1. Good Evening
2. Let's begin at 9:10 pm
3. Topic - Splitwise Algorithm

Agenda

1. Settlement algorithm for splitwise → 2
2. Undo algorithm for TTT → 3

Next class

1. Command Design Pattern] 1
2. Restful API] 1
3. One data flow end to end
4. O(1) WS from TTT] 20

Settlement Algo

✓ 1. Pull Reqd. data ✓ → EPRepo
ESRepo

✓ 2. Transform ✓→ Make balance HM

✓ 3. Settlement. Algo < Round trip
Greedy

Users → A, B, C, D, E, F ✓

Groups → Hostel, Office ✓

→ Hostel : A^{*}, $B^{*\$}$, C, D ✓ →

→ Office : A, B, E^{*}, $F^{*\$}$

User

id	name	phn	pwd
1	A		
2	B		
3	C		
4	D		
5	E		
6	F		

Group

id	name	cb
1	Hostel	2
2	Ofc	6

Group Participants

id	gid	uid	isAdmin
1	1	1	✓
2	1	2	✓
3	1	3	✗
4	1	4	✗
5	2	1	✗
6	2	2	✗
7	2	5	✓
8	2	6	✓

Hostel

✓ → Dinner ✓ 19/2/2023

Amount : 1000 ✓

PB : A (500) ✓, B (500) ✓

SB : A(300) ✓, B(300) ✓, C (200) ✓, D(200) ✓

✓ → New Year ✓ 01/01/2023

Amount : 2000 ✓

PB : D (2000)

SB : A (500), B (500), C(500), D(500)

PB = B(3000), E (2000), F (1000)

Ofc

✓ Trip ✓ 10/1/2023

Amount : 4000 ✓

PB : B(2000), E(2000) ✓

SB : A(1000), B(1000), E(1000), F(1000)

✓ Farwell ✓ 31/1/2023

Amount : 2000 ✓

PB : B(1000), F(1000)

SB : A (500), B(500), E(500), F(500)

✓ Booze 2/2/2023

Amount : 3000

PB : A(1500), B(1500)

SB : A(1000), B(1000), C(1000)

Expense

id	title	amount	date
1 ✓	Dinner	1000	
2 ✓	New Year	2000	
3	~~Trip~~ →	4000	
4	~~Farwell~~ →	2000	
✓ 5	Booze	3000	

Group Expense

id	gid	eid
1	1	1
2	1	2
3	2	3
4	2	4

EP ✓

id	uid	eid	amount
1	1	1	500
2	2	1	500
3	4	2	2000
4	2 B	2	2000 ✓
5	5 E	3	2000 ✓
6	6 F	4	1000 ✓
7	1	5	1500
8	2	5	1500
9.	2 B	4	1000

ES ✓

id	uid	eid	amount
1	1	1	300
2	2	1	200
3	3	1	200
4	4	1	200
5	1	2	500
6	2	2	500
7	3	2	500
8	4	2	500
9	1	3	1000
10	2	3	1000
11	5	3	1000
12	6	3	1000
13	1	4	500
14	2	4	500
15	5	4	500
16	6	4	500
17	1	5	1000
18	2	5	1000
19	3	5	1000

Ofc →

PB = { B (2000), E(2000), F(1000)
B(1000) }

B (3000), E (2000), F (1000)

```
Select   ep.usrid, SUM(ep.amount)
FROM     Group g
JOIN     GroupExpense ge   ON  g.gid =
                               ge.gid
JOIN     Expense  e   on ge.eid = e.eid
JOIN     ExpensePaidBy ep  ON  e.eid = ep.eid
GROUP BY   ep.usrid
Where   g.gid = 2
```

Ofc

PB = B(+3000), E(2000), F(1000)

SB = A(-1500), B(1500), E(1500), F(1500)

