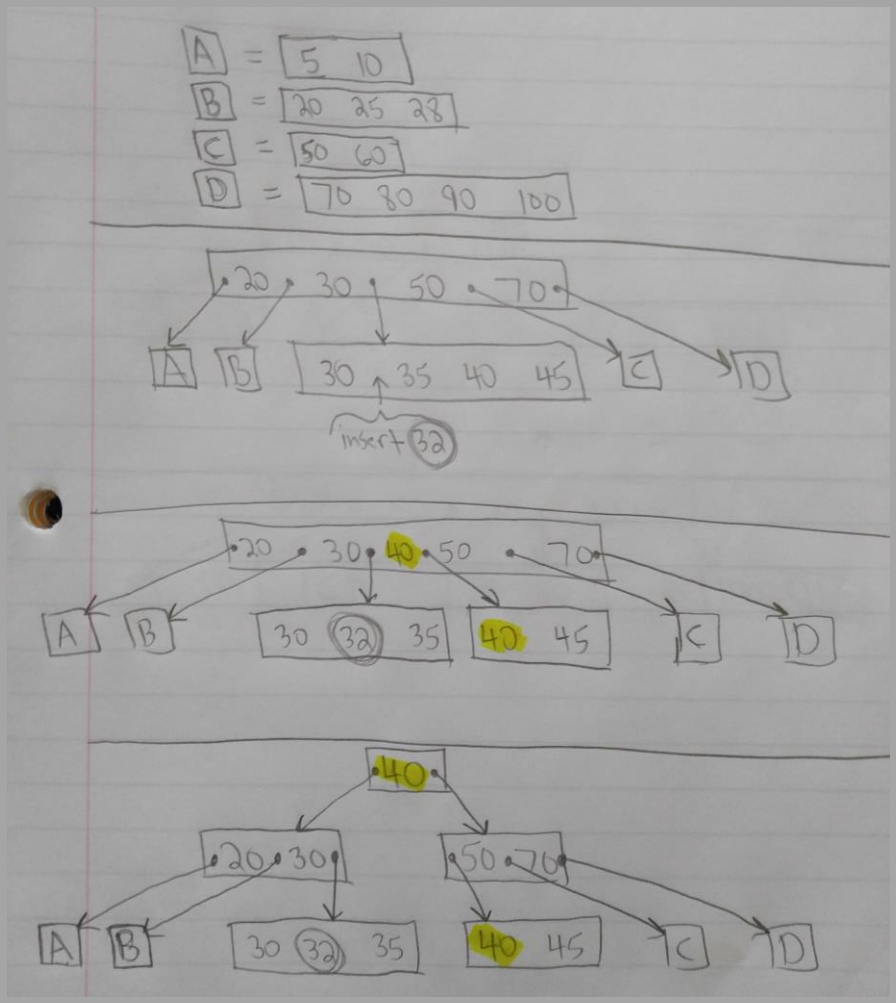
Tom Hills
CS565: Databases and Scripting Languages
Homework 5

1)

