Theory of Algorithms Assignment 4

I declare that all material in this assessment task is my work except where there is clear acknowledgement or reference to the work of others. I further declare that I have complied and agreed to the CMU Academic Integrity Policy at the University website.
http://www.coloradomesa.edu/student-services/documents
Submissions that do not include the above academic integrity statements will not be considered.

Student Name: Jesse Holland UID: 700-445-452 Date: November 14, 2020

---

1. Q and A - 10 points

1. Both Divide and Conquer and Dynamic Programming use prior information to come to a solution. For Divide and Conquer, the problem is split into smaller sub-problems and then the solutions for the sub-problems are combined to solve the original problem. For Dynamic Programming, a memoization of previous operations or information is used to prevent the recalculation of the operation or information. This allows for a tedious or long operation to only be calculated once, rather than having to be recalculated each time the operation is needed and allows for information to only need to be gathered once. Both use prior knowledge to work to a solution, but Dynamic Programming takes the knowledge from another sub-programs to apply it to another where the same operations are being done. In this sense, Dynamic Programming uses the overlap of sub-problem to help solve the original problem.

2. Due to the limited number of coins, we can brute force this problem easily. All of the possible choices are listed below:

   (a) $5 + 2 + 6 = 13$
   (b) $5 + 10 = 15$
   (c) $1 + 10 = 11$
   (d) $1 + 6 = 7$
   (e) all other possibilities are contained within one of the above (ex. $2 + 5 + 6 = 13$ is contained within a)

   As item b has the greatest sum, the solution is to pick coins 5 and 10 for a total of 15.

3. If $F(n)$ is the max amount that can be picked up in a row of n coins, then $F(n) = max\{current\_coin + F(n-2), F(n-1)\}$. As the method starts at the nth coin, and recursively calls itself for both n-2 and n-1, it will loop for all of the previous values of n, n times, making it exponential.

4. In the exhaustive search approach, n loops will be needed, as a loop will be needed to loop through all of the remaining coins for **each** coin selected. As such, the time complexity is at least exponential, as this does not take into account for the operations required to check if a coin can be chosen or not.

5. Let $F(n) = [n]$ represent the the row each queen is on in a $n * n$ board.

```
Algorithm : N-Queens
    Input: number of queens
    Result: Array of row positions for each queen.
    f1 := new array[n/2]
    f2 := new array[n-n/2]
    for i in f1 do
        f1[i] := 2i
    end
    for i in f2 do
        f2[i] = 2i+1;
    end
    if n mod 6 = 2 then
        swap(f2[0], f2[1])
        move_to_end(f2[2])
    end
    if n mod 6 = 3 then
        move_to_end(f1[0])
        move_to_end(f2[0])
        move_to_end(f2[1])
    end
    f := f1 cat f2

    return f
```

(note: Used reference from: https://en.wikipedia.org/wiki/Eight_queens_puzzle which uses: https://dl.acm.org/doi/10.1145/122319.122322 as a reference itself)

6. See attached file: HamiltonianPath.jpg

7. The best case scenario for the Assignment Problem would be:
$$\begin{bmatrix} 1 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{bmatrix}$$

8. For n=4, there would be 9 nodes in the state tree. See attached file: BranchAndBoundAssignment.jpg

9. See attached file: BranchAndBoundKnapsack.jpg

10.
```
        BESS K|NEW A|BOUT |BAOBABS
        BAOBAB|     |     |
              |BAOBA|B    |
                    |BAOBA|B
                          |BAOBAB
```

11.
```
Algorithm : Greedy Assignment
    Input: a[n][n] matrix representing jobs
    Result: The minimum cost of n jobs with n people
    for i in jobs do
        for j in people do
        if job is already picked then
            continue
        else
            current_min:=current_min + a[i][j]
        end
        if current_min < min then
            min:=current_min
        end
    end

    return min
```

12.
```
Algorithm : Rumor Spreading
    Input: number of people
    Result: minimum number of messages needed to be sent
    count:=0
    for i in n-1 do
        i sends message to i+1
        count:=count+1
    end
    for i in n-1 do
        n sends message to i
        count:=count+1
    end

    return count
```

2. Dynamic Programming - 7.5 points

See attached files: fib.cpp, makefile

References

I discussed possible approaches with Kimlong Seng, Emerson Flom, and Matthew Behnke. Our textbook was used as a reference:

Introduction to the Design and Analysis of Algorithms (3rd ed.) [Levitin 2011-10-09]

I also used the following as a reference:

https://en.wikipedia.org/wiki/Eight_queens_puzzle

which uses:

https://dl.acm.org/doi/10.1145/122319.122322

as a reference itself