TEORÍA DE ALGORITMOS 1

Guia de ejercicios: Backtracking y Branch and Bound.

Por: ING. VÍCTOR DANIEL PODBEREZSKI
vpodberezski@fi.uba.ar

Para cada ejercicio considere, además de resolver lo solicitado, calcular la complejidad temporal, espacial.

Referencias

- ⭐ Fácil
- ⭐⭐ Medio
- ⭐⭐⭐ Difícil

Enunciados

1. ⭐Contamos con un conjunto de “n” puntos (x,y) en el plano cartesiano. Un par de puntos es el más cercano si la distancia euclidiana entre ellos es menor a la de cualquier otro par. Resuelva el problema mediante un algoritmo naive que nos informe cuales son los 3 pares de puntos más cercanos.

2. ⭐Se encuentran en un río 3 caníbales y 3 vegetarianos. En la orilla hay un bote que permite pasar a dos personas atravesarlo. Los 6 quieren cruzar al otro lado. Sin embargo existe un peligro para los vegetarianos: si en algún momento en alguna de las márgenes hay más caníbales que vegetarianos estos los atacarán. Tener en cuenta que el bote tiene que ser manejado por alguien para regresar a la orilla. Determinar si es posible establecer un orden de cruces en el que puedan lograr su objetivo conservando la integridad física. Explicar cómo construir el grafo de estados del problema. Determinar cómo explorarlo para conseguir la respuesta al problema. Brinde, si existe, la respuesta al problema.

3. ⭐Resuelva el problema de las reinas en el tablero de ajedrez mediante backtracking planteado como permutaciones. Brinde el pseudocódigo y determine la cantidad máxima posible de subproblemas a explorar.

4. ⭐⭐En un tablero de ajedrez (una cuadrícula de 8x8) se ubica la pieza llamada “caballo” en la esquina superior izquierda. Un caballo tiene una manera peculiar de moverse por el tablero: Dos casillas en dirección horizontal o vertical y después una casilla más en ángulo recto (formando una forma similar a la letra “L”). El caballo se traslada de la casilla inicial a la final sin tocar las intermedias, dado que las “salta”. Se quiere determinar si es posible, mover esta pieza de forma sucesiva a través de todas las casillas del tablero, pasando una sola vez por cada una de ellas, y terminando en la casilla inicial. Plantear la solución mediante backtracking.

5. ⭐Se cuenta con “n” trabajos por realizar y la misma cantidad de contratistas para realizarlos. Ellos son capaces de realizar cualquiera de los trabajos aunque una vez que se comprometen a hacer uno, no podrán realizar el resto. Tenemos un presupuesto de cada trabajo por cada contratista. Queremos asignarlos de forma tal de minimizar el costo del trabajo total. Proponer un algoritmo por branch and bound que resuelva el problema de la asignación.

6. ⭐⭐Contamos con un conjunto de “n” actividades entre las que se puede optar por realizar. Cada actividad x tiene una fecha de inicio I_x, una fecha de finalización f_x y un valor v_x que obtenemos por realizarla. Queremos seleccionar el subconjunto de actividades compatibles entre sí que maximice la ganancia a obtener (suma de los valores del subconjunto). Proponer un algoritmo por branch and bound que resuelva el problema.

7. ⭐⭐Se cuenta con “n” servidores especializados en renderización de videos para películas animadas en 3d. Los servidores son exactamente iguales. Además contamos con “m” escenas de películas que se desean procesar. Cada escena tiene una duración determinada. Queremos determinar la asignación de las escenas a los servidores de modo tal de minimizar el tiempo a esperar hasta que la última de las escenas termine de procesarse. Determinar dos metodologías con la que pueda resolver el problema y presente cómo realizar el proceso.

8. ⭐Un granjero debe trasladar un lobo, una cabra y una zanahoria a la otra margen de un río. Cuenta con un bote donde solo entra él y un elemento más. El problema es que no puede quedar solo el lobo y la cabra. Dado que la primera se comería a la segunda. De igual manera, tampoco puede dejar solo a la cabra y la zanahoria. La primera no dudaría en comerse a la segunda. ¿Cómo puede hacer para cruzar? Explicar cómo construir el grafo de estados del problema. Determinar cómo explorarlo para conseguir la respuesta al problema. Brinde, si existe, la respuesta al problema.

9. ⭐⭐Contamos con una cuadrícula de n*n celdas. Cada celda puede pintarse de 2 colores: blanco o negro. Por cada fila (y cada columna) nos indican cuantas celdas hay que pintar de negro. Debemos obtener todas las coloraciones válidas que cumplan con las instrucciones. Resolver mediante Backtracking.

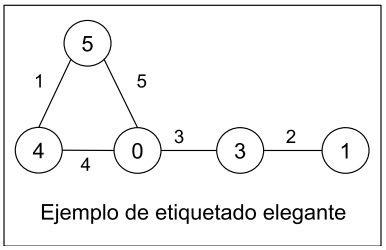
10. ⭐⭐En una variante del problema de la mochila, tenemos “n” elementos que podemos incluir dentro de un contenedor que acepta un total de “K” kilos. Cada elemento tiene un peso, un valor y un subconjunto de otros elementos con el que es incompatible seleccionarlo. Debemos seleccionar la combinación de elementos que sume el mayor valor posible sin incumplir las restricciones. En caso de existir diferentes soluciones máximas se prefiere a aquella que requiere un menor peso. Resolver por branch and bound.

