COMP3600/6466 — Algorithms

Convenor: Hanna Kurniawati
Lecturers: Hanna Kurniawati & Edward Kim
E-mail: comp_3600_6466@anu.edu.au

Assignment 2
Due: Monday, 31 October 2022 23:59 Canberra Time
Grace Period Ends: Tuesday, 1 November 2022 13:00 Canberra Time
Late Penalty: 100%

IMPORTANT NOTES:

- Please read the entirety of this question sheet before working on the solutions.
- Your assignment should consist of a single .pdf, named A2-[studentID].pdf (clear scans of handwritten scripts are fine) and a single .cpp file, named A2-[studentID].cpp. You should zip these two files together into a single file, named A2-[studentID].zip and submit via Wattle before the due date.
- There is a 13 hour grace period. This means, there will be no penalty if you submit before the grace period ends. However, we will NOT accept assignment submissions beyond this time (i.e. the late penalty after the grace period ends is 100%).
  - You can update your assignment until the grace period ends.
- Assignment marking:
  - There are 100 points total for this assignment which will contribute 20% to your overall mark.
  - In each question, the marking criteria will award marks for a correct answer *as well as* a sensible explanation (including mathematical derivations as appropriate). More weight will be assigned to the latter.
- Discussion with your colleagues is encouraged. However, the final script must be your own work. You will need to cite all references and the full names of everyone you have discussed this assignment with.

[30 pts] Part A

1. [10 pt] A binary tree is said to be *perfect* if every *interior node* (i.e. non-leaf node) has two children and every leaf node has the same depth. Suppose P is a perfect binary search tree.
   (a) Is P an AVL tree?
   (b) Suppose C is the resulting complete binary tree after a single leaf node of P is deleted. Is C an AVL tree?
   (c) In which node is the maximum key of C located?
   (d) In which node is the median key of P located?
2. [10 pt] Mrs Bright is provided with the source code for a hash table that resolves collisions by chaining using linked lists. Mrs Bright randomly samples n distinct keys from the universe of keys U and inserts them into a hash table of size m using the code's hash function. Assume the code's hash function satisfies the assumption of *simple uniform hashing* under her sampling distribution. Let N_i denote the length of the i-th linked list where each $i = 0, 1, \ldots, m-1$ represents the i-th slot of the table. For a given slot position i in Mrs Bright's hash table:
   (a) What is the probability that N_i is of a given length? In other words, compute $P(N_i = l)$ for $l = 0, 1, \ldots, n$.
   (b) What is the expected length of N_i? I.e. compute $E(N_i)$.
3. [10 pt] Assume that the universe of keys U is a subset of the natural numbers with the usual ordering. Mrs Bright takes the source code from Question 2 and modifies the chaining component by replacing the linked list with an AVL tree. An adversary gives her n distinct keys which she inserts into the modified hash table. What will be the resulting:
   (a) [5 pt] Worst case time complexity of inserting an additional key?
   (b) [5 pt] Worst case time complexity of searching for a key?

[40 pts] Part B

4. [20 pt] Suppose the local florist Mr Posy stocks n different species of flowers, $f_1, f_2, \ldots, f_n$ with respective positive real-valued prices $\$p_1, \$p_2, \ldots, \$p_n$. Unfortunately, economic circumstances have been tough with a recession looming, so Mr Posy has decided to adopt an unusual strategy to attract customers. He will run a mix-and-match sale on his products so that they can either be sold as: (1) an individual item f_i at the original price p_i ($i = 1, \ldots, n$), or (2) a packaged pair (f_i, f_{i+1}) that sells for $p_i p_{i+1}$ ($i = 1, \ldots, n-1$). Note, only adjacent pairs of species can be packaged; e.g. a pairing like (f_1, f_4) is not offered.

   Dr Thoughtful would like to impress his wife by buying her exactly one of each species on offer at Mr Posy's shop. Because he is prudent though, he would like to do this sparingly by working out the minimal amount he needs to spend. He decides to devise an algorithm using Dynamic Programming.