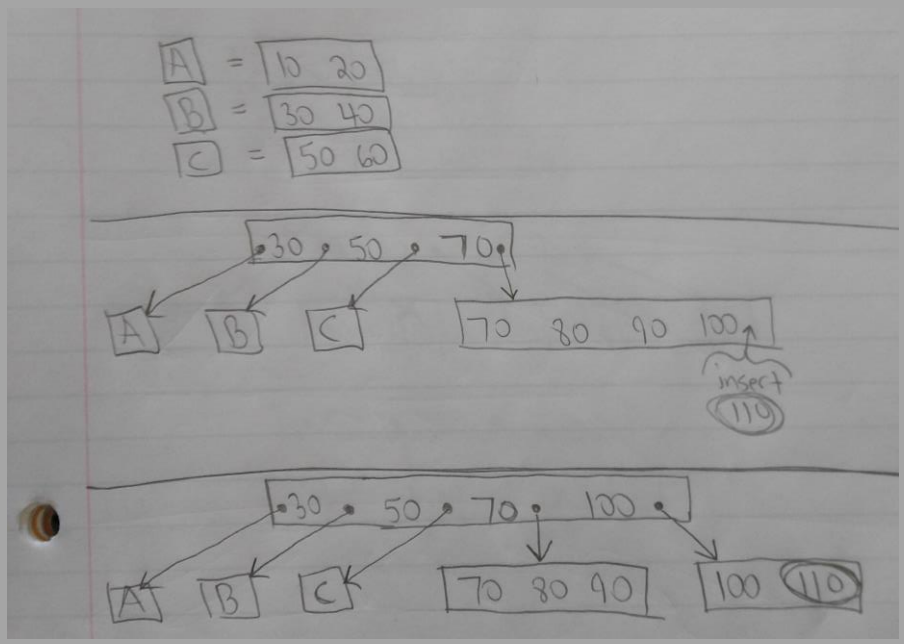
- Hotel (hotelNo, hotelName, city)
  - hotelNo: **Secondary index** because it will frequently be used in joins.
  - HotelName: **Secondary index** because it will frequently be used in joins (e.g., “show me all guests staying at Days Inn.”)
  - City: **Secondary index** because it will frequently be used directly in queries (e.g., “show me all hotels in Nashville.”)
- Room (roomNo, hotelNo, type, price)
  - roomNo: **Secondary index** because it will frequently be used in joins.
  - hotelNo: **Secondary index** because it will frequently be used in joins.
  - type: **Secondary index** because it will frequently be involved in ORDER BY queries (e.g., show me all rooms of a hotel ordered by room type.)
  - price: **Clustering index** because this attribute will most likely be used in range queries (e.g., show all rooms with price < $50.00 per night).
- Booking (hotelNo. guestNo, dateFrom, dateTo, roomNo)
  - hotelNo: **Secondary index** because it will frequently be used in joins.
  - guestNo: **Secondary index** because it will frequently be used in joins.
  - dateFrom: **Clustering index** because this attribute will most likely be used in range queries (e.g., show all rooms with dateFrom <= CURDATE()).
  - dateTo: **Clustering index** because this attribute will most likely be used in range queries (e.g., show all rooms with dateTo > CURDATE()).
  - roomNo: **Secondary index** because it will frequently be used in joins.
- Guest (guestNo, guestName, guestAddress)
  - guestNo: **Secondary index** because it will frequently be used in joins.
  - guestName: **Secondary index** because guests will almost always be queried by name, not guestNo.
  - guestAddress: **No index** since this attribute will not be used much in queries.

2)

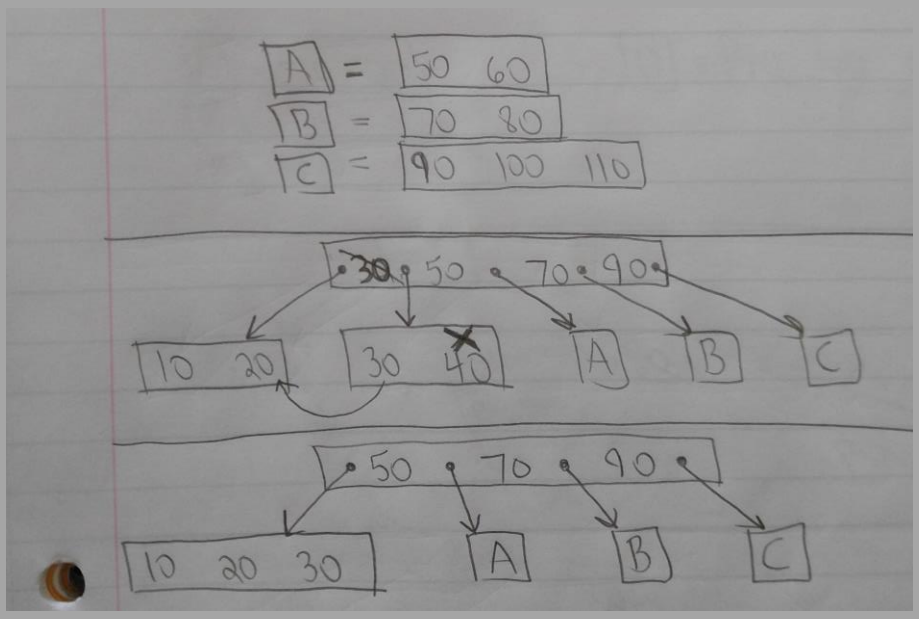
a)


