Major 2 Old Exams:

062_Major2

3. (45 points) Dynamic Programming
   a. (10 points) Write the recursive solution to the longest common subsequence problem
   b. (25 points) Use dynamic programming to find the length of the longest common subsequence and a longest common subsequence of the two strings: *xyxyxxxzy* and *zxxyzxzy*.
   c. (10 points) Is the longest common subsequence unique in this case? If yes, justify your answer. If no, give two different longest common subsequences.

(a)

Let $L[i,j]$ be longest common subsequence between the first i letters of string A and first j letters of string B.

$$L[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ L[i-1,j-1]+1 & \text{if } a_i = b_j \\ \max\{L[i-1,j], L[i,j-1]\} & \text{if } a_i \neq b_j \end{cases}$$

(b)

		x	y	x	y	x	x	x	z	y
	0	0	0	0	0	0	0	0	0	0
z	0	↑0	←0	↑0	↑0	↑0	↑0	↑0	↖1	←1
x	0	↖1	←1	↖1	←1	↖1	↖1	↖1	↑1	↖2
x	0	↖1	↑1	↖2	←2	↖2	↖2	↖2	←2	↖2
y	0	↑1	↖2	↑2	↖3	←3	←3	←3	←3	←3
z	0	↑1	↑2	↑2	↑3	↑3	↑3	↑3	↖4	←4
x	0	↖1	↑2	↖3	↑3	↖4	↖4	↖4	↑4	↖5
z	0	↑1	↑2	↑3	↑3	↑4	↑4	↑4	↖5	↑5
y	0	↑1	↖2	↑3	↖4	↑4	↑4	↑4	↑5	↑5

(c) (1) x x y z z (2) x x y z x

No, there are two different subsequences.

092_Major2

II. (55 points) Dynamic Programming Algorithms

1. (30 points) Consider the matrix chain multiplication problem:
   a. (6 points) This problem is one application that is solved using dynamic programming. What are the two main properties of an optimization problem that direct an algorithm designer to investigate dynamic programming as a possible solution? Explain them very briefly.

(1) Optimal substructures, where an optimal solution consists of sub solutions that are optimal.

(2) Overlappy sub problems. Meany some sub problems are solved multiple times.

   b. (10 points) Write the recursive solution for finding the cost of the optimal parenthesization in evaluating the product of n matrices with dimensions

$$M_1 = r_1 \times r_2,\ M_2 = r_2 \times r_3,\ M_3 = r_3 \times r_4,\ \ldots,\ M_n = r_n \times r_{n+1}$$

We can represent the cost of optimal parenthesization of $M_1 \ldots M_n$ as $C[1, n]$.

cost of multiplying $M_1 \ldots M_{K-1}$ with $M_K \ldots M_n$ can be represented as $r_1 \times r_K \times r_{n+1}$

Therefore, total optimal cost of $M_1 \ldots M_n$ parenthesization is:

$$C[1,n] = \min_{2 \le K \le n} \left[C[1, K-1] + C[K, n] + r_1 r_K r_{n+1} \right]$$

c. (10 points) Fill the table below by running the dynamic programming algorithm for matrix chain multiplication developed in class on the following matrices to find the least number of scalar multiplications.

$r_1 = 5, r_2 = 2, r_3 = 3, r_4 = 6, r_5 = 4, r_6 = 2, r_7 = 4$

1 2 3 4 5 6

0 M_1	30 $M_1..M_2$	96 $M_1[M_2..M_3]$	124 $M_1[M_2..M_4]$	116 $M_1[M_2..M_5]$	152 $M_1[M_2..M_6]$	1
	0 M_2	36 $M_2..M_3$	84 $[M_2..M_3]M_4$	96 $M_2[M_3..M_5]$	112 $[M_2..M_5]M_6$	2
		0 M_3	72 $M_3..M_4$	84 $M_3[M_4..M_5]$	108 $[M_3..M_5]M_6$	3
			0 M_4	48 $M_4..M_5$	96 $[M_4..M_5]M_6$	4
				0 M_5	32 $M_5..M_6$	5
					0 M_6	6

