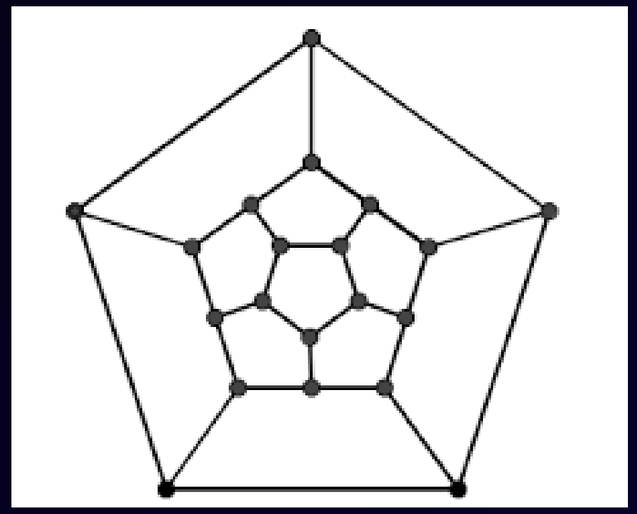
Algoritmos NP

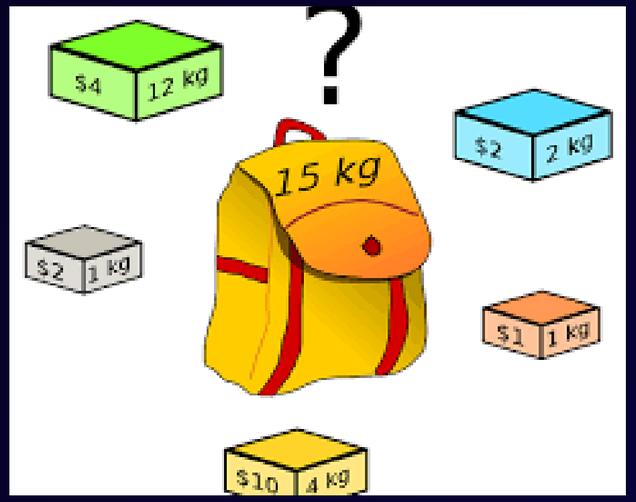
Problemas

Viajante de comercio (TSP)

Ciclo Hamiltoniano



Knapsack



Problema 1

Viajante de Comercio (TSP)

Cual es el problema?

Dado una cantidad de ciudades y sus distancias entre cada par de ellas, se busca una ruta que pase por todas las ciudades teniendo en cuenta el retorno

Solucion del Estudiante

1. En cada iteración, selecciona la ciudad más cercana que aún no ha sido visitada.
2. La elección se basa únicamente en minimizar la distancia local en cada paso, sin considerar cómo las decisiones actuales pueden afectar todo el recorrido.
3. Finalmente, regresa a la ciudad inicial desde la última ciudad visitada.

Tiempo de ejeccuion: O(n^2)

Tiempos y resultados SMALL (5)

Promedio de Tiempo:
0.0192 milisegundos

Tiempo (ms)	Resultado
0.0611 ms	8372.04 Km
0.0045 ms	8372.04 Km
0.0033 ms	8372.04 Km
0.0098 ms	8372.04 Km
0.0173 ms	8372.04 Km

Tiempos y resultados MEDIUM (10)

Promedio de Tiempo:
0.00634 milisegundos

Tiempo (ms)	Resultado
0.0061 ms	12018.31 Km
0.0085 ms	12018.31 Km
0.0064 ms	12018.31 Km
0.0058 ms	12018.31 Km
0.0049 ms	12018.31 Km

Tiempos y resultados LARGE (20)

Promedio de Tiempo: 0.03132 milisegundos

Tiempo (ms)	Resultado
0.0128 ms	14065.28 Km
0.0383 ms	12018.31 Km
0.0401 ms	14065.28 Km
0.0364 ms	14065.28 Km
0.0290 ms	14065.28 Km

Solucion de la comunidad – Held Karp

1. Crear la tabla memo y parent para almacenar los costos mínimos y las ciudades previas.
2. Inicializar los casos base donde no se han visitado ciudades o solo la ciudad 0 está visitada.
3. Iterar sobre todos los subconjuntos de ciudades y calcular el costo mínimo de visitar un conjunto de ciudades y terminar en una ciudad específica.
4. Buscar el costo mínimo de regresar a la ciudad 0 desde cualquier ciudad que haya sido visitada.
5. Reconstruir la ruta a partir de la tabla parent.