A = 5 10
B = 20 25 28
C = 50 60
D = 70 80 90 100
20 30 50 70
A B 30 35 40 45 C D
insert 32
20 30 40 50 70
A B 30 32 35 40 45 C D
40
20 30 50 70
A B 30 32 35 40 45 C D

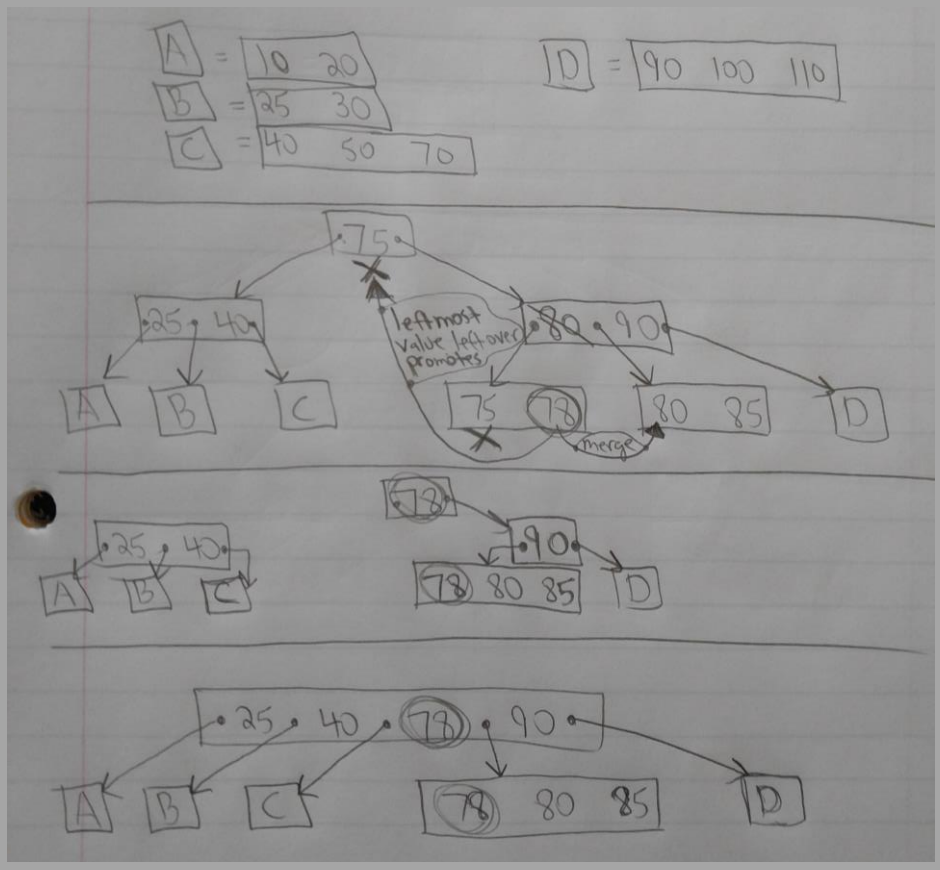
b)


A = 10 20
B = 30 40
C = 50 60
30 50 70
A B C 70 80 90 100
insert 110
30 50 70 100
A B C 70 80 90 100 110

