**2 hours**

**UNIVERSITY OF MANCHESTER**
**SCHOOL OF COMPUTER SCIENCE**

# COMP26120 Revision for Part C

**May 2019**

**Time: ?**

## Marking Scheme Included

## Do not publish

Answer all questions

---

The use of electronic calculators is permitted provided they are not programmable and do not store text.

---

## Section A

1.

2.

3.

# Section B

4.

## Section C

Throughout this section you may assume the existence of standard data structures (e.g. lists, queues, stacks, trees) but to get full marks you must state the complexity of the operations you perform on your data structure(s).

5. It is your first day as Ant and Dec's new agent helping them set up their new Game Show. Solve all of the problems and you can have a nice sit down by the pool.

   a) Your first job is to help Ant and Dec organise a charity fundraiser to advertise the show. They need to work out how much airtime to give to different celebrities. They have a list of the amount of time each celebrity is available for and the expected donation they will receive for every 10 minutes the celebrity is on air. For example:

   | Celebrity | Availability | Donations |
   |---|---|---|
   | Robbie Williams | 30 minutes | £50 per 10 minutes |
   | Ed Balls | 300 minutes | £0.1 per 10 minutes |
   | David Mitchel | 10 minutes | £10 per 10 minutes |
   | Pudsy the Dog | 2 minutes | £500 per 10 minutes |

   You ask Ant and Dec how often they have to do something like this and they tell you that it's about once a week so you decide to design a general solution.

   Let the length of the fundraiser be $N$ minutes, let $k$ be the number of celebrities on the list, and let $a_i$ and $d_i$ be the availability and donation rate for celebrity $i$. Design a general solution for deciding how much airtime to give each celebrity to maximise donations. You confirm with the producer that there is no minimum amount of time a celebrity can be on air e.g. they could be on air for just 1 second if needed.

   Give the complexity of your solution in terms of $N$ and $k$. (5 marks)

**Model answer:** This is a fractional knapsack problem that can be solved efficiently using a greedy algorithm.
General solution:

i. Compute for each row the maximum amount of money that can be earned as $a_i \times d_i$

ii. Sort rows by this value

iii. Select the first $m$ rows in this list such that it is the largest number of rows such that their total time $M$ is still less than $N$

iv. Select $N - M$ minutes from the next row (if there is a next row)

The complexity is $O(k \log k)$ e.g. it is dominated by the sorting operation as all other steps are linear in $k$.
**Distribution of Marks:** 1 mark for noticing it is fractional knapsack. 3 marks for greedy solution (computing value, sorting, selecting). 1 mark for complexity.

b) Next, Ant has proposed a new game for their show but Dec is uncertain if they will have time for it. The proposed game is as follows:

1. Ant hides an inflatable toy somewhere in the audience.
2. Dec splits the audience in half and gets them to cheer if the toy is in their half
3. Dec repeatedly splits the audience and repeats (2) until he finds the toy

You notice that this game is following an *algorithmic technique*. Identify the technique. Given that it takes 30 seconds to get the right number of audience members to cheer, how long will the game take if there are $N$ people in the audience (evenly distributed). (3 marks)

**Model answer:** The algorithmic technique is *divide and conquer*. Time taken is in the order of $30 \times \lceil \log_2 N \rceil$ seconds - the ceiling operator is there because we cannot perform a fraction of a round.
**Distribution of Marks:** 1 mark for identifying technique. 2 marks for the time required (1 for $\log_2$, 1 for taking the ceiling + multiplying by 30 appropriately).

c) This time Dec proposes a new game for their show. A contestant is faced with a floor made up of a grid of numbers. For example:

| | | | | |
|---|---|---|---|---|
| 40 | 70 | 80 | 60 | 40 |
| 60 | 70 | 30 | 90 | 20 |
| 30 | 80 | 10 | 20 | 30 |
| 70 | 10 | 70 | 30 | 70 |
| 20 | 90 | 80 | 90 | 30 |

They must then walk from one corner (e.g. top left) to the opposite corner (e.g. bottom right) by stepping on squares that are to their right, below them, or diagonally below them to the right (e.g. towards the opposite corner only). They then win the total of the value of the tiles they step on e.g. the above path would win them £300.

Ant has remembered that all prize money comes out of their lunch budget and is worried that there won't be enough money left over for his favourite brand of sausage roll. Devise a general solution that, given a board, computes the maximum amount of money a contestant can win.

They will only have a short amount of time to prepare for each game so a brute-force solution will not be fast enough (and will only receive partial marks).
(6 marks)

**Model answer:**
Best Solution: Dynamic Programming
This is similar to the example in lectures with the Chess board. As with most dynamic programming tasks we need to decide (i) how to update the value of a square, and (ii) how to iterate through the table. In pseudocode:

```
for k=0 to 2N-1
  for j=0 to k
   i = k-j
   if i < N and j < N then
    aboveLeft = values[i-1][j-1] if defined else 0
    above = values[i-1][j] if defined else 0
    left = values[i][j-1] if defined else 0
    values[i][j] = max(aboveLeft,above,left) + table[i][j]
return values[N-1][N-1]
```

Assuming a $N \times N$ grid. Solution is $O(N^2)$ although question didn't ask for complexity

We cannot use Dijkstra's here, which would be possible (although not optimal) if the problem were to find a 'shortest' path (here we want the longest). Note that the graph is acyclic so another graph algorithm may be possible.

The brute force solution involves enumerating all paths and checking their value.