   (a) [5 pt] Give the sub-problem definitions over prefixes of the sequence $1, \ldots, n$ that Dr Thoughtful can use to devise his Dynamic Programming algorithm.

   (b) [5 pt] Using the sub-problem definitions from part (a), characterise the optimal substructure of the problem by giving the sub-problem relation. Justify, in words, why the sub-problem relation holds. Provide the base case(s) of your sub-problem relation.

   (c) [5 pt] Provide the pseudo-code for a bottom-up algorithm that will solve Dr Thoughtful's problem in $O(n)$ time.

   (d) [5 pt] Given the output of the algorithm in part (c), provide the pseudo-code for another algorithm that prints a sequence of singletons and pairs that achieves the minimum price. E.g. for the problem with $n = 3$ and price list $(0.4, 30, 0.5)$, the code should ~~return~~ print some permutation of the sequence $\{(1, 2), 3\}$ where each integer here represents an index of a flower species. The algorithm should run in $O(n)$ time.

5. [20 pt] Due to the tragic news that a much-beloved queen has passed, her citizens have put together an organising committee for the funeral. As the death was sudden, the committee has little time. Tradition dictates that the monarch's funeral service must feature richly embroidered carpet that covers the length of the church aisle. That is, it must be exactly $T = 200$ metres long, no more or no less. The organisers have searched through their inventory and discovered that they have a very large collection of embroidered carpets that are suitable. Each carpet is of integer length (in metres). Due to the time constraints, the organisers have decided that they will simply line these carpets end-to-end to achieve the desired length. Cutting them to size is not an option because of their immense value; nor is overlapping the carpets, as this would not allow the embroidery to be on full display.

   However, the committee cannot be sure if this strategy will indeed exactly cover the desired length of the church. In a panic, they summon Professor Smart at the local university to see if she can help determine its feasibility.

   (a) [5 pt] Let n be the number of carpets in their collection. If the committee were to exhaustively check *all* possible selections from their carpet collection, how many selections would they encounter?

   (b) [5 pt] Give a sub-problem definition using suffixes that Professor Smart can use to devise (using Dynamic Programming) an algorithm that answers the committee's question. *Hint: The committee's question is a decision problem having True or False outputs.*

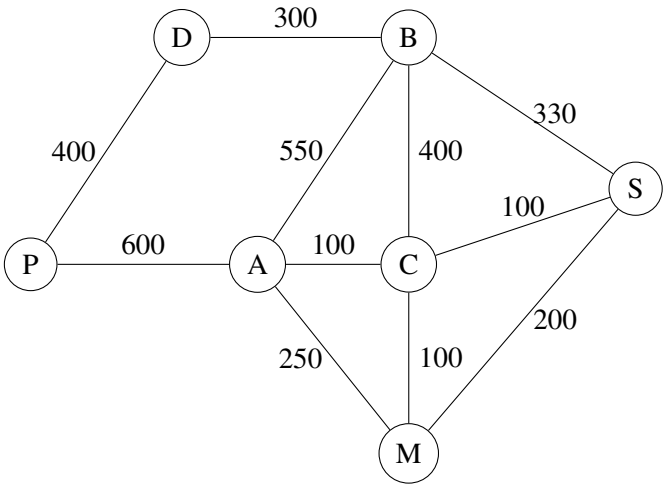
   (c) [5 pt] Using the sub-problem definitions from part (b), characterise the optimal substructure of the problem by giving the sub-problem relation. Justify, in words, why the sub-problem relation holds. Provide the base case(s) of your sub-problem relation.

   (d) [5 pt] Provide the pseudo-code for a bottom-up algorithm that will tell Professor Smart whether the committee's plan is feasible or not in $O(nT)$ time.

