固定大小的array of array

概念:

利用多個固定size (k)的排序array來存資料

(陣列與陣列間不需排序)

Insert

插入當前陣列

如果陣列是滿的就再開一個新陣列

O(n/k)

Search

對每個陣列做 binary search

O(n/k \* log(k))

分析 可以經由調整k值讓insert 與 search時間trade off

要最平衡可以選擇 k=根號(n)

則 insert = (根號n) search=(根號n \* logn)

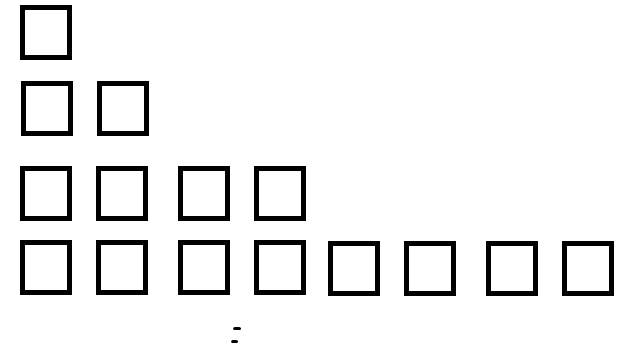
注意:當k=1，就是普通array

當k=n 就是排序好的array

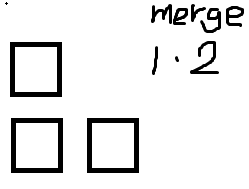
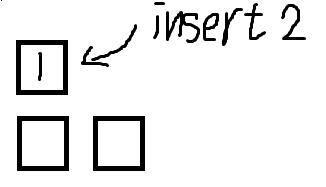
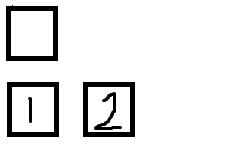
動態配置的array of array

借鑑 dynamic array的概念

每次array滿了就再開兩倍大的array



但是插入資料發現空間不夠時，會先merge該層資料，然後試著往下搬移

大概演算法為 insert (data,layer){

if(layer > 最大層數){

創建上一層兩倍大的新array

放入新array

}

else If(layer is full){

data = merge(layer.data , data)

Insert(data,layer+1);

}

else {

放入新array

}

}

原因: 希望搬動到大array機會越少越好

Fact: \*第一層(只有一格)array會有n/2時間可直接插入

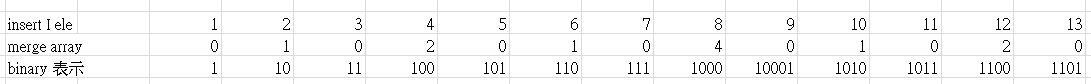
\*最後一層長度是資料的O(n/2)

\*每一層不是全滿就是全空

Search

因為array共會有logn個，所以search 為(logn\*logn) (對每個做binary search)

Insert:



注意: 可以用二進位表示此元素插入後，總體的狀態，1代表全滿、0代表空

發生進位時則merge兩個長度為2^(m-1)的array

另一角度

每2^(m+1)次，merge兩個長度2^m的array

所以 sigma(2^ M \* n/2^(M+1)) = O(n logn)

(最大一項可看為 N ，也就是每N項merge 長度為 O(N)的陣列 )

因此 為 N\*N\*1/N = N 然後有logN項

由於有B tree等更好的資料結構，其實實務上很少用到