隊名：ㄎㄧㄜˊ

(1) the team members' names and school IDs

林良翰 B03902089

楊力權 B03902101

宋吟軒 B03902070

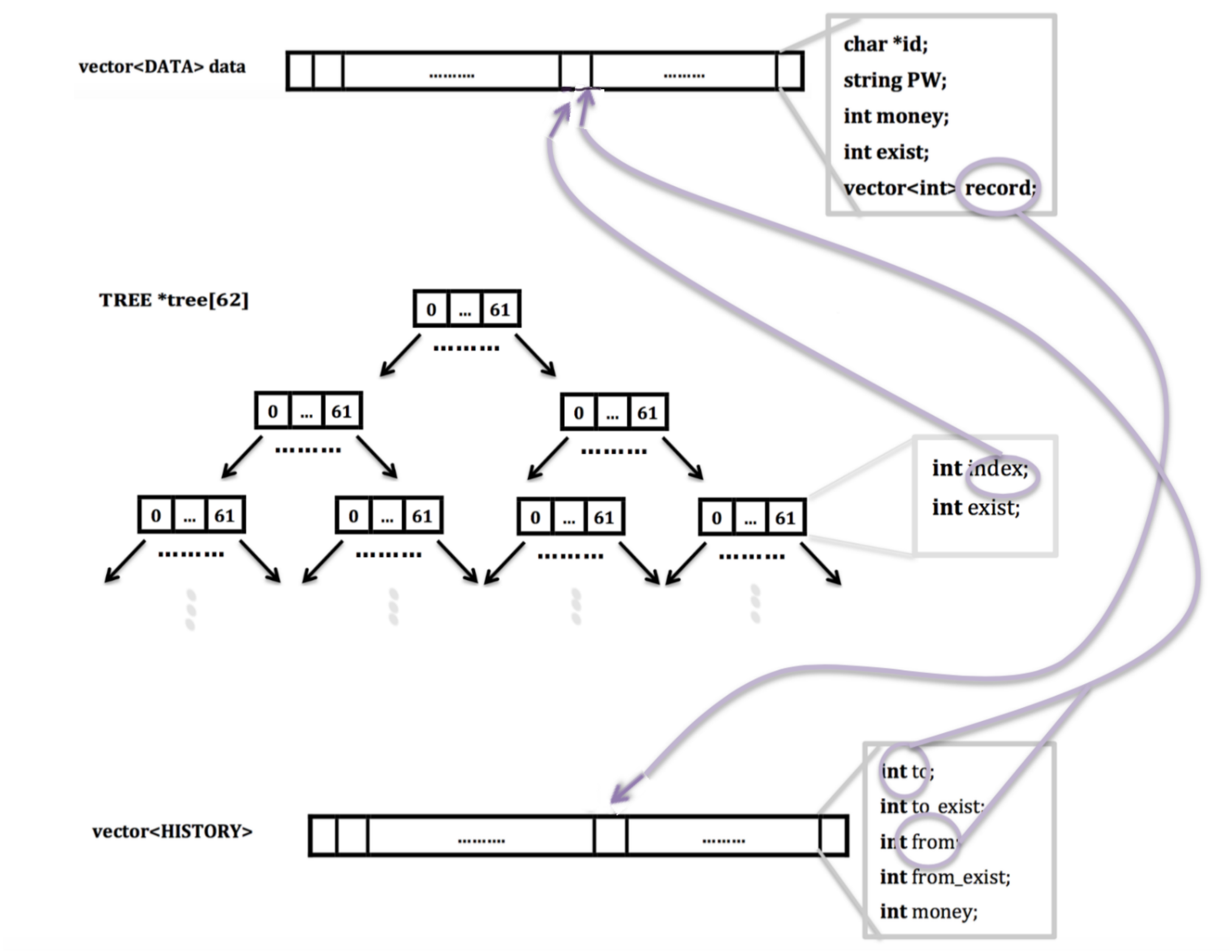
(2) how you divide the responsibilities of the team members

items				林良翰	楊力權	宋吟軒
data structures	tree + vector	data structure	dictinary tree	O		
			vector	O	O	O
		algorithm	account advise (create)		O	
			account advise (transfer)	O		
			find account by recursion	O		
			find account by regex		O	
			search history			O
		password hashing(md5)			O	O
		main program		O		
		account managing (create, delete, merge)		O		
		transfer history managing (transfer, merge)		O		O
		money managing (deposit, withdraw, transfer)				O
	tree only			O		
	vector only				O	
final report				O		O