Tiempo de ejeccuion: O(n^2 x 2^n)

Tiempos y resultados SMALL (5)

Promedio de Tiempo:
0.9400 milisegundos

Tiempo (ms)	Resultado
1.3489 ms	8372.04 Km
0.6468 ms	8372.04 Km
0.8777 ms	8372.04 Km
1.0604 ms	8372.04 Km
0.7662 ms	8372.04 Km

Tiempos y resultados MEDIUM (10)

Tiempo (ms)	Resultado
18.0890 ms	10593.28 Km
1.2837 ms	10593.28 Km
0.8184 ms	10593.28 Km
0.7456 ms	10593.28 Km
0.7947 ms	10593.28 Km

Promedio de Tiempo:
4.34628 milisegundos

Tiempos y resultados LARGE (20)

Tiempo (ms)	Resultado
1469.2259 ms	13973.5 Km
1414.1665 ms	13973.5 Km
1490.9301 ms	13973.5 Km
1471.5966 ms	13973.5 Km
1442.8697 ms	13973.5 Km

Promedio de Tiempo: 1457.75776 milisegundos

Tiempos y resultados LARGE (20)

Tiempo (ms)	Resultado
1469.2259 ms	13973.5 Km
1414.1665 ms	13973.5 Km
1490.9301 ms	13973.5 Km
1471.5966 ms	13973.5 Km
1442.8697 ms	13973.5 Km

Promedio de Tiempo: 1457.75776 milisegundos

CONLUSIONES

Algoritmo Propuesto:

- Es eficiente en tiempos de ejecución
- Fácil de implementar.
- En problemas grandes, puede generar rutas ineficientes.

Algoritmo Propuesto:

- Encuentra la ruta más corta posible.
- Es más preciso en problemas grandes.
- Más complejo y costoso en términos computacionales.

Problema 2

Ciclo Hamiltoniano

Cual es el problema?

Dado un grafo y una posición inicial, se ocupa encontrar un camino en el que se pueda recorrer todos los nodos una sola vez, y regresar a la posición inicial.

Solucion del Estudiante

1. En cada recursión se le pone al nodo actual como visto. y se agrega a la lista del camino;
2. Se revisa si el tamaño del camino es igual a las cantidad de nodos en el grafo.
3. En un for, se recorre todas las aristas no vistas del nodo actual, y le hacemos recursividad.
4. Si sale del for, no hay una arista no vista, y retrocedemos.

Tiempo de ejeccuion: O(n!)

Tiempos y resultados SMALL (4x4)

Promedio de Tiempo:
0.38508 milisegundos

Posición inicial	Tiempo
A4	0.4554 ms
D4	0.2317 ms
A1	0.5894 ms
D1	0.3634 ms
B3	0.2855 ms

Tiempos y resultados MEDIUM (5x5)

Promedio de Tiempo:
69.0621 milisegundos

Posición inicial	Tiempo
A5	82.2860 ms
E5	81.6531 ms
A1	73.0758 ms
E1	78.3674 ms
C3	29.9282 ms

Tiempos y resultados LARGE (6x6)

Promedio de Tiempo: 10438.74006 milisegundos

Posición inicial	Tiempo
A6	13064.3192 ms
F6	170.9536 ms
A1	138.9290 ms
F1	38682.7505 ms
C4	136.7480 ms

Solucion de la comunidad

1. En cada recursión se compara si la copia actual de el camino tiene la misma cantidad de nodos que el grafo.
2. Si culpe la primera condición se mira si el ultimo nodo el path tiene como arista a el primer nodo del path.
3. En un for, se recorre todas las aristas no vistas del nodo actual, y le hacemos recursividad.
4. Si sale del for, no hay una arista no vista, y retrocedemos.

Tiempo de ejeccuion: O(n!)

