (chicken, 1) (eggs, 1) (mineral water, 3) (spaghetti, 5)
 (chicken, 1) (chocolate, 1) (eggs, 1) (french fries, 1) (mineral water, 1) (soup, 12)
 (chicken, 1) (green tea, 4) (soup, 12)
 {energy bar, 3}
 (energy bar, 1) (milk, 1) (mineral water, 1) (green tea, 1)
 (energy bar, 1) (ground beef, 1) (milk, 1) (mineral water, 3) (spaghetti, 5)
 (energy bar, 1) (protein bar, 1) (soup, 12)
 {honey, 3}
 (honey, 1) (protein bar, 1) (french fries, 1) (milk, 1)
 (honey, 1) (white wine, 1) (burgers, 1)
 (honey, 1) (mineral water, 1) (spaghetti, 4) (soup, 12)
 {protein bar, 3}
 (protein bar, 1) (french fries, 1) (milk, 1)
 (protein bar, 1) (green tea, 4) (soup, 12)
 (protein bar, 1) (soup, 12)
 {white wine, 3}
 (white wine, 1) (burgers, 1)
 (white wine, 1) (chocolate, 1) (green tea, 2) (spaghetti, 5)
 (white wine, 1) (ground beef, 1) (mineral water, 3) (spaghetti, 5)
 {burgers, 4}
 (burgers, 1)
 (burgers, 1) (french fries, 1) (milk, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
 (burgers, 1) (eggs, 1) (french fries, 2) (green tea, 4) (soup, 12)
 (burgers, 1) (ground beef, 1) (chocolate, 2) (eggs, 2) (soup, 12)
 {ground beef, 4}
 (ground beef, 1) (milk, 1) (mineral water, 3) (spaghetti, 5)

```

(ground beef, 1) (milk, 1) (mineral water, 3) (spaghetti, 5)
(ground beef, 1) (mineral water, 3) (spaghetti, 5)
(ground beef, 1) (chocolate, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
(ground beef, 1) (chocolate, 2) (eggs, 2) (soup, 12)
{chocolate, 5}
(chocolate, 2) (eggs, 2) (soup, 12)
(chocolate, 1) (eggs, 1) (french fries, 1) (mineral water, 1) (soup, 12)
(chocolate, 1) (green tea, 2) (spaghetti, 5)
(chocolate, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
{eggs, 5}
(eggs, 1) (mineral water, 3) (spaghetti, 5)
(eggs, 2) (soup, 12)
(eggs, 1) (french fries, 1) (mineral water, 1) (soup, 12)
(eggs, 1) (french fries, 2) (green tea, 4) (soup, 12)
{french fries, 5}
(french fries, 1) (milk, 1)
(french fries, 1) (mineral water, 1) (soup, 12)
(french fries, 2) (green tea, 4) (soup, 12)
(french fries, 1) (milk, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
{milk, 5}
(milk, 1) (mineral water, 1) (green tea, 1)
(milk, 1)
(milk, 1) (spaghetti, 4) (soup, 12)
(milk, 1) (mineral water, 3) (spaghetti, 5)
(milk, 1) (green tea, 2) (spaghetti, 4) (soup, 12)
{mineral water, 7}
(mineral water, 1) (green tea, 1)
(mineral water, 3) (spaghetti, 5)
(mineral water, 1) (green tea, 2) (spaghetti, 5)
(mineral water, 1) (soup, 12)
(mineral water, 1) (spaghetti, 4) (soup, 12)
{green tea, 9}
(green tea, 1)
(green tea, 2) (spaghetti, 5)
(green tea, 4) (soup, 12)
(green tea, 2) (spaghetti, 4) (soup, 12)
{spaghetti, 9}
(spaghetti, 5)
(spaghetti, 4) (soup, 12)
{soup, 12}
(soup, 12)
=====