[30 pts] Part C

6. Zamamon is a pioneering company that trucks a large number of goods around Australia. It services transportation between $n = 7$ major cities on the mainland whose initials are A, B, C, D, M, P and S so that there are $R = \frac{1}{2}n(n-1)$ routes that it services in total (i.e. a *route* is simply an unordered pair of cities). Given rising energy costs, Zamamon has fixed its fuel prices with its suppliers over the next five years according to the graph below. Each edge weight represents the price (in dollars) of fuelling a truck for a trip between the cities at its

vertices.



Zamamon services its routes as efficiently as possible. To minimise weight, each truck only takes the precise amount of fuel to reach the next city. Then it refuels again for the next leg of the route if it needs to continue.

Following a bumper year at the tail-end of the pandemic, Zamamon's Chief Financial Officer (CFO) has allocated a budget of $\$B \times 1000$ (assume $B = 0, 1, \ldots$) towards the procurement of a self-driving fleet that will be commissioned at the start of 2023. These trucks will be fitted with the latest in self-driving technology and each truck will take the route that minimises its fuel expenditure without exception.

Unfortunately, by law, once a truck has been commissioned on a given route it will not be permitted to service any other route; e.g. a car that has been assigned to service the route between A and S will not also be allowed to service the route between C and S. Furthermore, Zamamon's engineers have placed a maximum cap of $T > 0$ trucks on any given route, to ensure that every truck that is allocated to a given route can be properly serviced. Zamamon's analysts have come up with the following estimates for the annual revenue generated on each route

$$\text{Expected annual revenue on route } r \text{ being serviced by } t \text{ trucks} = 50(g_r - f_r)\,t \qquad t = 0, 1, \ldots, T \qquad (1)$$

where $\$g_r$ is the estimated gross revenue per week for a truck on route r and $\$f_r$ is the fuel expenditure required to complete a trip on route r. Because of Zamamon's relationships with its suppliers and its engineers' know-how, it can procure and commission a self-driving truck on any route r at a fixed cost $\$c_r \times 1000 > 0$ (assume c_r is an integer).

The CFO has asked you to procure trucks for each route in such a way that: (i) you remain within the budget $\$B \times 1000$ for the project and (ii) the self-driving fleet maximises expected annual revenue over all of its routes. Zamamon has ambitions to scale up its operations to many more routes in coming years, so you sense the CFO will ask you the similar questions in the future. To avoid unnecessary future work you design and implement an algorithm to answer her question.

(a) [5 pts] Provide the pseudo-code for a bottom-up dynamic programming algorithm to determine f_r for each route r in $O(n^3)$ time.

(b) [10 pts] Suppose each f_r has been computed using the algorithm from part (a). Provide the pseudo-code for a bottom-up dynamic programming algorithm that will: (i) return the maximum expected annual revenue, (ii) print the total capital expenditure (i.e. expenditure on trucks), and (iii) print the number of trucks that should be purchased for each route under the CFO's mandate.

(c) [5 pts] Analyse the asymptotic time complexity of your pseudo-code in part (b).

(d) [10 pts] Implement the algorithm you have designed in part (b) in C++.

Input to Program is the name of a text (.txt) file. The text file will contain $(R + 1)$ lines whose entries are separated by a white space. The first line will store the parameters ~~T and B~~ T, B and R in that order. Each line in the subsequent R lines is associated with a given route (i.e. line $r + 1$ in the file will be associated with route r) and will store the parameters f_r, g_r and c_r.

Example: The input to the program is input1.txt, where the contents of input1.txt are as follows:

```
37 4750 6
800 1300 50
200 1000 45
330 1000 45
100 1200 47
630 1000 50
200 1300 40
```

The first line of the above input file means that there is a cap of 37 trucks per route and the available total budget is \$4.75 million and six routes. Each subsequent line represents a route so that line 7 means that $f_6 = 200$, $g_6 = 1300$ and $c_6 = 40$

Output of the Program is three lines in standard output.