C[1,6]

minimum of the following {k from 2 to 6}

k=2: C[1,1] + C[2,6] + $r_1 r_2 r_7$ = 0 + 112 + 5×2×4 = 112 + 40 = 152

k=3: C[1,2] + C[3,6] + $r_1 r_3 r_7$ = 30 + 108 + 5×3×4 = 138 + 60 = 198

k=4: C[1,3] + C[4,6] + $r_1 r_4 r_7$ = 96 + 96 + 5×6×4 = 192 + 120 = 312

k=5: C[1,4] + C[5,6] + $r_1 r_5 r_7$ = 124 + 32 + 5×4×4 = 156 + 80 = 236

k=6: C[1,5] + C[6,6] + $r_1 r_6 r_7$ = 116 + 0 + 5×2×4 = 116 + 40 = 156

d. (4 points) Derive from the table the optimal exact parenthisization of the product generating the least number of scalar multiplications.

$= M_1 [M_2 \cdots M_6]$

$= M_1 [[M_2 \cdots M_5] M_6]$

$= M_1 [[M_2 [M_3 \cdots M_5]] M6]$

$= M_1 ((M_2 (M_3 (M_4 M_5))) M_6)$

2. (25 points) Consider the knapsack problem

a. (10 points) Write the recursive solution for finding the maximum value a knapsack of size C can hold from n items, each having a size and a value.

given n items with values v_i and sizes s_i

define $V[i,j]$ as the optimal value of fitting first i items in a knapsack of size j

$$V[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ V[i-1,j] & \text{if } s_i > j \\ \max\{V[i-1,j],\ V[i-1,j-s_i]+v_i\} & \text{if } s_i \le j \end{cases}$$

b. (15 points) Solve the following instance of the knapsack problem. There are four items of sizes 2, 3, 5, and 6 and values 3, 4, 5, and 7, and the knapsack capacity is 11.

s_i	2	3	5	6
v_i	3	4	5	7

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7	7	7
3	0	0	3	4	4	7	7	8	9	9	12	12
4	0	0	3	4	4	7	7	8	10	11	12	14

max value : items 4, 2, 1

IV. (50 points) NP-Completeness and Turing Machines

1. (30 points) Given the following Boolean formula, which is an instance of the *Satisfiability* problem

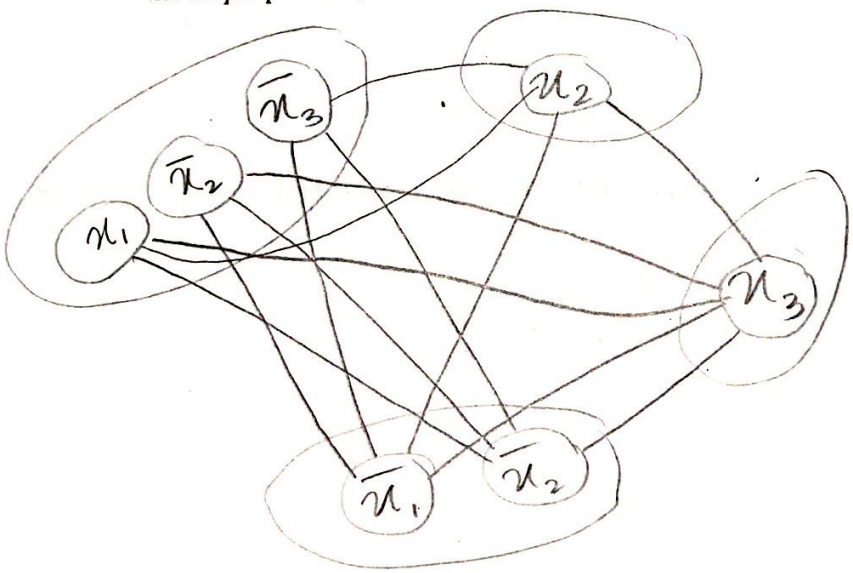
$$f = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (x_3)$$

a. (5 points) Clearly define what it means when we say a problem P is NP-Complete.