```

(3) SAVE 및 BTLOAD 및 PRINT_BPTREE eggs 2


```

=====SAVE=====
Success
=====

=====BTLOAD=====
Success
=====

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

```

SAVE와 BTLOAD가 성공적으로 이루어졌음을 볼 수 있고, 형식에 맞게 eggs에 대해 frequency가 2이상인 frequent pattern이 PRINT된 것을 볼 수 있다.

```

(4) PRINT_RANGE      green tea      2      3
PRINT_CONFIDENCE     green tea      0.4

```

```

=====PRINT_RANGE=====
FrequentPattern      Frequency
{brownies, green tea}      2
{burgers, green tea}      2
{chocolate, green tea}    2
{milk, green tea}         2
{mineral water, green tea} 2
{pancakes, green tea}     2
{body spray, pancakes, green tea} 2
{body spray, soup, green tea} 2
{burgers, french fries, green tea} 2
{burgers, soup, green tea} 2
{chocolate, spaghetti, green tea} 2
{soup, spaghetti, green tea} 2
{burgers, french fries, soup, green tea} 2
{body spray, green tea}    3
{french fries, green tea}  3
{french fries, soup, green tea} 3
=====
=====PRINT_CONFIDENCE=====
FrequentPattern      Frequency  Confidence
{spaghetti, green tea} 4 0.44
{soup, green tea}      6 0.67
=====
=====EXIT=====
Success
=====

```

green tea에 대해 frequency가 2이상 3이하인 frequent pattern이 잘 출력되는 것을 볼 수 있고, confidence가 0.4이상인 frequent pattern이 소수점 2자리수까지만 표현되어 출력되는 것을 볼 수 있다. EXIT도 성공하였다.