A → +0 − 1500~~0~~ = −1500

B → +3000 − 1500 = +1500

E → +2000 − 1500 = +500

F → +1000 − 1500 = −500

A = - 1500
B = + 1500
E = + 500
F = - 500

Settle a group → GC → GS

L<EP> EPR

ESR L<ES>

1. ExpensePaidByRepository → Retreive List of ExpensePaidBy using group id

2. ExpenseSharedBy Repo → Retreive List of ExpenseSharedBy using group id

3. Make balance hashmap

List<EP> ✓

B	Trip	+ 2000	✓
E	Trip	+ 2000	✓
F.	FW	+ 1000	✓
B	FW	+ 1000	✓

List <ES>

A	Trip	+ 1000	✓
B	"	+ 1000	✓
E	"	+ 1000	✓
F	"	+ 1000	✓
A	FW	+ 500	✓
B	"	+ 500	✓
E	"	+ 500	✓
F	"	+ 500	✓

B' → +2000 +1000 -1000 - 500

Ɛ → +2000 - 1000 - 500

F → +1000 - 1000 - 500

A → -1000 - 500

B → + 1500

Ɛ → + 500

F → - 500

A → - 1500

+ → You paid more than your real share

- → You paid less than your real share

1. Getting data
2. Making Bal Hashmap
3. Settle.

A → +200

B → +500

C → -300

D → -200

E → -100

F → -100

A → ~~+ 200~~

B → ~~+100~~

C → ~~-200~~ 0

D → ~~-100~~

1. $C \xrightarrow{200} A$.

2. $D \xrightarrow{100} B$

+ → Paid extra

− → Paid Less

Payments should be from -ve to +ve

For a transaction add to payer & subtract from receiver.

Algorithm 1 (Round trip)

A → +200
B → +100
C → -200
D → -100

Guarantee → For n users it will always produce n-1 transactions.

A^{+200} B^{+100} C^{-200} D^{-100}

1. $B \xrightarrow{200} A$

$A^{+200-200}$ $B^{+100+200}$ C^{-200} D^{-100}

A^{0} B^{+300} C^{-200} D^{-100}

2. $C \xrightarrow{300} B$

A^{0} $B^{+300-300}$ $C^{-200+300}$ D^{-100}

A^{0} B^{0} C^{+100} D^{-100}

3. $D \xrightarrow{100} C$

A^{0} B^{0} $C^{\overset{0}{\cancel{+100-100}}}$ $D^{\overset{0}{\cancel{-100+100}}}$

A → +200
B → +500
C → -100
D → -300
E → -200
F → -100

5 transactions

+ Receive
− Pay

A^{200}	B^{500}	C^{-100}	D^{-300}	E^{-200}	F^{-100}
-200	+200				
0	+700	-100	-300	-200	-100
✓	-700	⋮			
0	0	600	-300	-200	-100
		0	300	-200	-100
			0	+100	-100
				0	0

1. $B \xrightarrow{200} A$
2. $C \xrightarrow{700} B$
3. $D \xrightarrow{600} C$
4. $E \xrightarrow{300} D$
5. $F \xrightarrow{100} E$

n users → n-1 transactions

O(n)

A → +~~200~~ ✓ 0

B → +~~100~~

C → -~~100~~

D → -~~200~~ 0

$D \xrightarrow{200} A$

$C \xrightarrow{100} B$

A → -100 + 100 = 0

B → -200 -100 = -300 + 300 = 0

C → +100 - 300 = -200 + 200 = 0

D → +200 - 200 = 0

1. $A \xrightarrow{100} B$ ☆
2. $B \xrightarrow{300} C$
3. $C \xrightarrow{200} D$

A -100
B -200
C +100
D +200

A,B has paid less

C,D has paid more

$A, B \longrightarrow C, D$

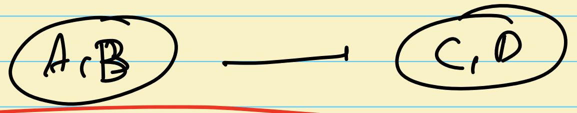
Less payers More payers

Round trip

1. Guaranteed n-1 transactions.

2. Problem = Not smart

↓



Break = 10:24 to 10:34

✓ A → +500

✓ B → -200

✓ C → +600]