(1) It is an NP problem

(2) Every problem Π in NP is reducible in polynomial time to P. $\forall \Pi \in NP \quad \Pi \propto_{poly} P$

b. (15 points) Run the polynomial time reduction from the *Satisfiability* problem to the *Clique* problem, described in class, on the above formula *f*.



c. (10 points) Can you answer the question: "Does the Boolean formula *f* have a truth assignment such that it evaluates to true?" just from looking at the constructed *Clique* instance developed in part b? Clearly justify your answer.

No. There isn't a clique of size 4. Since there isn't a clique, we can't set truth values to the variables without having a contradiction. Hence the formula is unsatisfiable.

Final Old Exams:

062_Final

Question I (40 points):

For each statement below, indicate whether it is a true statement with **T**, or a false statement with **F**. For false statements, you have to correct the statement. Note that adding a "not" in the statement will not be considered as a correction:

a. (T) $f = O(g)$ if and only if $g = \Omega(f)$

b. (F) If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$ then $f_1(n)f_2(n)$ is in $O(\max\{g_1(n), g_2(n)\})$

$f_1(n) f_2(n) = O(g_1(n) g_2(n))$

c. (F) Average case analysis of an algorithm is the average value of its best case and worst case complexities.

It is the average value of cost of input weighted with probability.

$$= \sum_{i \in \text{inputs}} C(i) P(i)$$

d. (F) The solution to the recurrence $T(n) = \begin{cases} 1 & n = 1 \\ 5T(\frac{n}{2}) + n^2 & n \geq 2 \end{cases}$ is $T(n) = \Theta(n^2)$.

$g(n) = n^2$ $\quad n^{\log_b a} = n^{\lg 5} \approx n^{2.4}$

since $g(n) = O(n^{2.4-\epsilon})$ where $\epsilon \doteq 0.01$,

$T(n) = \Theta(n^{\lg_b a}) = \Theta(n^{\log_2 5})$

e. (F) Radix sort can sort n integers in the range $[1..n!]$ in $O(n \log n)$ time.

$= \Theta(kn)$

$k = \lg n! = \Theta(n \lg n)$

$\therefore = \Theta(n \lg n \times n) = \Theta(n^2 \lg n)$

f. (T) The SELECT algorithm, when run on n elements, guarantees that the number of elements in Partitions A_1 or A_3 does not exceed 0.7n + 1.2 elements each.

g. (F) The dynamic programming algorithm that solves the integer (0-1) Knapsack problem runs in polynomial time.

it runs in $\Theta(Cn)$
C is an integer input.
Therefore, in terms of input size
its $\Theta(2^k n)$ where $k = \log_2 C$
its exponential

h. (F) If you manage to find a polynomial time reduction from a problem $\pi \in$ NP to the Element-Uniqueness decision problem, you will prove that P = NP.

you will only prove that $\pi \in P$.
You need to proove that an NP complete problem is transformed to Element-uniqueness to prove that P=NP.

i. (T) For any language L that can be accepted by a non-deterministic Turing machine, there is a deterministic Turing machine that can accept it.

Question II (60 points): Dynamic Programming and Greedy Algorithms

In class, we have developed a dynamic programming algorithm to solve the matrix chain multiplication problem.

1. (25 points) Given 5 matrices with the following dimensions

$M_1 : 7\times 2,\ M_2 : 2\times 9,\ M_3 : 9\times 3,\ M_4 : 3\times 2$ and $M_5 : 2\times 8$

   a. (20 points) Fill the table below solving the matrix chain multiplication problem for the above instance.

   b. (5 points) Find the optimal solution corresponding to the optimal value found in part a.