(3) the data structures you compared, including the results submitted to the mini-competition site

data structure	說明	速度				
		competition (cmds / 5s)	self test data			
			62 ids, no find, 200K cmds		find only, 16K cmds	
			s	(cmds / s)	m, s	(cmds / s)
vector only	所有id皆存在vector	185	3.033	65941	6m 3.566	44
tree only	所有id皆存在tree	29910	5.304	37707	43.044	371
tree + vector	如第四小題所述	106230	3.697	54098	42.168	379

如下圖：我們的字典樹的每個***node***有數字***0~9***、***a~z***、***A~Z***



（4）**the data structure you recommend**

我們推薦的資料結構是：tree + vector

首先用一個vector存所有帳號的資料，並用一個dictionary tree存每個使用者的index來輔助此vector，因此存取vector所需的時間為O(1)。再用另一個vector存所有交易紀錄，並且每一個帳號都有自己交易紀錄的index，因此所有的交易紀錄都會按照時間順序排列。總之，這樣的資料結構都汲取了dictionary tree和vector的優點，dictionary tree大幅提升了搜尋速度，而vector則彌補了dictionary tree極端資料下的問題，且完善的解決了交易紀錄時間順序的問題。

function	vector only	tree only	vector + tree
create, delete	O(1)	O(log*n*)	O(log*n*)
account advise(create)	O(n)	O(log*n*)	O(log*n*)
account advise(transfer)	O(n)	O(n)	O(n)
find account	O(n)	O(log*n*)	O(log*n*)

（5）the advantages of the recommendation

a. 歷史紀錄依照時間順序排列

b. 存取使用者所需時間時間為O(1)

c. 對於較短的帳號，字典樹不會佔太多空間

d. find, create可以用recursion找

（6）the disadvantages of the recommendation

a. 帳號數量少時，transfer不可以用recursion找

b. 帳號名稱若長，則會佔較多空間

c. 遞迴不太好寫，容易 segmentation fault

d. search前必須把record內的index重新排序（使用quick sort）

（7）how to compile your code and use the system（makefile g++）

a. How to compile?

```
==============================================================
CPPFLAG = -std=c++11 -O3
all:
	g++ $(CPPFLAG) main.cpp all.cpp md5.cpp advise.cpp -o final_project

run:
	g++ $(CPPFLAG) main.cpp all.cpp md5.cpp advise.cpp -o final_project
	./final_project

clean:
	rm -rf final_project
	rm -rf maker
	rm -rf test.out
	rm -rf test.in

maker:
	g++ $(CPPFLAG) maker.cpp -o maker
	./maker 200000 > test.in

test:
	g++ $(CPPFLAG) main.cpp all.cpp md5.cpp advise.cpp -o final_project
	./final_project < test.in > test.out
==============================================================
```

b. How to use the system?

command	parameters	instruction
login	[id] [pw]	login id
create	[id] [pw]	create new id
delete	[id] [pw]	delete exist id
deposit	[money]	deposit money into current account
withdraw	[money]	withdraw money from current account
transfer	[id] [money]	transfer money to id from current account
merge	[id1] [pw1] [id2] [pw2]	merge id2 to id1
find	[id]	find id, * and ? for advance functions
search	[id]	search the transfer history between current account and id
account		list all exist, been deleted, been merged accounts
history		list all transfer history by timeline
help		instructions for user

(8) the bonus features you implement and why you think they deserve the bonus

a. Test Data Maker：產生create、delete、log in、deposit、withdraw 、transfer、merge、search的測資，並自由決定測資的大小、帳戶數量與金錢上限，以利本組以及其他組同學debug。（command line：**./maker [size] > [name].in**）

b. 可以印出**所有人**交易的歷史紀錄，而且按照**時間順序**排列。（input line：**history**）

c. Print all existed, been deleted, or been merged accounts.（input line：**account**）

d. Instructions for user.（input line：**help**）