- The first line should contain four numbers separated by a single white space and ended with a line break. The first number should be the truck capacity for all routes T, the second number the budget B, the third number the total capital expenditure (use units of \$ '000) and the fourth the maximum expected annual revenue over all its routes (again in units of \$ '000).
- The second line should have R entries representing the indices for each route in order.
- The third line should have R entries representing the number of trucks that will be commissioned on that route.

If more than one solution exists, any set of them will be acceptable.
If the problem has no solution, the output should be "No Solution".

Example of the output for the input1.txt (input example above):
37 4750 4749 5430
1 2 3 4 5 6
0 34 0 37 0 37

Program Marking: If your program compiles and runs, you will get 2 points. We will run your program on 4 test cases. For each test case, your dynamic programming algorithm (including data structure construction) will be given a total of $0.0001 \times B \times R \times T$ ms CPU time to find a solution. You can assume your program will have access to at most 12GB RAM. It will be run as a single thread process on a computer with Intel i7 3.4GHz processor. For each test case that your program solves correctly within the given time and memory limit, you will get 2 points.

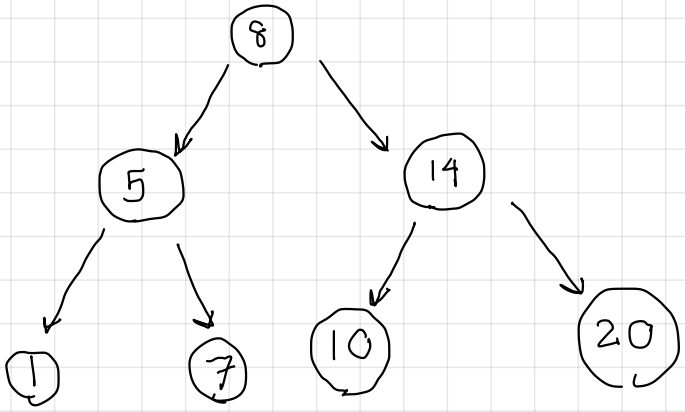
oOo That's all for the questions folks oOo

[30 pts] Part A

1. [10 pt] A binary tree is said to be *perfect* if every *interior node* (i.e. non-leaf node) has two children and every leaf node has the same depth. Suppose P is a perfect binary search tree.

   (a) Is P an AVL tree?

Assumption: Example of perfect binary tree refer to the question



a) AVL tree is a binary tree that hold an additional properties "Balance Rule" which is any node X. Left subtree and Right subtree ^ absolute height differ must not exceed 1

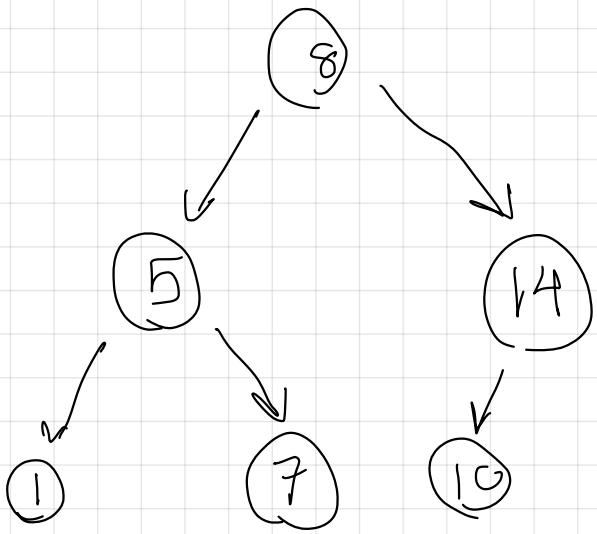
Therefore, from the example of perfect binary tree, it is obvious that every single node of P. The height different is 0 which is hold the property of AVL tree.

(b) Suppose C is the resulting complete binary tree after a single leaf node of P is deleted. Is C an AVL tree?

(c) In which node is the maximum key of C located?

Complete binary tree is a special type of binary tree where all the levels of the tree are filled completely except the lowest level nodes which are filled from as left as possible

Assumption From P perfect binary tree delete a single leaf node (20) from perfect binary tree example.



b). From balancing property of AVL tree. C is still an AVL which diff height of node of Left subtree is 1 which still not violate the AVL tree property

(c) In which node is the maximum key of C located?

