Student name : Rakyan Satrya Adhikara

Student ID : 219548135

SIT111 Algorithms and Computer System Exam

Question 16

```
def subsets(L):
    temp_res = []
    subsets_util(L, [0 for i in range(len(L))], temp_res, 0)
    result = []
    for i in temp_res:
        temp = []
        for j in range(len(i)):
            if (i[j] == 1):
                temp.append(L[j])
        result.append(temp)
    return result

def subsets_util(L,temp,result,index):
    if (index == len(L)):
        result.append([i for i in temp])
        return
    temp[index] = 0
    subsets_util(L, temp, result, index + 1)
    temp[index] = 1
    subsets_util(L, temp, result, index + 1)

print(subsets([1,2,3,4]))
```

- From base-7 642 to base 10:
    1. 600 = 6 x 7^3 = 6 x 343 = 2058
    2. 40 = 4 x 7^2 = 4 x 49 = 196
    3. 2 = 2 x 7 = 14
    4. 2058 + 196 + 14 = 2268

Question 17

- Best case and Worst case:
  - Best case: O(n)
    - Reason: if the list is already sorted in the first place
  - Worst case: O(n^2)
    - Reason: If the list is not yet sorted in the first plaace
-
```python
# Python code has been modified into descending order
def bubblesort(alist):
    size = len(alist)
    for i in range(size):
        index = alist[i]
        for j in range(size - 1, i, -1):
            # changed from less than to more than
            if alist[j] > alist[j-1]:
                temp = alist[j-1]
                alist[j-1] = alist[j]
                alist[j] = temp
```
-
```
PS C:\Users\Rakyan\Documents\GitHub\DeakinUni\SIT111 (Algorithms)\MockExam> python .\exam_test.py
before   : [34, 8, 34, 0, 80, 12, 90, 2, 20]
after    : [90, 80, 34, 34, 20, 12, 8, 2, 0]
```

Question 18

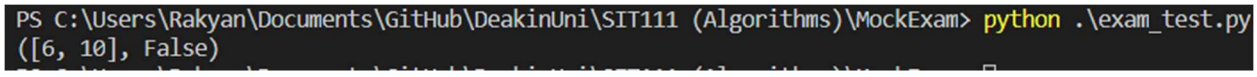
- Binary search: an efficient algorithm for finding a specified element from a sorted list of items. It works by repeatedly dividing in half the size of the list that could contain the item, until the program narrowed down the possible locations to just one.
- Commonly data that can be used for binary search such as an array
-

```
def BinarySearchIterative(item_list, item):
    left = 0
    right = len(item_list) - 1

    while (left <= right):
        mid = (left + right)//2
        if (item_list[mid] == item):
            return mid
        else:
            if (item < item_list[mid]):
                right = mid - 1
            else:
                left = mid + 1

    return -1
```

Question 19

- Graph A is a tree, since in Graph B, the node 2 is connected to 3 and 1, which is not part of a tree
- Home → Page 1 → Page 1.a → Page 1.b → Page 2 → Page 2.a → Page 2.b
- Pros and Cons:
    - For BFT:
        - Pros:
            - Solution will be found if there is a solution
            - BFT will never hit a dead end
            - If BFT found a solution, it will provide the solution with minimal steps
        - Cons:
            - Consumes time and memory much more than DFT
    - For DFT:
        - Pros:
            - Less time and space complexity compared to BFS
            - Solution can be found without much more search
        - Cons:
            - Not guaranteed to get a solution
- 

- Depth-first search can get trapped on infinite branches and never find a solution, even if there is one. It can be avoided by maintaining relatively small memory size by using a combination of DFS and BFS.