(5) save에 대한 result.txt파일

2	almonds	burgers			
2	almonds	burgers	eggs		
2	almonds	burgers	eggs	soup	
2	almonds	burgers	soup		
2	almonds	eggs			
2	almonds	eggs	soup		
2	almonds	soup			
2	avocado	soup			
2	black tea	mineral water			
2	black tea	mineral water	salmon		
2	black tea	mineral water	salmon	spaghetti	
2	black tea	mineral water	spaghetti		
2	black tea	salmon			
2	black tea	salmon	spaghetti		
2	black tea	spaghetti			
3	body spray	green tea			
2	body spray	green tea	pancakes		
2	body spray	green tea	soup		
2	body spray	pancakes			
2	body spray	soup			
2	brownies	green tea			
2	burgers	eggs			
2	burgers	eggs	soup		
2	burgers	french fries			
2	burgers	french fries	green tea		
2	burgers	french fries	green tea	soup	
2	burgers	french fries	soup		
2	burgers	green tea			
2	burgers	green tea	soup		
3	burgers	soup			
2	chicken	eggs			
2	chicken	eggs	mineral water		
2	chicken	mineral water			
2	chicken	soup			
3	chocolate	eggs			
3	chocolate	eggs	soup		
2	chocolate	green tea			
2	chocolate	green tea	spaghetti		
2	chocolate	ground beef			
2	chocolate	ground beef	soup		
4	chocolate	soup			
2	chocolate	spaghetti			
2	eggs	french fries			
2	eggs	french fries	soup		
2	eggs	mineral water			
4	eggs	soup			
2	eggs	turkey			
2	energy bar	milk			
2	energy bar	milk	mineral water		
2	energy bar	mineral water			
2	escalope	frozen smoothie			
2	escalope	frozen smoothie	mineral water		
2	escalope	frozen smoothie	mineral water	spaghetti	
2	escalope	frozen smoothie	spaghetti		
2	escalope	mineral water			
2	escalope	mineral water	spaghetti		
2	escalope	spaghetti			
3	french fries	green tea			
3	french fries	green tea	soup		
2	french fries	milk			
4	french fries	soup			
2	fresh tuna	mineral water			
2	fresh tuna	mineral water	spaghetti		
2	fresh tuna	spaghetti			
2	frozen smoothie	mineral water			
2	frozen smoothie	mineral water	spaghetti		
2	frozen smoothie	spaghetti			

```

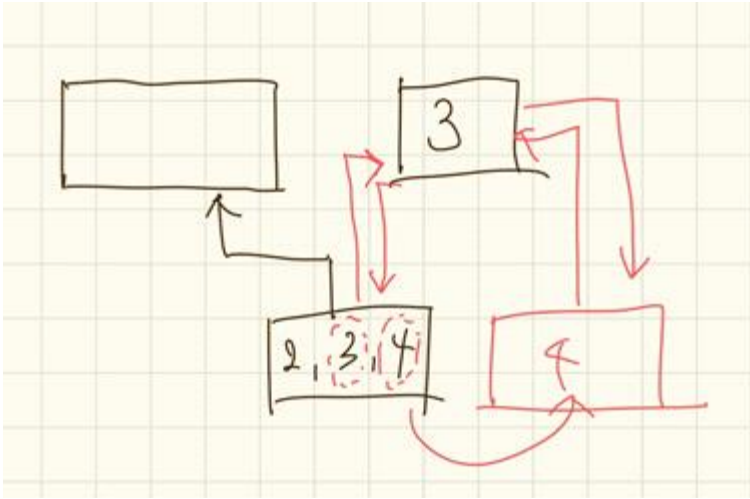
2 frozen smoothie mineral water
2 frozen smoothie mineral water spaghetti
2 frozen smoothie spaghetti
2 frozen vegetables soup
2 frozen vegetables soup spaghetti
2 frozen vegetables spaghetti
2 grated cheese white wine
2 green tea milk
2 green tea mineral water
2 green tea pancakes
6 green tea soup
2 green tea soup spaghetti
4 green tea spaghetti
2 ground beef mineral water
2 ground beef mineral water spaghetti
2 ground beef soup
3 ground beef spaghetti
2 hot dogs soup
2 milk mineral water
2 milk soup
2 milk soup spaghetti
3 milk spaghetti
2 mineral water salmon
2 mineral water salmon spaghetti
2 mineral water soup
5 mineral water spaghetti
2 mineral water whole wheat rice
2 pasta shrimp
2 protein bar soup
2 salmon spaghetti
4 soup spaghetti
2 spaghetti white wine

```

리눅스 환경에서는 delay가 생겨서 끝까지 출력하는데 시간이 좀 걸리지만 끝까지 잘 출력되는 것을 볼 수 있다.

5. Consideration

map, set, multimap, list, vecotr 등 stl에 대한 정의 및 활용은 처음이라서 맨처음에 많이 헷갈렸다. 특히 pair형을 요소로 갖는 map의 자료구조는 맨처음에 잘못 이해해서 아예 코드를 잘못 짜기도 했다. map 이나 set의 경우 중복을 제거하여 insert를 해주기 때문에 같은 줄에 같은 상품이 있는 transaction을 읽을 때 등 중복을 없애줄 때 사용하면 유용했다. multimap의 경우도 자동으로 key값에 따라 sorting을 해주는 기능이 있어서 frequency가 큰 순서대로 transaction이 정돈되어 fp-tree를 구현할 때 유용했다. b+-tree를 구현할 때, index노드를 split해주는 경우 split된 기존 노드의 부모를 새로 만들어진 부모노드를 가리키게 하지않았더니 split자체는 잘 되는데 그 다음에 새로 frequency를 넣을 때 자꾸 다른 곳에 삽입되는 것을 확인했다. 따라서 다음 그림과 같이 빨간 부분을 추가해주었다.(order = 3으로 가정)



testcase1으로 했을 때는 잘 돌아갔던 코드들이 testcase2로 경우의 수가 많아지자마자 자잘 자잘 한 곳에서 오류가 많이 나서 힘들었지만 덕분에 stl이나 iterator를 이해할 수 있었던 시간이었다.