(d) In which node is the median key of P located?

c. from the example of tree C above. The maximum key of C is the predecessor of that deletion node which is (14) (20) or the most left node in result or inorder C tree traversal

d. From P tree property as a perfect binary tree the location of median key can find in case of odd nodes as $\left(\frac{n+1}{2}\right)$ node in inorder tree traversal

from example P, n = 7

so $\left(\frac{7+1}{2}\right)$ = 4th node

4th node.

inorder 1 5 7 (8) 10 14 20

(d) In which node is the median key of T located?

2. [10 pt] Mrs Bright is provided with the source code for a hash table that resolves collisions by chaining using linked lists. Mrs Bright randomly samples n distinct keys from the universe of keys U and inserts them into a hash table of size m using the code's hash function. Assume the code's hash function satisfies the assumption of *simple uniform hashing* under her sampling distribution. Let N_i denote the length of the i-th linked list where each $i = 0, 1, \ldots, m-1$ represents the i-th slot of the table. For a given slot position i in Mrs Bright's hash table:

   (a) What is the probability that N_i is of a given length? In other words, compute $P(N_i = l)$ for $l = 0, 1, \ldots, n$.

   (b) What is the expected length of N_i? I.e. compute $E(N_i)$.

a) Given: n distinct keys $\binom{1}{i}\binom{n}{l}$

N_i : # keys that insert into hash table

i slots represent $i = 0, 1, \ldots, m-1$

$P(N_i = 1) + P(N_i = 2) + \ldots P(N_i = n)$

$$P(N_i = l) = \binom{n}{l}\left(\frac{1}{m}\right)^l \left(1 - \frac{1}{m}\right)^{n-i} \quad \text{for } l = 0, 1, \ldots, n$$

b)

$$E[N_i] = \sum_{l=1}^{n} P(N_i = l) :$$

$$= \sum_{l=1}^{n} \binom{n}{l}\left(\frac{1}{m}\right)^l \left(1 - \frac{1}{m}\right)^{n-1}$$

$$= \frac{n}{m}$$

(b) What is the expected length of N_i? I.e. compute $E(N_i)$.

3. [10 pt] Assume that the universe of keys U is a subset of the natural numbers with the usual ordering. Mrs Bright takes the source code from Question 2 and modifies the chaining component by replacing the linked list with an AVL tree. An adversary gives her n distinct keys which she inserts into the modified hash table. What will be the resulting:
   (a) [5 pt] Worst case time complexity of inserting an additional key?
   (b) [5 pt] Worst case time complexity of searching for a key?

a. Normally it is constant time to specify the location in hash table for the assumption $O(1+X)$ is the time complexity of inserting an item which in this cases replace linked list into AVL which AVL tree inserting an element hold $\log_2(n)$ so replace X with $\log_2(n)$ so $O(1+\log_2 n)$ which $\log_2(n)$ will consumed 1 which is a constant. so the worst case time complexity equal $O(\log_2 n)$

b. The time complexity for searching for a key in AVL tree is $O(\log_2 n)$ So if we use AVL tree instead of linked list we will get the worst case time complexity for searching is $O(\log_2 n)$

[40 pts] Part B

4. [20 pt] Suppose the local florist Mr Posy stocks n different species of flowers, $f_1, f_2, \ldots, f_n$ with respective positive real-valued prices $\$p_1, \$p_2, \ldots, \$p_n$. Unfortunately, economic circumstances have been tough with a recession looming, so Mr Posy has decided to adopt an unusual strategy to attract customers. He will run a mix-and-match sale on his products so that they can either be sold as: (1) an individual item f_i at the original price p_i ($i = 1, \ldots, n$), or (2) a packaged pair (f_i, f_{i+1}) that sells for $p_i p_{i+1}$ ($i = 1, \ldots, n-1$). Note, only adjacent pairs of species can be packaged; e.g. a pairing like (f_1, f_4) is not offered.