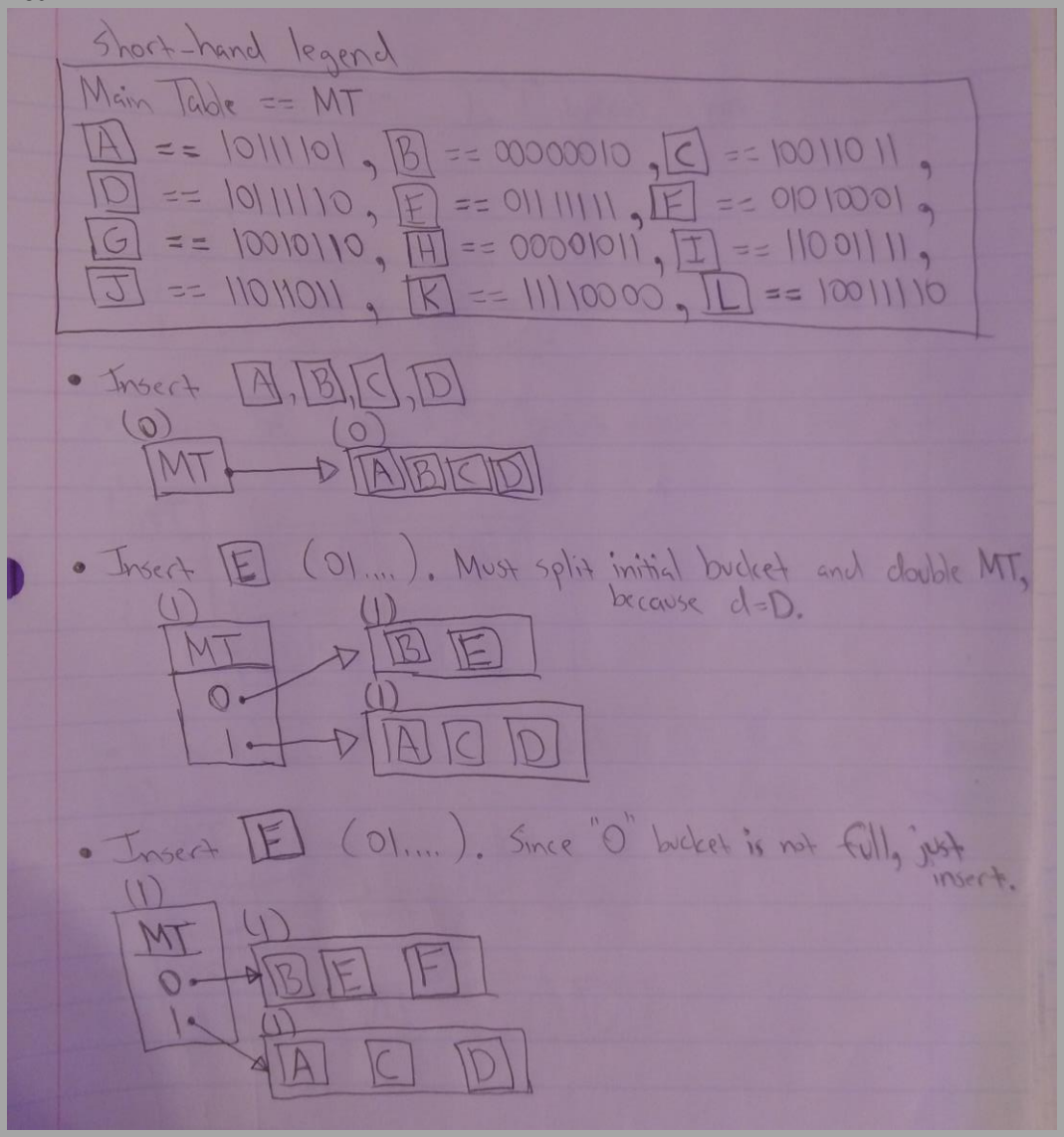
c)