✓ D → -400

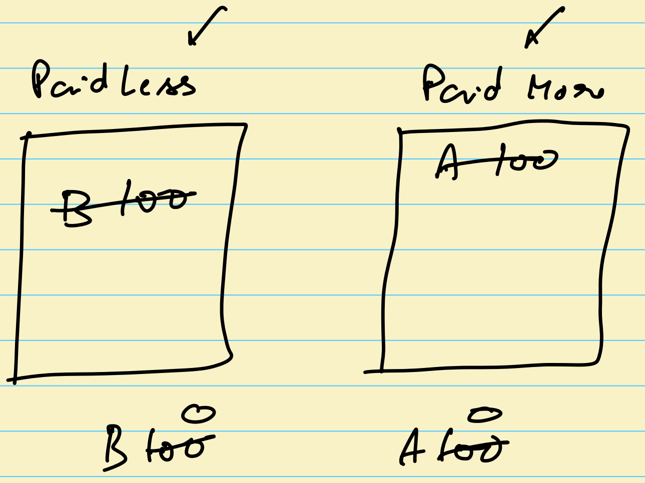
✓ E → -300

✓ F → -500

✓ G → +400

✓ H → -400

✓ I → +300



1. F $\xrightarrow{500}$ C

2. H $\xrightarrow{400}$ A

3. D $\xrightarrow{400}$ G

4. E $\xrightarrow{300}$ I

5. B $\xrightarrow{100}$ C

6. B $\xrightarrow{100}$ A

```
1. Add -ves to paid Less PQ  [take abs of value]

2. Add +ves to paid more PQ

3. While PL PQ & PM PQ are not
   empty.
     a) Remove ~~a~~ x from pl PQ
     b)   "    y   "  pm PQ
     c)  Amount = Min (x.val, y.val)
     d)  Trans = $x \xrightarrow{amount} y$
     e.  Reduce x.val, y.val by amount
     f.  If x.val != 0 add back to
         paid Less PQ
     g.  If y.val != 0 add back to
         paid more PQ.
```

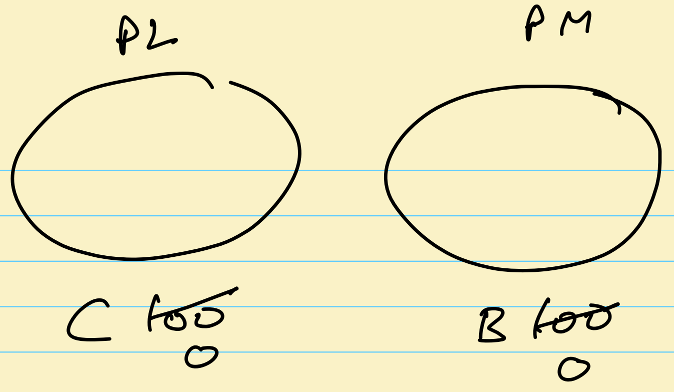
example

A -200

B +100

C -100

D +200



1. A $\xrightarrow{200}$ D
2. C $\xrightarrow{100}$ B

Round trip ✓

1. n-1 transactions
2. O(n) algo
3. Non-smart

Greedy Algo ✓

1. < n-1 transactions
2. nlogn algo.
3. Smart { All trans are from PL to pm side }

✓ Does the greedy algo always give minimum transactions?

Not necessarily.

The algorithm that guarantees minimum transaction is NP hard [2^n time complexity]

1. Round trip ✓ → $O(n)$ [n-1 transactions]
2. ✓ Greddy Algo ✓ → $n \log n$ [smart enough but no guarantees]
3. Backtracking Algo → Guarantees minimum transaction [2^n]

NP Hard

$n = 35$

$$2^{35} = 10^9 \cdot 2^6 = 10^9 \cdot 64$$

Next class

1. Command Design Pattern
2. Restful API
3. End to End Data Flow
4. TTT Discussion.

2.5 hours.