Grado en Diseño y Desarrollo de Videojuegos
Algoritmos para Juegos
Práctica 5: Algoritmos Voraces

1.- Implementar una versión del algoritmo voraz para resolver el problema del tiempo mínimo de espera en el sistema, ordenando el vector de entrada de manera no decreciente.

2.- Implementar una versión del algoritmo voraz para resolver el problema de la mochila real.

3.- Dado un array de enteros positivos y un número *k*, implementar un algoritmo voraz que divida este array en dos subarrays tales que la diferencia entre las sumas de sus respectivos elementos sea máxima. Por ejemplo, si el array es [9, 4, 5, 2, 10] y *k* = 2, la salida debe ser 18, ya que los subarrays que maximizan la diferencias entre las sumas de sus elementos son [4, 2], de longitud k, y [9, 5, 10], de longitud N – k, y (9 + 5 + 10) - (4 + 2) = 18.

4.- Implementar un programa en Python que resuelva el problema del árbol de recubrimiento mínimo en un grafo utilizando el algoritmo de Kruskal

5.- Implementar un programa en Python que resuelva el problema del árbol de recubrimiento mínimo en un grafo utilizando el algoritmo de Prim.

6.- Implementar un programa en Python que calcule la longitud de los caminos mínimos en un grafo utilizando el algoritmo de Dijkstra.

7.- Modificar el programa anterior para que, además de las longitudes mínimas, calcule los caminos mínimos.

8.- El problema del viajante de comercio se puede enunciar del siguiente modo: *"Dado un conjunto de ciudades A={a_1,...,a_n}, y una matriz de distancias entre ellas C={c_{ij} dist entre a_i y a_j}, encontrar un recorrido de longitud mínima para un viajante que tiene que recorrer todas las ciudades y volver a la de partida"*. Se pide realizar e implementar en Python un algoritmo Voraz que resuelva este problema. Utilizar la instancia de la Tabla 1.

Tabla 1. Instancia para el problema del viajante de comercio.

	a_2	a_3	a_4	a_5	a_6
a_1	5	7.07	16.55	15.52	18
a_2		5	11.70	11.05	14.32
a_3			14	14.32	18.38
a_4				3	7.6
a_5					5

9.- Dado un tablero de ajedrez de tamaño *n*x*n* y una casilla inicial (*x*0,*y*0), queremos averiguar si es posible que un caballo recorra todos y cada uno de los escaques (casillas) sin pasar dos veces por el mismo. No es necesario en este problema que el caballo vuelva al escaque de partida. Un posible algoritmo ávido decide, en cada iteración, colocar el caballo en la casilla desde la cual domina el menor número posible de casillas aún no visitadas:

7.a) Implementar dicho algoritmo.

7.b) Utilizando el algoritmo realizado en el apartado anterior, buscar todas las casillas iniciales para los que el algoritmo encuentra solución.

10.- El problema de la máxima diversidad se puede enunciar como sigue: "*dado un conjunto de n elementos, seleccionar el subconjunto de m elementos que maximiza la suma de las distancias entre ellos*". Se pide realizar e implementar en Phyton un algoritmo Voraz que resuelva este problema. Utilizar la instancia de la Tabla 2.

Tabla 2. Instancia para el problema de la máxima diversidad.

	2	3	4	5	6	7	8	9	10
1	2.65	2.83	2.65	2.00	2.83	2.45	2.65	3.00	2.65
2		3.32	3.74	1.73	1.00	2.65	3.16	1.41	2.00
3			2.65	3.46	3.16	2.45	2.65	2.65	4.12
4				3.87	3.61	3.87	2.00	3.46	3.74
5					2.00	2.00	3.87	2.24	1.98
6						2.83	3.32	1.73	1.73
7							3.87	2.24	3.32
8								3.16	4.00
9									2.83

NOTA: el primer elemento que forma parte de la solución se elegirá aleatoriamente del conjunto de los candidatos.

11.- Supongamos que una sociedad en una isla se modela como un grafo, en el que los vértices son las poblaciones y las aristas representan las relaciones entre ellas. Por motivos de supervivencia, se debe migrar toda la población a una isla cercana. El modo deseado de hacerlo es minimizando el número de relaciones afectadas entre las poblaciones en la nueva localización y la antigua. La migración se llevará a cabo moviendo poblaciones de una en una, desde la isla origen a la destino. Se pide desarrollar un algoritmo voraz que planifique el orden en el que debe planificarse la migración.

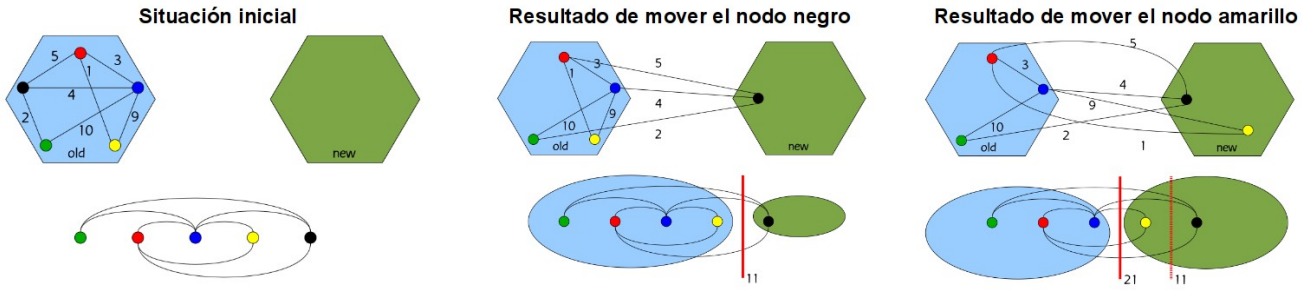

Fig 1. Migración de 5 poblaciones a una isla vecina.

12.- Consideramos el problema de la mochila modificado, consistente en dado un tablero de dimensiones M1 x M2 y n piezas de dimensiones i1 x j1, i2 x j2, ..., in x jn, cada una de ellas con beneficio b1, b2, ..., bn respectivamente, maximizar el beneficio de poner las piezas en el tablero teniendo en cuenta que las piezas se pueden separar en cuadriculas para ponerlas en el tablero y que el beneficio de poner una cuadrícula de una pieza de dimensión i x j con beneficio b es b/ij. Resolver el problema para el caso de un tablero 3x3 y piezas 2x3, 1x4, 3x1 y 2x2, con beneficios 6, 3, 4 y 2, respectivamente.