1. 0 ，沒有問問題
2. 插入資料時較容易。如果要在一組資料’a’, ’c’, ‘d’ ,’e’ ,‘f’ 中加入’b’使其符合abcdef的順序，dynamic array 必須先找到一個長度為6的位置，然後去把資料copy 過去，相比之下，single linked list 只要改變指針就行
3. 假設資料……, ‘d’ ,’e’ ,‘f’,……. 要刪去 ’e’ 變成 ‘d’, ‘f’在single linked list 的情況下，要找到 「指向e的」和「e指向的」然後連接起來，找到前者只能用線性搜索，花費Ｏ（ｎ）；double linked list 也要找到 「指向e的」和「e指向的」然後做一樣的事，但可以很容易的找到它們（因為是雙向的）。
4. 新增（和刪除)資料快速，sorted dynamic array 要花O(n)，binary search tree只要花O(logn)
5. 新增（和刪除)資料快速，和bst一樣，期望的時間複雜度為O(logn)
6. 當資料是被排序好的時候，搜尋時skip list 具有更好的效率，並且用更少的空間（<https://zh.wikipedia.org/wiki/%E8%B7%B3%E8%B7%83%E5%88%97%E8%A1%A8>

）

1. 與6相反，如果資料是亂序的，hashtable 能有更好的效率
2. 假設M=100 那麼資料1,2,3,…,1000就只會對應到0,10,20,30,…,90，換言之就是極容易發生「碰撞」
3. 想法：在這個stack的最後儲存總和

在stack是空的時候，push 0，

當要加入一個資料時，1)先pop，2)然後把pop的結果跟要加入的資料x相加，3)然後push x, push 2)的結果

當要刪除一個資料時，1)先pop，2)然後再pop，3)計算 1)的回傳減去2)的回傳，4) 然後push 3）結果

要有總合只要查看最後一個位置的資料

1. 否，如果要用char array 存字串”0123456789”，必須再加上結尾/0，也就是要有十個字元的空間再加2個單元才放得下
2. x-32 ，ASCII的相應大小寫編碼差32，小寫為x，那大寫就是x-32
3. 2D array視作matrix，要刪掉edge (u,v)，即找到matrix中u,v的位置，然後改它的value改成0，查訪位置即[u][v]為constant time改數字為constant time 因此時間複雜度為O(1)

假設是complete graph，有n個節點，記作k\_n，worst case是要刪掉任一(i,n) i是任意節點，因為每一節點(對應到的list)都有n-1的長度，因此要花費O(n-1）=O(n)去刪除一邊，對於(n,i)則要花 O(i)，總花費O(n)+O(i)=O(n)

1. 我認為亂數之所以可以構築好的演算法是基於其隨機性，隨機性講求其每個數字選到的機率是均等的，均等的資料結構通常較好的效能（hashtable）。再加上在程式中被執行數多次，更滿足了大數法則，使其平均性變得可靠。