**Distribution of Marks:** 1 mark for dynamic programming approach. 1 mark for overall algorithm structure. 2 marks for iteration. 2 marks for updating table. 3 marks for a correct brute-force solution.

d) The final game (proposed by their friend Cat Deeley) requires pitching teams of families or friends against each other for a number of mini-challenges. Given teams with $m$ and $n$ contestants respectively there will be $m \times n$ mini-challenges played. Each mini-challenge takes a fixed constant amount of time $c$ but the format of the show requires the overall game to take a different amount of time each week. You propose dealing with this by varying the number of contestants.

Before each show you will be given a list of teams and their number of contestants. For example:

| Team Name | Contestants |
|---|---|
| The A Team | 4 |
| McBusted | 6 |
| The Regers | 3.5 |
| Take that | $\cancel{5}$ $\cancel{4}$ 3 |

You must find a pair of teams such that their mini-challenges can fit exactly into a given time $N$. Given that the show is running once a day, you decide to design a general solution to the problem. Give your solution and its complexity, remembering to be explicit about the data structures you are using and the complexity of operations you perform on them.

(5 marks)

**Model answer:** Note that there was a typo in this question that was fixed when writing up the solution, sorry! It should have been clear that we need to make $N$ exactly.

You might recognise this problem as the problem posed in the very first lecture of the course. We can rephrase the problem as: given a list of numbers find a pair of different numbers such that their product is less than $N$.

The simplest (maybe not most efficient but good enough) solution is to sort the list and then maintain a *bottom* pointer and a *top* pointer initialised to the first and last element in the list then run

```
while top > bottom
 if list[top]*list[bottom]
   return team names of top and bottom
 if list[top]*list[bottom] < N
  bottom = bottom+1
 else
  top = top-1
return ``No solution''
```

As we are working on the number of contestants and need to return team names then we should probably store a team in a structure containing the name and number of contestants and implement a sorting function on such structures that sorts by number of contestants only.

The complexity is dominated by sorting, which is $O(k \log k)$.

**Distribution of Marks:** 4 marks for solution and 1 mark for stating correct complexity. Lose a mark if doesn't ensure that two different teams returned. Lose a mark if no comment on how to make sure team names are returned. Lose a mark if a data structure is used without stating complexity of operations being used.

e) Finally, Ant and Dec ask you to maximise the fee they get paid for the show. Controversially, the production company have said that they will pay Ant £1k per minute he is on screen but they will by Dec £1.2k. They also stipulate that their joint screen time must be lower than 60 minutes but their individual screen time must be at least 20 minutes.

Cast this problem as a linear programming problem and solve it using the Simplex algorithm. State how many minutes of screentime each star should make and what their overall fee will be between them (they share everything). (8 marks)

**Model answer:** Hopefully this is an obvious linear programming question.
Let $a$ be Ant's screen-time and $d$ be Dec's screen-time. The problem is

$$\begin{array}{ll} \text{Maximize:} & z = a + 1.2d \\ \hline \text{Subject to:} & a + d \leq 60 \\ & a \geq 20 \\ & d \geq 20 \end{array}$$

Set up initial tableaux using positive slack variables for $\leq$ and negative surplus variables for $\geq$ e.g. we have the following. If using the M-method properly then we also need artificial variables for the last 2 constraints but if we can solve the following tableaux it will be a solution.

| $a$ | $d$ | $s_1$ | $s_2$ | $s_3$ | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 60 | $s_1$ |
| 1 | 0 | 0 | $-1$ | 0 | 20 | $s_2$ |
| 0 | 1 | 0 | 0 | $-1$ | 20 | $s_3$ |
| $-1$ | $-1.2$ | 0 | 0 | 0 | 0 | |

Then iterate process of identifying entering/leaving variables and performing elimination.

| $a$ | $d$ | $s_1$ | $s_2$ | $s_3$ | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 40 | $s_1$ |
| 1 | 0 | 0 | $-1$ | 0 | 20 | $a$ |
| 0 | 1 | 0 | 0 | $-1$ | 20 | $s_3$ |
| 0 | $-1.2$ | 0 | $-1$ | 0 | 20 | |

| $a$ | $d$ | $s_1$ | $s_2$ | $s_3$ | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 20 | $s_1$ |
| 1 | 0 | 0 | $-1$ | 0 | 20 | $a$ |
| 0 | 1 | 0 | 0 | $-1$ | 20 | $d$ |
| 0 | 0 | 0 | $-1$ | $-1.2$ | 44 | |

Note that we don't actually produce a basic solution until the third iteration. We were lucky here. The process could have diverged. To stop it diverging we would use other methods but in this simple example we don't need to.

| $a$ | $d$ | $s_1$ | $s_2$ | $s_3$ | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 20 | $s_3$ |
| 1 | 0 | 0 | $-1$ | 0 | 20 | $a$ |
| 0 | 1 | 1 | 1 | 0 | 40 | $d$ |
| 0 | 0 | 1.2 | 0.2 | 0 | 68 | |

Final answer is that they get $64k$ between them with 20 minutes of Ant and 40 minutes of Dec.
**Distribution of Marks:** 3 marks for writing down the problem (objective function, two kinds of constraint) and 5 marks for solving it (2 for initial tableaux, 3 for rest). Other properly explained applications of the Simplex algorithm would be fine. **Note: this was probably more work than expected for 8 marks. If there are $\geq$ constraints in the exam then treating them as in this solution will always be sufficient.**