19

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Analysis and Design of Algorithms

CS2102
Analysis of Algorithms
2020-II

Submission deadline: 23 December, 08:10

- Write your answers (images) inside the *answers* folder in order to generate a single PDF file. Replace the image files that are already included in the project. Do not change the file name.
- Read the questions carefully and write your answers clearly. Answers that are not legible and that doesn't follow the format will not have any score.

Outcomes:

a. Apply appropriate mathematical and related knowledge to computer science.

b. Analyze problems and identify the appropriate computational requirements for its solution.

Problem 1 (Outcome b) - 3 points

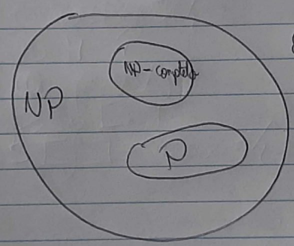
Explain what is an NP-complete problem and how can you determine that a problem is NP-complete in terms of reductions? Define the Independent Set and Vertex Cover problems and explain the reduction between them.

(1) Para que un problema cualquiera sea NP-completo debe cumplir lo sgte:

- Ser NP: Debe existir un verificador de su solución en tiempo polinomial, lo cual no significa que la solución como tal sea hallada en tiempo polinomial.

- Reducción: Todo problema $\in NP$ puede ser reducido al problema en cuestión (M)

$$\forall x \in NP \quad x \leq_p M$$



El verificador significa pertenecer al conjunto más grande (NP)

· Independent set: Dado un grafo y una restricción se retorna un subconjunto de vértices que cumplan la condición y ninguno sea adyacente. Tiene verificador en tiempo polinomial $\therefore \in NP$

· Vertex cover: Dado un grafo, su vertex cover es un subconjunto de vértices tal que para cada eje se incluya al menos uno de sus endpoints