A = 50 60
B = 70 80
C = 90 100 110
30 50 70 90
10 20
30 40
A
B
C
50 70 90
10 20 30
A
B
C

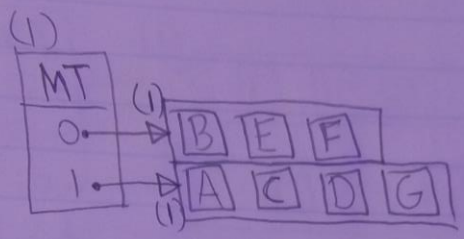
d)


A = 10 20
B = 25 30
C = 40 50 70
D = 90 100 110
75
25 40
A
B
C
leftmost value leftover promotes
80 90
75 78
80 85
D
merge
78
90
25 40
A
B
C
78 80 85
D
25 40 78 90
A
B
C
78 80 85
D

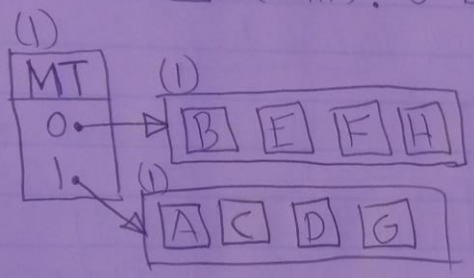
3) Asd


Short-hand legend
Main Table == MT
A == 10111101, B == 00000010, C == 10011011,
D == 10111110, E == 01111111, F == 01010001,
G == 10010110, H == 00001011, I == 11001111,
J == 11011011, K == 11110000, L == 10011110
Insert A, B, C, D
(0)
MT
(0)
A B C D
Insert E (01....). Must split initial bucket and double MT, because d=D.
(1)
MT
0
1
(1)
B E
(1)
A C D
Insert F (01....). Since "0" bucket is not full, just insert.
(1)
MT
0
1
(1)
B E F
(1)
A C D

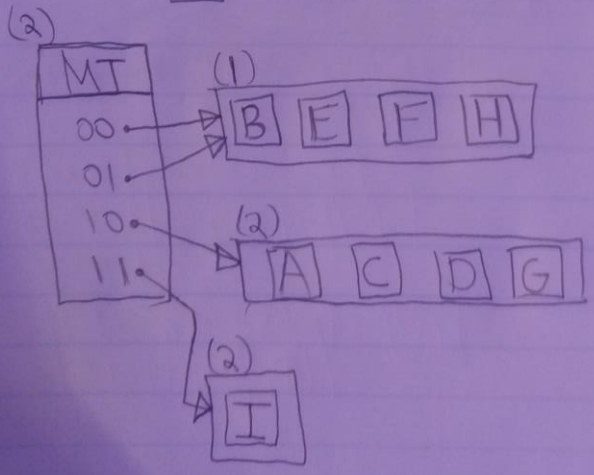
- Insert G (10....). "1" bucket is not full. Just insert.



- Insert H (00...). "0" bucket is not full. Just insert.



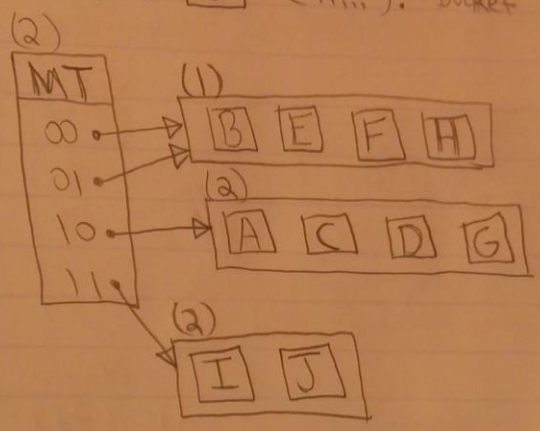
- Insert I (11...). "1" bucket is full. Must split "1" bucket and double MT because d = D.