1	2	3	4	5	
0 M_1	126 M_1M_2	96 $M_1(M_2M_3)$			1
	0 M_2	54 M_2M_3	66 $(M_2M_3)M_4$		2
		0 M_3	54 M_3M_4	198 $(M_3M_4)M_5$	3
			0 M_4	48 M_4M_5	4
				0 M_5	5

r_1	r_2	r_3	r_4	r_5	r_6
7	2	9	3	2	8

(a) $C[1,3] = \min\{ k=1: C[1,1]+C[2,3]+r_1r_2r_4 = 0+54+7\times2\times3 = 54+42 = 96$

$k=2: C[1,2]+C[3,3]+r_1r_3r_4 = 126+0+7\times9\times3 = 126+189 = 315$

$C[2,4] = \min\{ k=3: C[2,2]+C[3,4]+r_2r_3r_5 = 0+54+2\times9\times2 = 54+36 = 90$

$k=4: C[2,3]+C[4,4]+r_2\times r_4\times r_5 = 54+0+2\times3\times2 = 54+12 = 66$

$C[3,5] = \min\{ k=4: C[3,3]+C[4,5]+r_3r_4r_6 = 0+48+9\times3\times8 = 48+180+9\times4 = 228+36 = 258+6 = 264$

$k=5: C[3,4]+C[5,5]+r_3r_5r_6 = 54+0+9\times2\times8 = 54+144 = 198$

{ incomplete solution }

Question III (50 points): NP-Complete Problems
Consider the DOUBLE-SAT decision problem given below

DOUBLE-SAT
Instance: Boolean formula f in CNF form
Question: Does f have at least two satisfying truth assignments?

1. (15 points) Consider the following Boolean formula:
$$f = (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_3) \wedge (x_3)$$
   a. (7 points) Does f have at least two satisfying truth assignments? List all its truth assignments, if any.
   b. (8 points) Let Boolean formula g be such that $g = f \wedge (y \vee \bar{y})$ where f is the Boolean formula above. Does g have at least two satisfying truth assignments? List all its truth assignments, if any.
2. (35 points) Prove that DOUBLE-SAT is NP-complete. (Hint: Part 1 of this question is very helpful)

(1) (a) x_1 = True , x_3 = True , x_2 = True
only one satisfying truth assignment.

(b) x_1 = True , x_3 = True , x_2 = True , y = True
x_1 = True , x_3 = True , x_2 = True , y = false
Yes, it does have at least 2 assignments

(2) We need to first prove that Double SAT is an NP problem.
We create a nondeterministic algorithm that guesses and verifies in polynomial time.
Guessing phase: Non deterministic algorithm guesses two assignments to the CNF that satisfy it. This is done in $O(n)$.
Verifying phase: a deterministic algorithm verifies that the number of assignments guessed is 2. Then it verifies that both the assignments satisfy the CNF. This is done in $O(n)$.

Prove that Satisfiability $\propto_{poly}$ Double SAT

Consider an instance of Satisfiability. The Input is a boolean formula f in CNF form.

Create a boolean formula g such that

f has atleast one satisfying assignment (instance of satisfiability)
$\iff$ g has atleast 2 satisfying assignment (instance of Double sat)

$g = f \wedge (y \vee \bar{y})$ such that y is a variable that does not exist in f.

g is now an instance of Double SAT.

This reduction is done in $O(n)$ time (need to verify that variable is not in f)

---

f has one satisfying assignment $\iff$ g has atleast 2 satisfying assignments.

f has one satisfying assignment $\rightarrow$ g has atleast 2 satisfying assignments

If f has 1 satisfying assignment, that assignment will be duplicated such that one is when y is true and one is when y is false. Therefore g will have atleast 2 satisfying assignments

---

g has atleast 2 satisfying assignments $\rightarrow$ f has one satisfying assignment

If g has 2 satisfying assignments, one of them is y = True, second is y = false. When we remove y, it combines into 1 satisfying assignment for f.