   Dr Thoughtful would like to impress his wife by buying her exactly one of each species on offer at Mr Posy's shop. Because he is prudent though, he would like to do this sparingly by working out the minimal amount he needs to spend. He decides to devise an algorithm using Dynamic Programming.

   (a) [5 pt] Give the sub-problem definitions over prefixes of the sequence $1, \ldots, n$ that Dr Thoughtful can use to devise his Dynamic Programming algorithm.

   (b) [5 pt] Using the sub-problem definitions from part (a), characterise the optimal substructure of the problem by giving the sub-problem relation. Justify, in words, why the sub-problem relation holds. Provide the base case(s) of your sub-problem relation.

   (c) [5 pt] Provide the pseudo-code for a bottom-up algorithm that will solve Dr Thoughtful's problem in $O(n)$ time.

   (d) [5 pt] Given the output of the algorithm in part (c), provide the pseudo-code for another algorithm that prints a sequence of singletons and pairs that achieves the minimum price. E.g. for the problem with $n = 3$ and price list $(0.4, 30, 0.5)$, the code should ~~return~~ print some permutation of the sequence $\{(1, 2), 3\}$ where each integer here represents an index of a flower species. The algorithm should run in $O(n)$ time.

a. Based on prefixes of the sequence 1,...,n Dr. thought full need to choose the V[i] minimum value for picking all the flowers based on store promotion on pick in single price or pick in adjacent form price

b. V[i] = min(V(i-1) + p_i, V(i-2) + $p_i p_{i-1}$) $i \in [3, n]$ with the sub problem we will continously decide the finest form to get the best price

which we need to also concerned the bases case which $i=1, 2$

so

$V[i]$ {

- P_i ; $i=1$ → which we only got this option to choose
- $\min(P_i + P_{i-1}, P_i P_{i-1})$; $i=2$
  - ↳ choose between pick seperately, or choose in adjacent promotion
- $\min(V(i-1) + P_i, V(i-2) + P_i P_{i-1})$
  - ↳ our relation to build up the recurrence relation

(c) $V[1 \ldots n]$ // array for memoization

```
Solve (P[1, n])
    for i=1 to n
        if (i=1) //base case.
            V[i] = P_i
        elif (i=2) // base case
            V[i] = min(P_i + P_{i-1}, P_i x P_{i-1}) // choose best formation
        else
            ↓
```

$V[i] = \min(V[i-1] + P_i , V[i-2] + P_i P_{i-1})$

return $V[n]$

which this algorithm only require $O(n)$ time complexity for outer loop in inner loop it constant time to interact array with specific index

(d). optimize part c. algorithm to get printed output

```
Seq[1.... n]  // array for store string.
V[1... n]     // array for memoization.
solve ( P[1.....n] )
    for  i : 1  to  n
        IF (i : 1)
            V[i] = P_i
            Seq[i] : string(i)
        else if  (i = 2)
            if  ( P_i + P_{i-1} < P_i P_{i-1} )
                V[i] = P_i + P_{i-1}
                Seq[i]: string(i-1) + " , " + string(i)
            else
                V[i] = P_i P_{i-1}
                Seq[i]: " ( " + string(i-1) + " , " + string(i) + " ) "
```

else

if $(V[i-1] + P_i < V[i-2] + P_i \times P_{i-1})$

$V[i] = V[i-1] + P_i$

seq[i] = seq[i-1] + "," + String(i)

else

$V[i] = V[i-2] + P_i \times P_{i-1}$

seq[i] = seq[i-2] + "," + "(" + String(i-1) + "," + String(i) + ")"

print seq[i].

Adding another array manipulation inside inner function still hold constant time.

Hence, this algorithm still get O(n) time complexity

each integer here represents an index of a flower species. The algorithm should run in $O(n)$ time.