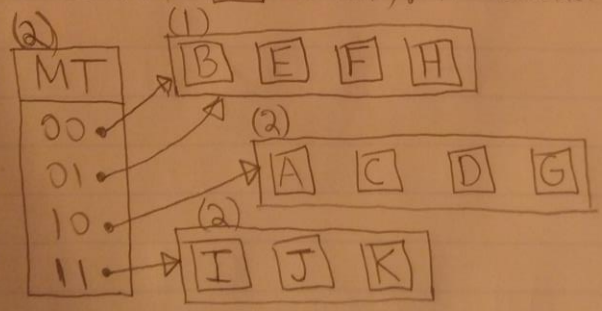
Main Table == MT

A == 10111101, B == 00000010, C == 10011011,
D == 10111110, F == 01111111, E == 01010001,
G == 10010110, H == 00001011, I == 11001111,
J == 11011011, K == 11110000, L == 10011110

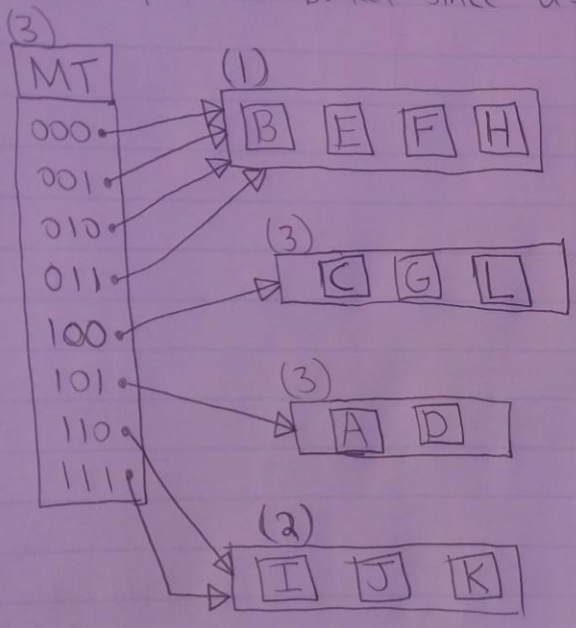
- Insert J (11...). Bucket "11" is not full. Just insert.

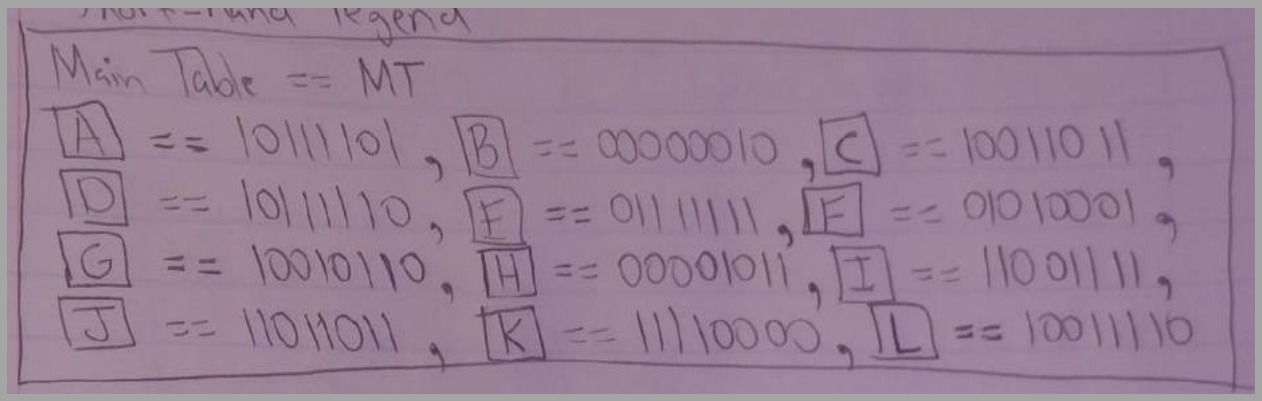


- Insert K (11....). "11" bucket is not full. Just insert.



- Insert L (100.....). Bucket "10" is full. Must double MT and split "10" bucket since d=D.




Main Table == MT
A == 10111101, B == 00000010, C == 10011011,
D == 10111110, F == 01111111, E == 01010001,
G == 10010110, H == 00001011, I == 11001111,
J == 11011011, K == 11110000, L == 10011110