Week 6 Submit Task

Part A

For these questions, take the time to practise your exam writing skills.

1. Define backtracking and explain its basic operation. (3 marks)

   Backtracking is a refinment of brute force where when a Permutation has no posibility of a valid solution it prunes said Perm and returns to the Previous dicision made

2. Explain the concept of recursion and its role in backtracking algorithms. (2 marks)

   Recurtion is where a function or algorithm calls itself, gererally making the sample space smaller on each iteration. Back tracking can make use of the properties of recursion by storing previous solutions further up the call tree allowing it to return back when a soluton is deemed invalid

3. What is the purpose of pruning in backtracking algorithms? Provide an example of how pruning can improve the efficiency of a backtracking solution. (2 marks)

   Pruning removes a large amount of Permutations of known bad solutions, that can never be Valid. greatly improving time complexities.

   An example of pruning is in the n-queens Problem, where when the Placed queen attacks a previous one, the entire branch can be disregarded.

4. What are some common applications of backtracking algorithms in computer science or real-world scenarios? (3 marks)

   Dicision making Problems
   TSP
   DNA sequencing
   Cryptography

5. Describe the graph colouring problem and explain how backtracking can be used to solve it. (3 marks)

   Given an undirected graph what is the fewest number, n, of colours needed such that no 2 advacent verticies have the same colour.

   Iterate over all nodes within the given graph, on each iteration attempt to assign each colour. If the assigned colour clashes the algorithm should return and kill the branch

6. Discuss the time complexity of the graph colouring problem when solved using backtracking. (2 marks)

A total of V, verticies can be coloured using at most m colours. m^V = total verticies to colors

$\therefore O(m^V)$

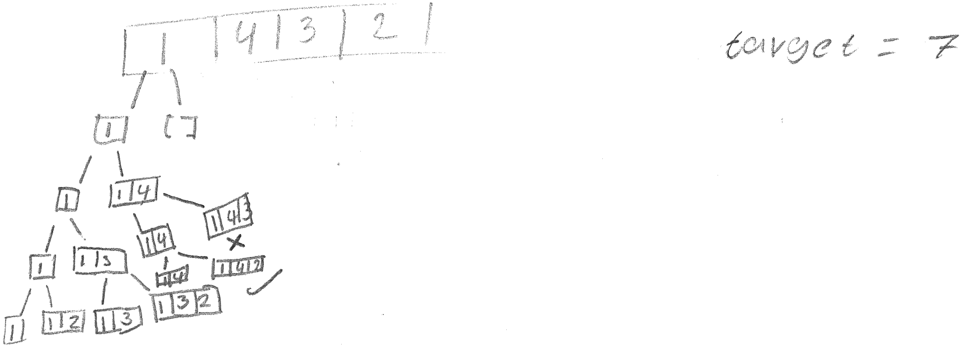
Backtracking only effects avg case

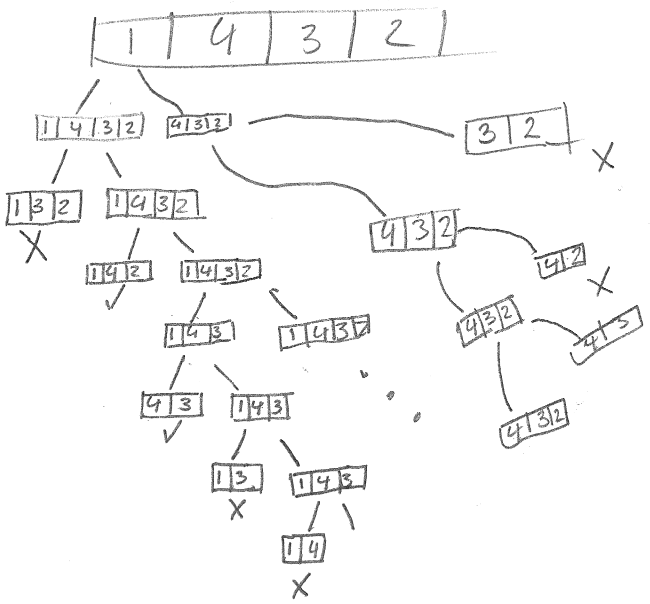
Part B – Optional Extension

The Subset Sum problem involves, for a given set of integers and target sum, determining whether there is a subset of the given set whose elements sum up to the target value. This can be solved with backtracking.

Write Python code based on the following pseudocode structure. It should return the current subset if the target is found, or null if the target cannot be found. Otherwise add the next element from the set to the current subset and call recursively. If the subset is not found, the element should be excluded and the call repeated with the next element.

```
function subsetSum(set, targetSum):
  return backtrack(set, targetSum, [], 0)
function backtrack(set, targetSum, currentSubset, currentIndex):
```


1 4 3 2
target = 7
1
[]
1
1 4
1 4 3
×
1 4
1
1 3
1 4
1 4 2
✓
1
1 2
1 3
1 3 2


1 4 3 2
1 4 3 2
4 3 2
3 2
×
1 3 2
×
1 4 3 2
4 3 2
4 2
×
1 4 2
✓
1 4 3 2
4 3 2
4 5
1 4 3
1 4 3
4 3
✓
1 4 3
4 3 2
1 3
×
1 4 3
1 4
×