5. [20 pt] Due to the tragic news that a much-beloved queen has passed, her citizens have put together an organising committee for the funeral. As the death was sudden, the committee has little time. Tradition dictates that the monarch's funeral service must feature richly embroidered carpet that covers the length of the church aisle. That is, it must be exactly $T = 200$ metres long, no more or no less. The organisers have searched through their inventory and discovered that they have a very large collection of embroidered carpets that are suitable. Each carpet is of integer length (in metres). Due to the time constraints, the organisers have decided that they will simply line these carpets end-to-end to achieve the desired length. Cutting them to size is not an option because of their immense value; nor is overlapping the carpets, as this would not allow the embroidery to be on full display.

   However, the committee cannot be sure if this strategy will indeed exactly cover the desired length of the church. In a panic, they summon Professor Smart at the local university to see if she can help determine its feasibility.

   (a) [5 pt] Let n be the number of carpets in their collection. If the committee were to exhaustively check *all* possible selections from their carpet collection, how many selections would they encounter?

   (b) [5 pt] Give a sub-problem definition using suffixes that Professor Smart can use to devise (using Dynamic Programming) an algorithm that answers the committee's question. *Hint: The committee's question is a decision problem having True or False outputs.*

   (c) [5 pt] Using the sub-problem definitions from part (b), characterise the optimal substructure of the problem by giving the sub-problem relation. Justify, in words, why the sub-problem relation holds. Provide the base case(s) of your sub-problem relation.

   (d) [5 pt] Provide the pseudo-code for a bottom-up algorithm that will tell Professor Smart whether the committee's plan is feasible or not in $O(nT)$ time.

[30 pts] Part C

a). if committe decidede to do exhaustively to check, their need to find subsets that sum equal to 200. so a power set contains all those subsets generated from a given set. The size of such a power set is 2^N. so they would encounter 2^n which n is the number of carpet collection they are looking through.

b). Construct table of boolean as bool subset [n+1] [sum+1] from suffix [i:] [j:]

element

sum

c). seperate case if sum is 0 then true

subset[i][0] = true

if sum is not 0 and set is empty then.

subset[0][i] = false

otherwise

we use suffix to fill the table

in case sum < set[j:]

subset[i][j] = subset[i+1][j];

else

sum >= set[j:]

subset[i][j] = subset[i+1][j]

|| subset[i+1][j-set[i+1]]

d.) bool help(int set[], int target)

bool subset[set[].lenght+1][target]

for(int i = set.lenght; i >= 0 ; i--)

subset[i][0] = true;

for(int i = sum; i >= 1; i--).

subset[0][i] = false;

↓

```
for (int i = last element-1; i >= 1; i--){
    for (int j = target; j >= 1; j--){
        if (j < set[i+1])
            subset[i][j] = subset[i+1][j];
        if (j >= set[i+1])
            subset[i][j] = subset[i+1][j]
                || subset[i+1][j-set[i+1]];
    }
}
return subset[1][1];
}
```

So we iterate through array O(n) time but it require work to check if target - set[i] so it O(nT) time complexity.

target

[30 pts] Part C

6. Zamamon is a pioneering company that trucks a large number of goods around Australia. It services transportation between $n = 7$ major cities on the mainland whose initials are A, B, C, D, M, P and S so that there are $R = \frac{1}{2}n(n-1)$ routes that it services in total (i.e. a *route* is simply an unordered pair of cities). Given rising energy costs, Zamamon has fixed its fuel prices with its suppliers over the next five years according to the graph below. Each edge weight represents the price (in dollars) of fuelling a truck for a trip between the cities at its

an algorithm to answer her question.

(a) [5 pts] Provide the pseudo-code for a bottom-up dynamic programming algorithm to determine f_r for each route r in $O(n^3)$ time.