11. ⭐⭐El problema del coloreo de grafos intenta asignar colores a los nodos de un grafo G=(V,E) de tal forma que dos nodos adyacentes no tengan el mismo color. Un parámetro de este problema es “c” la cantidad de colores a utilizar. El polinomio cromático es una función f(G,c) que determina cuántas coloraciones (soluciones al problema de coloreo) válidas diferentes pueden realizarse en un determinado grafo G utilizando exactamente “c” colores. Se pide generar mediante backtracking los primeros “c” resultados de f(G,c) para un determinado grafo G.

Ejercicios extra: generar y probar

1. ⭐Un cuadrado mágico de tamaño “n” es una disposición de los números enteros desde 1 a n^2 en una matriz de nxn que cumple la siguiente condiciones. Cada número aparece solo una vez. La suma de cada fila, columna y diagonal principal da el mismo valor. Proponer un algoritmo por generar y probar que dado un valor “n” calcule, si existe, un cuadrado mágico de ese tamaño.

2. ⭐En la teoría de gráfos, se conoce como etiquetado de vértices a asignarle a cada vértice una etiqueta diferente. De igual manera se puede realizar un etiquetado de ejes. Generalmente el etiquetado se puede representar mediante un número entero. Existen diferentes etiquetados posibles. Trabajaremos con el “etiquetado elegante” (graceful labeling). Dado un grafo G=(V,E) con m ejes se asignará como etiqueta a cada uno de sus vértices un número entre 0 y m. Se computará para cada eje la diferencia absoluta entre las etiquetas de vértices y esa será su etiqueta. Se espera que los ejes queden etiquetados del 1 al m inclusive (y que obviamente cada una sea única). Construya mediante generar y probar un algoritmo que dado un grafo determine un etiquetado elegante (si es posible).



Ejemplo de etiquetado elegante

3. ⭐Un ciclo euleriano en un grafo es un camino que pasa por cada arista una y solo una vez, comenzando por un vértice y finalizando en el mismo. Sea un grafo G=(V,E) se busca

generar si es posible un ciclo euleriano. Resolverlo mediante generar y probar.

Ejercicios resueltos

Cada ejercicio resuelto busca mostrar cómo se debe analizar, resolver y justificar la resolución del ejercicio, tanto en un trabajo práctico o en un parcial

1. ⭐ Contamos con un conjunto de “n” de equipamientos que se deben repartir entre un conjunto de “m” equipos de desarrollo. Cada equipo solicita un subconjunto de ellas. Puede ocurrir que un mismo equipamiento lo soliciten varios equipos o incluso que un equipamiento no lo solicite nadie. Queremos determinar si es posible seleccionar un subconjunto de equipos de desarrollo entregándoles a todos ellos todo lo que soliciten y al mismo tiempo que ninguno de los equipamientos quede sin repartir. Resolver el problema mediante backtracking.

Explicación de la solución

Para solucionar el problema con la técnica de Backtracking se plantea iterar las diferentes equipaciones (que son las que deben estar completamente asignas), y a su vez iterar los equipos disponibles para asignar cada equipación. La recursividad con backtracking se da al asignar una equipación a un equipo particular, donde se ejecutará la recursión para las equipaciones restantes. De esta manera, se estarán probando las diferentes combinaciones para encontrar la solución final.

Una vez asignadas (o no), las equipaciones, se realizarán los chequeos para comprobar si la solución es válida o no, donde se verifica que las equipaciones fueron totalmente asignadas y los equipos que fueron asignados no están incompletos (poseen todas las equipaciones que necesitaban)

Si se halla una solución válida durante la ejecución del programa, se devuelve la misma. La propiedad de corte puede distinguirse en esta sentencia.

Pseudocódigo

```python
Backtracking (equipamientos, equipos)
	for equipamiento in equipamientos:
		quito equipamiento de equipamientos

		for equipo in equipos:
			# cubro la posibilidad de asignar el equipamiento al
equipo
			if equipo requiere equipamiento:
				asigno equipamiento a equipo

				es_valido, solucion = Backtracking (equipamientos,
				equipos)
				if es_valido:
					return solucion
				else:
					des-asigno equipamiento a equipo
			# la posibilidad de NO asignar el equipamiento al equipo
			y ser asignado a otro equipo será cubierta por el mismo
			bucle de equipos

	if equipamientos.len != 0:
		return "solución inválida" # no pueden haber equipamientos sin
	asignarse
	solucion = {}
	for equipo in equipos:
		if equipo.parcialmente_completo:
			return "solución inválida" # no pueden haber equipos
incompletos
		else if equipo.completo:
			solucion.add(equipo)
	return "solución válida" (solucion)
```

Análisis de complejidad

La complejidad temporal del algoritmo planteado es O(N * E^N), donde E representa la cantidad equipaciones y N la cantidad de equipos. En el peor de los casos, se iteran todas las combinaciones de equipaciones por cada equipo (E^N), pero además por cada iteración se recorren los equipos para verificar si la solución fue válida o no (O(N)). Las demás operaciones que se realizan dentro de cada iteración, tales como la asignación de las equipaciones a equipos o desaginaciones de las equipaciones se realizan en O(1) y no afectan la complejidad de la solución.