Assignment 3

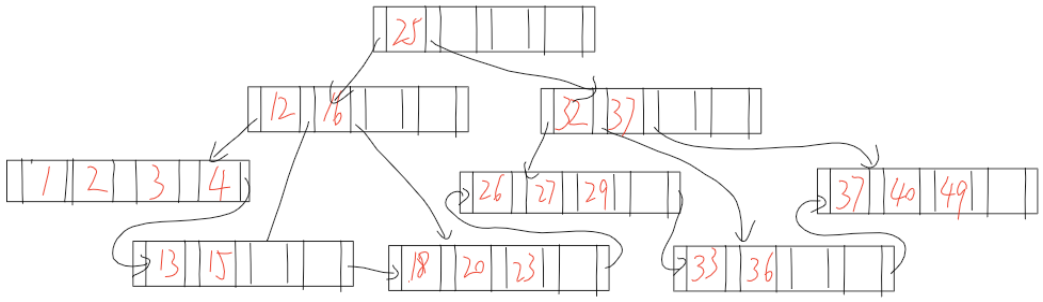
Name: Yu Feng ID: z5094935

Q1.

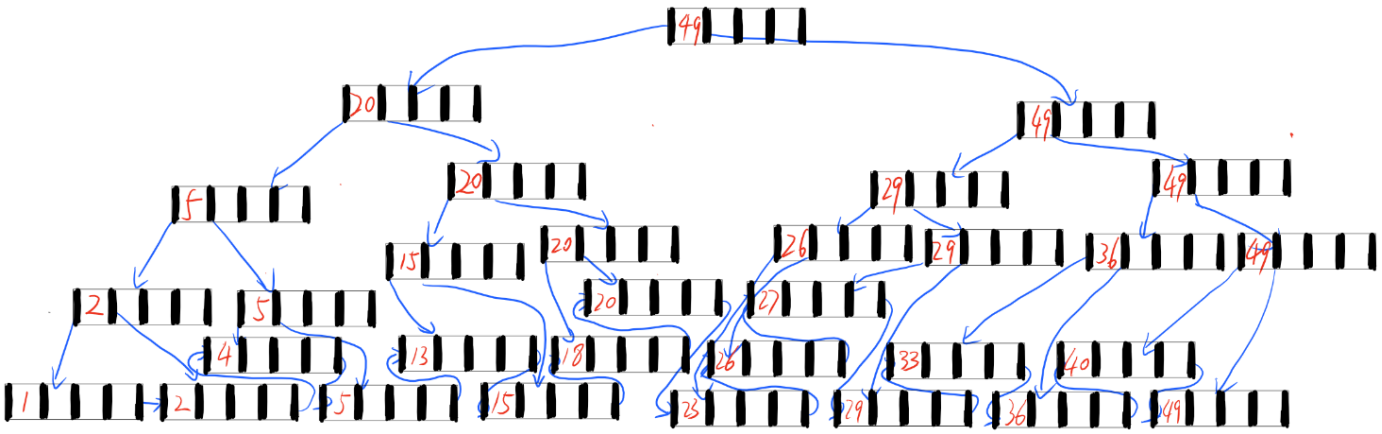
1) Assume that the index is sparse. For all data (the numbers of them is 100), we have 10 index data. In the worst case, sorted index should search 10+10 = 20 times. And binary search takes $\log_2(100) = 30.1$ times. So the sorted index is better.
2) Yes, it is possible. For example, the global depth is 3 and directory is array of size 4. Only "000" and "100" have pointed to data pages, the data are "000, 000, 000, 000" and "100". When deleting "100", the extendible hashing became '0' → ' 0, 0, 0, 0'. It's depth became 1.
3) Assume the threshold is 1.7. We have 4 buckets and 5 data. One of bucket is full. We inset a data in this bucket. It's overflowed. Using the 1st method, the system should split the bucket, but as for 2rd method, it can just add it in the overflow pages(because less than 1.7).

Q2.

1)



2)



Q3.

Opinion 3 "*Use a linear hashed index on attribute R.a.*" is better. Because it search a range of data. It will jump to every bucket if used hashed index, which cost a lot of I/O. But for the sorted index it just need to get the first value that more than 60,000 and follow it. Because the number of data is large, using B+ tree is a better way. It just use the value of R.a, so the index of R.b is useless.

Q4.

1) The database could roll back Trans3 using old data value and redo the changes made by Trans2.
2) Just undo Trans3 and redo Trans 2...
3) **Yes**
4) **It does not exist**. For Transaction1 it just use X, so it cannot have deadlock. For T2 and T3, both R2(Y) and R2(X) are started before R3(X) and R3(Y). it cannot become a deadlock.

Q5.