Tiempos y resultados SMALL (4x4)

Promedio de Tiempo:
2.46152 milisegundos

Posición inicial	Tiempo
A4	4.4395 ms
D4	1.8892 ms
A1	2.1347 ms
D1	2.2579 ms
B3	1.5863 ms

Tiempos y resultados MEDIUM (5x5)

Promedio de Tiempo: 16614742 milisegundos

Posición inicial	Tiempo
A5	1889.3924 ms
E5	1883.1487 ms
A1	1836.6581 ms
E1	1989.1986 ms
C3	708.9732 ms

Tiempos y resultados LARGE (6x6)

Posición inicial	Tiempo
A6	232414.9424 ms
F6	1398.6351 ms
A1	996.8234 ms
F1	897487.5766 ms
C4	2101.8391 ms

Promedio de Tiempo: 226879.9634 milisegundos

CONLUSIONES

Algoritmo Propuesto:

- Es eficiente en tiempos de ejecución
- Es preciso en todas las situaciones.
- Puede ser mejorado
- En grafos mas grandes es mas ineficiente.

Algoritmo Propuesto:

- Es mas corto de hacer
- Es preciso en todas las situaciones
- Costoso en términos de tiempo y espacio en memoria.

Problema 1

La Mochila (Kanpsack)

Cual es el problema?

Dado una casa con objetos que tiene valor y peso, encontrar la mejor combinación en el que podemos maximizar el valor total sin excedernos del limite que tiene la mochila.

Solucion del Estudiante

1. Vamos a recorrer desde el ultimo ítem hasta el primero.
2. Vamos a considerar dos casos, si agarrar el objeto y si no agarrar el objeto, y los comparamos. (Para considerarlo, lo llamamos recursivamente e enviamos la posicion y el peso sobrante para cada caso)
3. Vamos a ir guardando los resultados del camino en un arreglo, para reutilizarlo y que sea mas rápido.

Tiempo de ejeccuion: O(n W)

Tiempos y resultados SMALL (5)

Promedio de Tiempo:
0.09236 milisegundos
Peso Máximo: 200

Tiempo (ms)	Resultado
0.3001 ms	$194
0.0344 ms	$194
0.0395 ms	$194
0.0414 ms	$194
0.0464 ms	$194

Tiempos y resultados MEDIUM (20)

Promedio de Tiempo:
0.0192 milisegundos
Peso Máximo: 500

Tiempo (ms)	Resultado
10.5567 ms	$791
6.4895 ms	$791
8.8186 ms	$791
5.8612 ms	$791
8.6210 ms	$791

Tiempos y resultados LARGE (50)

Promedio de Tiempo:
0.0192 milisegundos
Peso Máximo: 1000

Tiempo (ms)	Resultado
94.2204 ms	$1720
81.3521 ms	$1720
81.4704 ms	$1720
76.5278 ms	$1720
85.6160 ms	$1720

Solucion de la comunidad – Geeks

1. Crear una lista llamada dp que este va a ir guardando los valores máximos.
2. Se recorre la lista de objetos, empezando de 1
3. Se recorre de atrás, hacia 0, los pesos posibles.
4. Si el valor del de atrás, es menor al que podemos hacer con uno nuevo, entonces se agrega a dp
5. Una vez recorrido todo sumamos el valor de los items para tener el valor máximo.

Tiempo de ejeccuion: O(n W)

Tiempos y resultados SMALL (5)

Promedio de Tiempo:
0.28078 milisegundos
Peso Maximo: 200

Tiempo (ms)	Resultado
0.5779 ms	$194
0.4315 ms	$194
0.1162 ms	$194
0.1572 ms	$194
0.1211 ms	$194

Tiempos y resultados MEDIUM (20)

Promedio de Tiempo:
2.60702 milisegundos
Peso Máximo: 500

Tiempo (ms)	Resultado
2.6967 ms	$791
2.1624 ms	$791
3.1814 ms	$791
3.0252 ms	$791
1.9694 ms	$791

Tiempos y resultados LARGE (50)

Promedio de Tiempo:
0.0192 milisegundos
Peso Máximo: 1000

Tiempo (ms)	Resultado
17.5585 ms	$1720
11.7451 ms	$1720
6.4023 ms	$1720
6.8832 ms	$1720
9.0386 ms	$1720

CONLUSIONES

Algoritmo Propuesto:

- Es eficiente con pcoas cantidad de objetos
- Es preciso con su solución.
- Facil de entender y implementar.
- Se puede optimizar.

Algoritmo Propuesto:

- Complicado de entender.
- Devuelve la mejor respuesta
- Tiene un buen tiempo de ejecución en altas cantidades de objetos.

MUCHAS GRACIAS