092_Final

Question I (40 points):

For each statement below, indicate whether it is a true statement with **T**, or a false statement with **F**. For false statements, you have to correct the statement. Note that adding a "not" in the statement will not be considered as a correction:

a. (F) The time complexity,for the dynamic programming algorithm solving the integer knapsack problem is polynomial and is $\Theta(n)$.

it is not polynomial
it is $\Theta(Cn)$ where C is a size of knapsack
C is ~~not~~ an integer input, therefore input
size = $k = \log_2 C$
$\therefore \Theta(2^k n)$ which is exponential

b. (T) If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$ then $f_1(n) + f_2(n)$ is in $O(\max\{ g_1(n), g_2(n)\})$

c. (F) The SELECT algorithm finds the k^{th} smallest element in $\Theta(n \log n)$.

in $\Theta(n)$ time

d. (F) The solution to the recurrence $T(n) = \begin{cases} 1 & n = 1 \\ 4T(\frac{n}{2}) + n^2 & n \geq 2 \end{cases}$ is $T(n) = \Theta(n^2)$

$n^{\log_2 4} = n^2$

since $g(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^2 \log n)$

e. (F) If you manage to find a polynomial time reduction from a problem $\pi \in$ NP to the halting problem, you will prove that P = NP.

halting problem is unsolvable, you are not proving anything.

f. (T) For any language L that can be accepted by a non-deterministic Turing machine, there is a deterministic Turing machine that can accept it.

g. (T) Radix sort can sort n natural numbers that are known to have values less than or equal to 999,999 in O(n) time.

~~log~~ yes = 6n

2.

Question II (60 points): NP-Complete Problems

1. (6 points) The **sum of subsets** decision problem is as follows:
   Input: A set A of n integers $\{a_1, a_2, \ldots, a_n\}$ and an integer M.
   Question: Does A contain a subset S_A such that the sum of the elements in S_A is equal to M?
   (i.e., does there exist a subset S_A such that $\sum_{a_i \in S_A} a_i = M$)
   Given $A = \{5,7,9,10,22,35,60\}$ and $M = 50$, find a subset S_A that solves the sum of subsets problem.

$S_A = \{5, 10, 35\}$

2. (28 points) The **Partition** decision problem is as follows:
   Input: A set S of n integers.
   Question: Is it possible to partition S into two subsets S_1 and S_2 so that the sum of the integers in S_1 is equal to the sum of the integers in S_2?
   a. (6 points) In order for the two subsets S_1 and S_2 to exist such that the sum of the integers in S_1 is equal to the sum of the integers in S_2, could the sum of all integers in S be odd? Explain your answer.

No, since both $\sum S_1 = \sum S_2$

and since $\sum S_1 + \sum S_2 = \sum S$

$\sum S = 2 \sum S_1$

Therefore it should be even.

   b. (6 points) For the same set $A = \{5,7,9,10,22,35,60\}$ in Question 1 and knowing that the sum of the elements in A is equal to 148, find two subsets of A that solve the **Partition** problem.

find sets that sum to $\frac{148}{2} = 74$

$= \{9, 5, 60\}, \{7, 10, 22, 35\}$

S_1 S_2

c. (10 points) Prove that the **Partition** problem belongs to the class of NP problems.

To prove that it belongs to NP, we construct a nondeterministic algorithm that guesses and verifies a solution in polynomial time.

(1) Guess step: Non deterministic algorithm that guesses two sets of numbers from S. This is done in $O(|S|)$

(2) Verification step: Deterministic algorithm verifies the guess. makes sure that two sets of numbers were guessed, the two sets are disjoint and they together make up the entire set S. It then verifies that they both sum to the same number. This is done in polynomial time

Hence, Partition $\in$ NP.

d. (6 points) Consider the following set

$$B = \{5,7,9,10,22,35,60,\mathbf{51},\mathbf{99}\}$$

where **51 = 50 + 1** and **99 = 148 + 1 – 50**. Knowing that a solution to the sum of subsets problem of part 1 exists (actually you already should have found it), can you find a solution to the Partition problem when applied to the set *B* above? (Hint: What should the sum be equal to in this case?)