Assume: We input zamamon graph paht and we interested to find minimum cost at A → S = 200

A → C → S = 100 + 100

then in graph assume low_{as} as lowest cost path an created a array to store min cost on a to S

if m = 0 so a = S $d_{as}(0) = 0$

if a ≠ s so $d_{as}(0) = \infty$

so $d_{as}(V - f_r)$ for min fr

$$d_{as}(m) = \min\left[d_{as}(m - fr), \min(d_{ac}(m - fr) + w_{cs})\right]$$

$$= \min_{1 \le k \in V}\left(d_{as}(m - fr) + w_{cs}\right)$$

then

interested path

```
minimumPath( Graph, s ){
  V = |V|
Dm-1[1...V] = ∞
Dm-1[s] = 0
for(n=1 to V-1){
    for(u=1 to V){
      Dm[u] = Dm-1[u]
        for(k=1 to V){
          Dm[u] = min(Dm[u], Dm-1[k] + w(k,u))
    }
}
 Dm-1[1...V] = Dm[1...V]
}
return Dm
```

array for the min cost path

(b) [10 pts] Suppose each f_r has been computed using the algorithm from part (a). Provide the pseudo-code for a bottom-up dynamic programming algorithm that will: (i) return the maximum expected annual revenue, (ii) print the total capital expenditure (i.e. expenditure on trucks), and (iii) print the number of trucks that should be purchased for each route under the CFO's mandate.

(c) [5 pts] Analyse the asymptotic time complexity of your pseudo-code in part (b).

construct an array of value

{ |——— ,, ———| weight

which is tricky cause we need to also concerned on truck for each path cause we got T truck so.

we just distributed the each route with 1... T truck so our size of value is val [6 x 3 8] from following input example

val [route 1 x T , route 2 x T ... route x T]

weight [cost r1 xT , cost r2 x T , ...]

the we construct dp table for our memoization process

dp[ ][ ] = [val.size][B] of 0

Solve (val, wt, B)

B maximum weight our max pay cost

```
for(i = 0 to val.size)
    for(j = 0 to B )
        if (i = 0 || j = 0){
            arr[i][j] = 0
        }else if( Wt[i-1] <= j ) {
```

```
dp[i][j] = max(val[i-1] +
        dp[i-1][j-wt[i-1]],
        dp[i-1][j]);
    } else {
        dp[i][j]=dp[i-1][j];
    }
return.
```

(i) maxExpectedRev = -dp[val.size][B]/100; #

result = maxExpectedRev

temp = B;

then we trace back our table path.

```
for (i=val.size to 1)
    if (result == dp[i-1][temp]) {
        continue;
    }else{
```

because our table split the truck so we count back on route,

(iii) trucks[(i-1)/T)] += 1; #

(ii) totCapEx += wt[i-1]; ※

```
        result = result -val[i-1];
        temp = temp -wt[i-1];
    }
```

print totCapEx ii

print (for each trucks) iii

should be purchased for each route under the CFO's mandate.

(c) [5 pts] Analyse the asymptotic time complexity of your pseudo-code in part (b).

actually it spend most time at construct the val[] and wt[] on O(R x T)

```
if(solExists){
    vector<int> val;
    vector<int> wt;
    for(int i = 0; i <= R; i++){
        int v = 50 * (g[i]-f[i]);
        for(int j = 0; j < T; j++){
            val.push_back(v);
            wt.push_back(c[i]);
        }
    }

    vector<vector<int>> dp(val.size()+1,vector<int>(B+1));

    int dp[val.size()+1][B+1];
    */
    //Build table dp[][] for bottom up manner
    for(int i = 0; i <= val.size();i++){
        for(int j = 0; j <= B;j++){
            if(i==0 || j==0){
                dp[i][j] = 0;
            }else if(wt[i-1]<=j){
                dp[i][j] = max(val[i-1]+
                               dp[i-1][j-wt[i-1]],
                               dp[i-1][j]);
            }else{
                dp[i][j] = dp[i-1][j];
            }
        }
    }
```

in solving pseda code it depends on val.size() which I would call n and in inner loop it still need to decide the max(val) on maximum b cost so in solve it is definitely O(n x B) which n is val.size for construct val array and B is maximum cost.