alpha

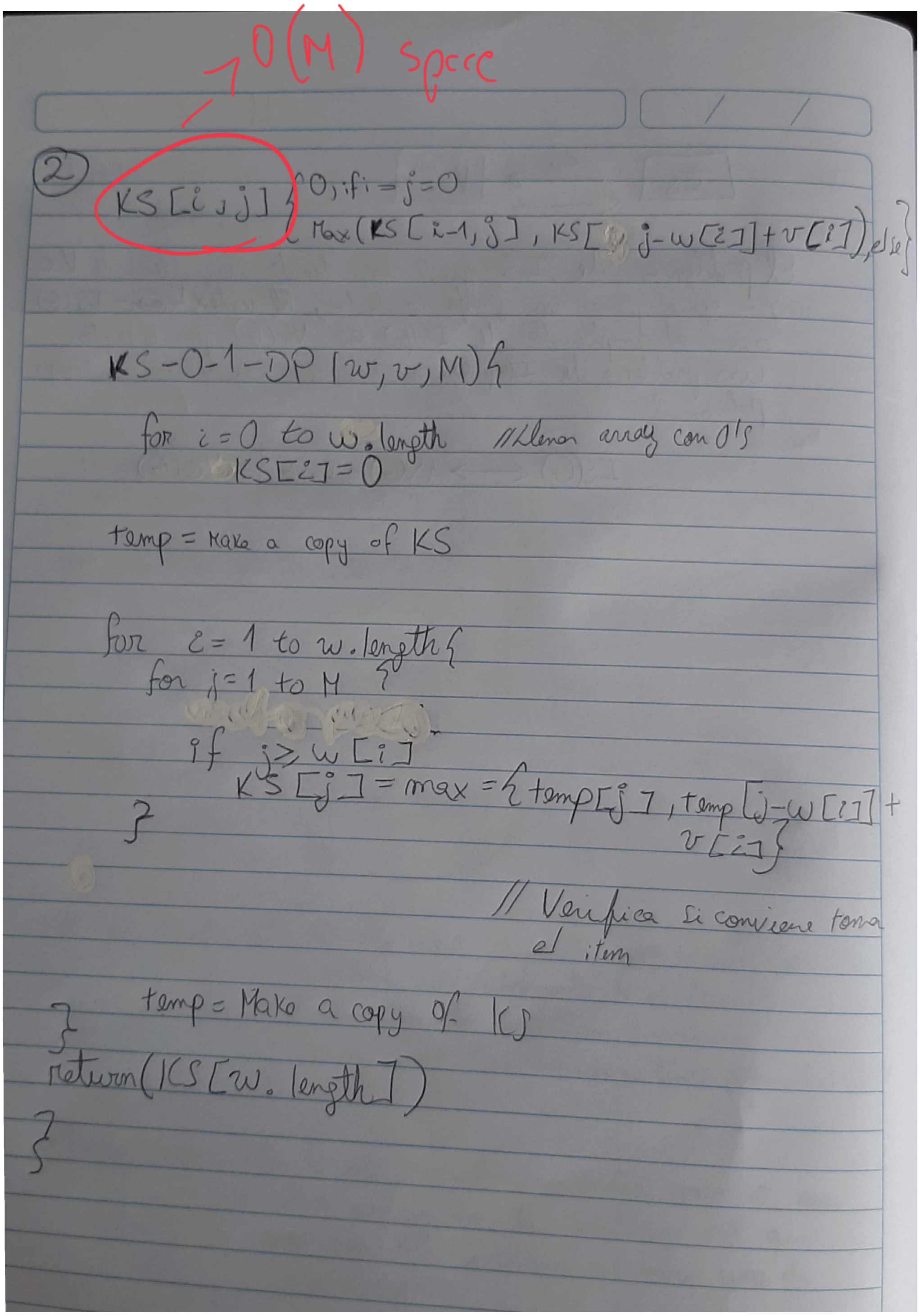
Problem 2 (Outcomes a, b) - 5 points

- Consider the 0-1 knapsack problem variation to write the recurrence relation and the pseudocode of an algorithm using Dynamic Programming. The algorithm should receive the n items (with its respective weights and values) and a maximum capacity M and return the maximum value supported by the bag. Your algorithm should have a complexity of O(n*M) and be restricted to a space consumption of O(M).

5/

- Based on the algorithm written above construct the dynamic programming table using the following information assuming that $M = 15$.

Item	1	2	3	4	5	6	7
Weight (kg)	3	8	6	2	1	9	10
Value ($)	1	7	8	4	2	5	3



O(M) space

```
(2) KS[i,j] { 0, if i=j=0
            { Max(KS[i-1,j], KS[ j-w[i]]+v[i]), else }

KS-0-1-DP(w,v,M){
    for i=0 to w.length    //Llenar array con 0's
        KS[i]=0
    temp = Make a copy of KS

    for i=1 to w.length{
        for j=1 to M {
            if j>=w[i]
                KS[j] = max = {temp[j], temp[j-w[i]] +
                                v[i]}
        }
                        // Verifica si conviene tomar
                           el item
        temp = Make a copy of KS
    }
    return(KS[w.length])
}
```

Problem 3 (Outcomes a, b) - 4 points

Given a rod of n meters and an array that contains prices of all pieces of size smaller than n:

3

- Write the recurrence relation and the pseudocode of an algorithm using dynamic programming to find the optimal way to cut the rod into smaller rods in order to maximize the profit.
- Consider the information below and construct the dynamic programming table to find the optimal way to cut the rod and maximize profit. Write your answer clearly.

Length (i)	1	2	3	4	5
Price (p_i)	1	5	8	9	10

3

$$cut(n) = \max_{1 \le i \le n} \{ price[i] + cut(n-i) \}$$

```
cut(n){
  if n=0
    return 0

  p_max = -∞

  for i = 1 to n
    max = ( p-max , price[i] + cut(n-i))

  return p-max }
```

4

Problem 4 (Outcome b) - 4 points

Choose one of the topics presented during the Final Project presentations and write a brief and concise definition of the problem, applications, foundation and analysis of your selected algorithm.

(4) Semi definite programming applied to Max-cut

↳ La programación semidefinida consta de minimizar o maximizar una función lineal obteniendo una combinación de matrices simétricas que son positivas semidefinidas es decir que para una matriz A tal que $Z_i^T A Z_i \geq 0$ para todo $Z_i \in \mathbb{R}^n$ y $A \in \mathbb{R}^{n \times n}$.