Total = 148 + 150 = 298

Therfore, total in each subset should be 298/2 = 149

$S_1 = \{ ,7,9, ,22,51,60\}$ $S_2 = \{5,10,35,99\}$

3. (20 points) Given that the sum of subsets problem is NP-Complete, find a polynomial time reduction from the sum of subsets problem to the Partition problem. Here, you need to prove two things
   a. (14 points) The transformation proposed by you is indeed a reduction.
   b. (6 points) The reduction can be done in polynomial time.
   Hint: The information in part 2(d) above is essential!

Given an instance of sum of subsets where there is a set A of n integers and integer M, and an integer $C = \sum_{a_i \in A} a_i$ = sum of elements in A.
Create a new set $B = A \cup \{M+1, C+1-M\}$
This set B represents an instance of Partition problem. This reduction is done in polynomial time.
We need to prove that if set A has a subset of size M, B can be partitioned into two equal sub arrays, and vice versa.

$\Rightarrow$
If A has a subset of size M, then set A contains $A = X \cup Y$ where $\sum X = M$ and Y could be an empty array.
Set B will contain $B = X \cup Y \cup \{M+1, C+1-M\}$
where $C = \sum X + \sum Y = M + \sum Y$. Thus set can be partitioned into $S_1 = Y \cup \{M+1\}$, $S_2 = X \cup \{C+1-M\}$

$\Leftarrow$
If set B can be partitioned into two sets S_1, S_2, then to be balanced, $\{C+1-M\}$ has to be in a set with elements that sum up to M.
Therefore there exists a subset of size M.

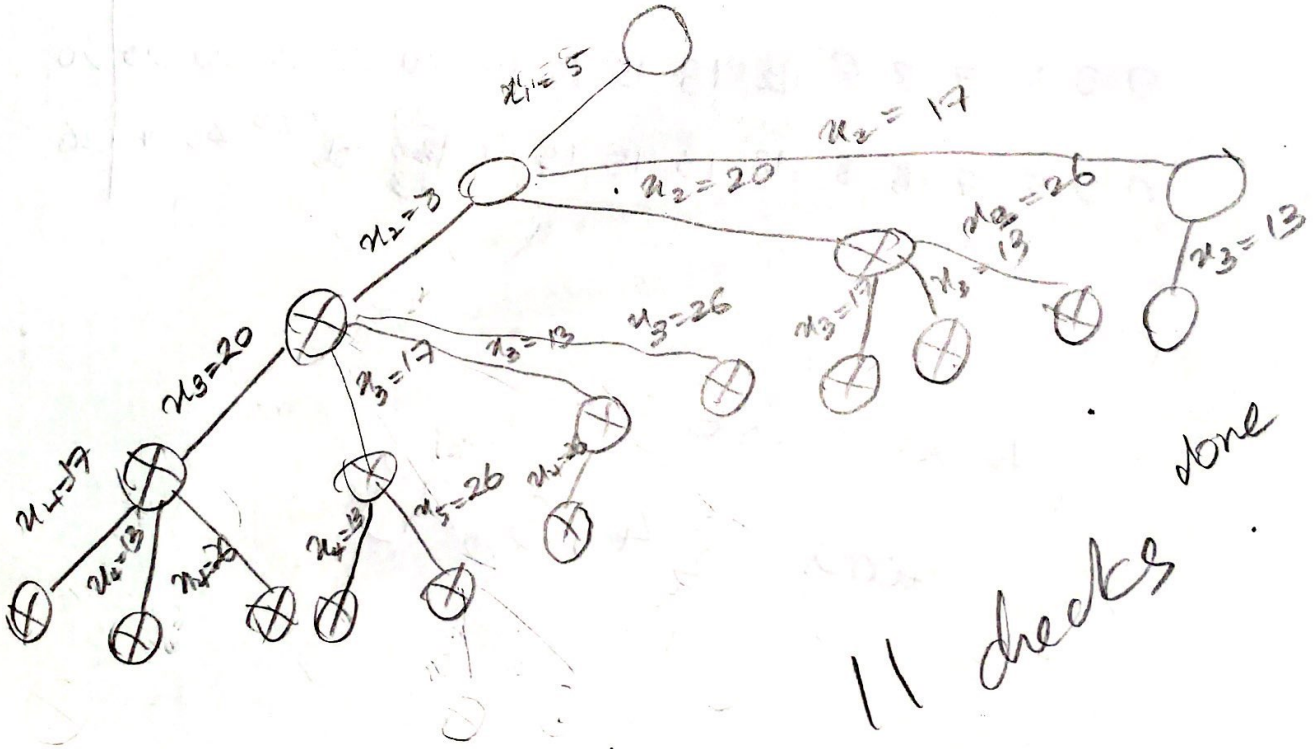
Question IV (30 points): Backtracking Algorithms.

1. (5 points) Clearly explain how backtracking algorithms are useful.

Back tracking algorithms help us reduce the search space by eliminating incorrect subtrees. It also provides a way for reverting incorrect decisions with minimal cost.

2. (25 points) Given a set of n +ve integers X = {5, 8, 20, 17, 13, 26} and an integer 35.
   a. (20 points) Use backtracking to answer the question of whether there is a subset of X of integers such that their sum is equal to 35. Make sure you show the intermediate steps.
   b. (5 points) Compare the number of subsets you checked to the total number of subsets of X.

5 + 8 = 13
13 + 8 = 21
21 + 8 = 29



11 checks done

vs

64 checks.

Q2. (15 points) Run the dynamic programming solution of the knapsack problem on a knapsack of size 17 with the following items:

Item	Size	Value
1	3	5
2	4	7
3	5	8
4	8	11

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	0	0	0	5	7	7	7	12	12	12	12	12	12	12	12	12	12	12
3	0	0	0	5	7	8	8	12	13	15	15	15	20	20	20	20	20	20
4	0	0	0	5	7	8	8	12	13	15	15	16	20	20	20	23	24	26

Max = 26

items: {4, 3, 2}

Q5. (20 points) Longest Path problem: Given a weighted graph G = (V,E), two distinguished vertices s, t ∈ V and a positive integer k, is there a simple path in G from s to t of length k or more?

Prove that the Longest Path problem is NP-Complete. (Hint: reduce the Hamiltonian Path problem to the Longest Path Problem)

first, we need to prove that Longest path problem is NP. We do this by specifying a nondeterministic algorithm that guesses and verifies its solution in polynomial time.

Guessing step: Non deterministic algorithm guesses a path of vertices that start at s and end at t. This is done in $O(n)$ time.

Verification step: A deterministic algorithm verifies the guess. it makes sure that the path starts at s and ends at t. if makes sure that every adjacent vertex in the guess is connected by an edge, this costs $O(|E|)$, lastly it makes sure that length of path is $\leq \geq k$.
This is done in polynomial time.

Hence, Longest path ∈ NP.

To prove that Longest path is NP-complete, we reduce Hamiltonian Cycle to it.

an instance of HC is an undirected graph $G=(V,E)$ we create a weighted graph $G'=(V,E')$ and s and t where $s \in V$, $t \in V$ and $s \neq t$.

these are two arbitrary vertices.

E' contains all edges in E, however, the edges are given a length of 1.

$k = |V|-1$.