4

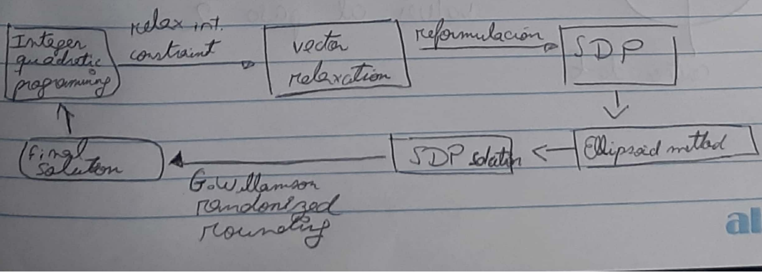
↳ El corte de un grafo consiste una bipartición de los vértices tal que $X \subseteq V(G)$ y el corte es $(X, V(G) \setminus X)$ y para $(u, v) \in E$ es corte si u y v están en diferentes partes de la bipartición. El tamaño de un graph-cut es la cantidad de edges con ese comportamiento (edge cut)

↳ Entonces el problema consiste en encontrar el corte de G que sea máximo, es decir, con mayor cantidad de edges cut. El problema es NP-Completo porque puede ser reducido a 3-SAT y por ende a SAT que ya se han probado como NP-completo

↳ Aplicaciones: Cualquier problema de optimización al que se le pueda dar la forma de Semi-definite programming

El método de aproximación de elipsoide usado para resolver este problema se usa en calculos de tamaño para transistores y cables; evaluación del potencial metabólico.

Approach general:



Problem 5 (Outcome b) - 4 points

A company produces two products: A and B. The production of each product of type A requires 3 hours to build and 1 hour to wrap up. The production of each product of type B requires 4 hours to build and 3 hours to wrap up. For building and wrapping up a product the maximum available hours are 60 and 30 respectively. The company makes a profit of \$8.000 on each item of product A and \$12.000 of each item of product B. How many items of product A and B should be produced to maximize the profit ? What is the maximum profit? Assume non-negative constraints and plot the inequalities highlighting the feasible region.

Remarks: Write all the operations of the row reduction process.

4

5 x_1: # producido en A
x_2: # producido en B

maximizar $8000x_1 + 12000x_2$

$3x_1 + 4x_2 \le 60$
$x_1 + 3x_2 \le 30$
$x_1 \ge 0$
$x_2 \ge 0$

K = 1000 como notación

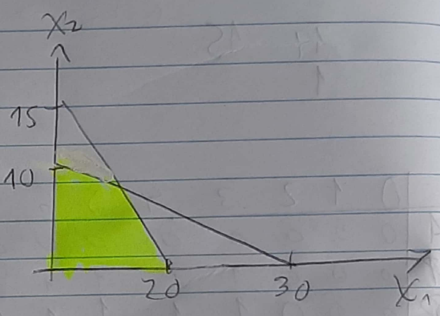
$$\begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & S_1 & S_2 & b \end{array} \\ \begin{bmatrix} 3 & 4 & 1 & 0 & 60 \\ 1 & 3 & 0 & 1 & 30 \\ -8000 & -12000 & 0 & 0 & 0 \end{bmatrix} \end{array} R_2 \times \frac{1}{2}$$

$$\begin{bmatrix} 3 & 4 & 1 & 0 & 60 \\ 1/3 & 1 & 0 & 1/3 & 10 \\ -8000 & -12000 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} -4R_2 + R_1 \\ \\ 12000R_2 + R_3 \end{array}$$

$$\begin{bmatrix} 5/3 & 0 & 1 & -4/3 & 20 \\ 1/3 & 1 & 0 & 1/3 & 10 \\ -4000 & 0 & 0 & 4000 & 120K \end{bmatrix} \frac{3}{5} R_1$$

$$\begin{bmatrix} 1 & 0 & 3/5 & -4/5 & 12 \\ 1/3 & 1 & 0 & 1/3 & 10 \\ -4000 & 0 & 0 & 4K & 120K \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 3/5 & -4/5 & 12 \\ 0 & 1 & -1/5 & 3/5 & 6 \\ 0 & 0 & 2.4K & 800 & 168K \end{bmatrix} \begin{array}{l} \\ R_1 \times (-\frac{1}{3}) + R_2 \\ R_1 \times (4000) + R_3 \end{array}$$



= feasible region

Producir 12 items A y 6 items B genera una ganancia máxima de 168,000.