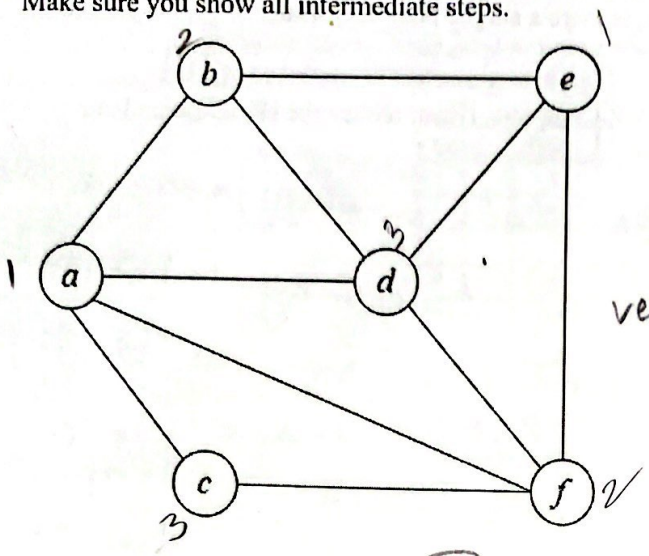
Therefore G', s, t, k are an instance of Longest path problem.

we need to prove that G has HC $\iff$ G' has a path between s and t of length at least k.

$\Rightarrow$ if G has an HC, the cycle visits $|V|$ vertices. the number of edges are $|V|-1$, therefore G' would have a path of length $k = |V|-1$.

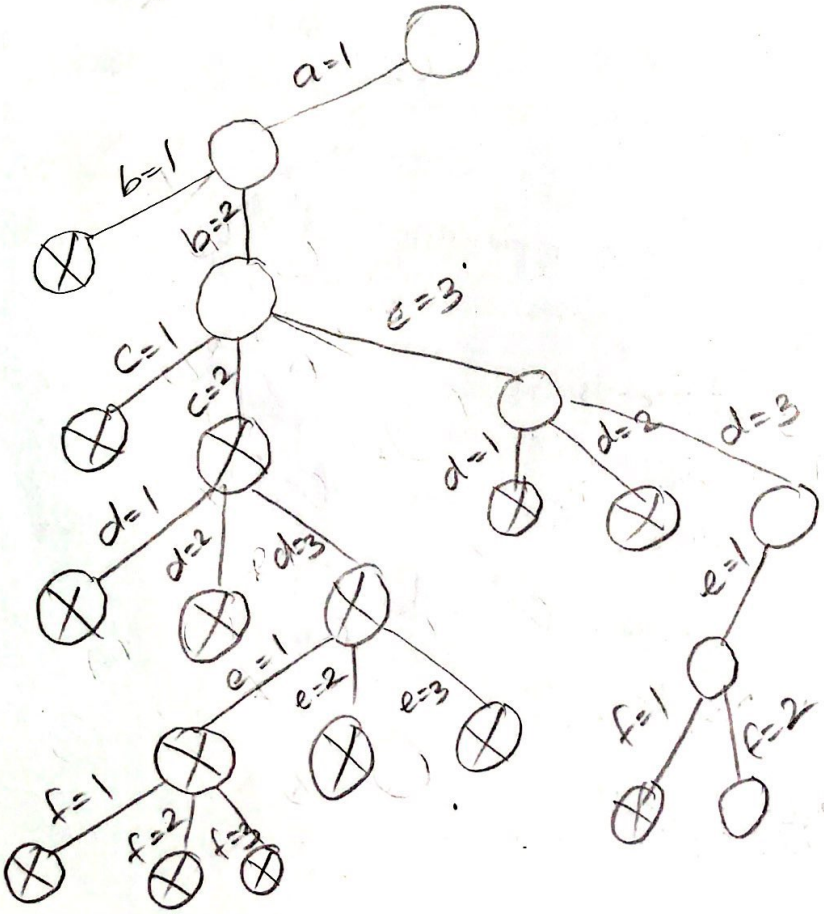
$\Leftarrow$ if G' has a path of length $k = |V|-1$, since it is a simple path, vertices are not repeated. therefore the number of distinct vertices covered in path is $|V|$. Hence, G has an HC.

Q6. (15 points) Apply the backtracking algorithm for the 3-coloring problem on the graph below. Make sure you show all intermediate steps.



colors: 1, 2, 3

vertices: a, b, c, d, e, f



(a=1, b=2, c=3, d=3; e=1, f=2)

الاجوبة ليست مؤكدة
لا تنسوني في دعائكم
بالتوفيق