RutaMinima

4. Dada una matriz D de $n \times n$ números naturales, queremos encontrar una permutación π[1] de $\{1, \dots, n\}$ que minimice $D_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} D_{\pi(i)\pi(i+1)}$. Por ejemplo, si

$$D = \begin{pmatrix} 0 & 1 & 10 & 10 \\ 10 & 0 & 3 & 15 \\ 21 & 17 & 0 & 2 \\ 3 & 22 & 30 & 0 \end{pmatrix},$$

entonces $\pi(i) = i$ es una solución optima.

   a) Diseñar un algoritmo de *backtracking* para resolver el problema, indicando claramente cómo se codifica una solución candidata, cuáles soluciones son válidas y qué valor tienen, qué es una solución parcial y cómo se extiende cada solución parcial.
   b) Calcular la complejidad temporal y espacial del mismo.
   c) Proponer una poda por optimalidad y mostrar que es correcta.

[1] Una permutacion de un conjunto finito X es simplemente una función biyectiva de X en X.

Vamos a "traducir" un poco nuestro enunciado

* Una permutación π de $\{1, \dots, n\}$ dado un número de ese conjunto devuelve el que pasa a estar en ese lugar
  → si tenemos $\{1,2,3\}$, $\pi(1)=3$ → $\{3,2,1\}$

- En este caso vamos a trabajar con permutaciones de filas de D

$D_{\pi(n)\pi(1)}$ = Elemento en la n-ésima fila 1ra columna
$D_{\pi(i)\pi(i+1)}$ = " " " i-ésima fila i+1-ésima columna → la diagonal arriba de la diagonal

→ En el caso del ejemplo la permutación es $\pi(i)=i$, es decir, dejar las filas como estaban

Resolución del ej

ⓐ Solución candidata: cualquier conjunto de n elementos → vamos a devolver un vector donde los números de cada posición indica qué fila va a ir a parar a cada por de la matriz. Es decir, $v[i] = \pi(i)$.

Soluciones válidas: los conjuntos de n elementos distintos / minimizan $D_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} D_{\pi(i)\pi(i+1)}$

Soluciones parciales: conjuntos de k elementos con $1 \leq k \leq n$
↪ van a ser vectores donde vayamos poniendo las permutaciones

$< \underbrace{\_}_{\text{fijo}} \ \underbrace{\_}_{\text{exploro}} \ \ldots >$

fijo

¿Cuál es el caso base? Cuando hayamos explorado todas las permutaciones posibles.

ÁRBOL DE BT SOBRE ES

{}

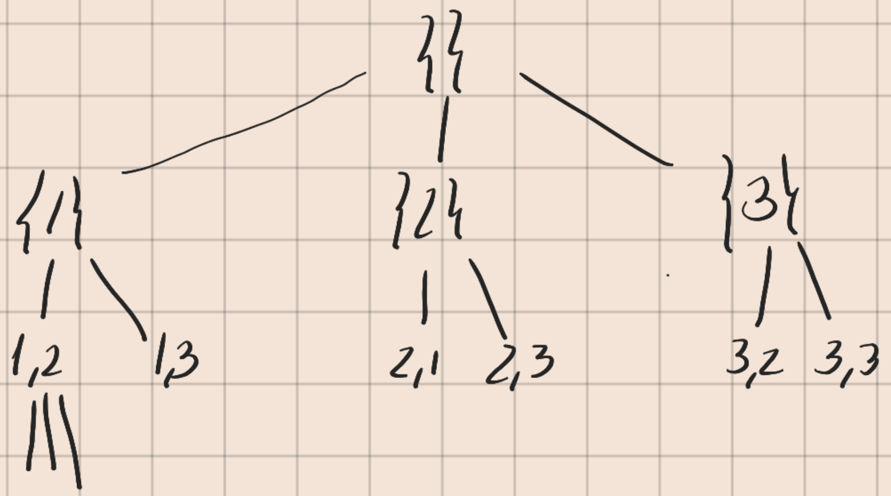
{1} {2} {3} {4}

{1,1} {1,2} {1,3} {1,4} {2,1} {2,2} {2,3} {2,4} 3,1 3,2 3,3 3,4 4,1 4,2 4,3 4,4

Vamos a llamar f a la función a optimizar

$$BT(D, i, \{1,\ldots,n\}, per) = \begin{cases} per & i = N+1 \wedge \text{minimiza } f(per) \\ \text{recursión c/todas las permutaciones para elegir al 1}^{\text{er}} \text{ elemento} & \end{cases}$$

Sea $D = \begin{pmatrix} 0 & 0 & 1 \\ 150 & 0 & 0 \\ 0 & 20 & 0 \end{pmatrix}$ $\pi(i) = i$ minimiza



```
vector per = {}
int min = INF
BT(D, i, suma, conj)
  if i = N+1 then  ∧ tiene Elementos Distintos (conj)
     if suma < min then
        per = conj;
        min = suma
     endif
  else
     ~~for j = 1 to D.size~~
        for k = 1 to D.size
           conj.pushback(k)
           BT( D, i+1, suma + D[k][(k+1)% D.size()], conj )
           conj.pop back(k)
```

NO queremos estos casos como soluciones

no termina nunca con un doble ciclo. Es un único ciclo donde j=i así vamos añadiendo instancias

} for j=i to D.size así entonces también tienes con elementos repetidos ✗ si hacemos j=1 perdemos combinaciones, hacemos j=0 ✓

El ciclo lo corta el caso base

ya si j=i → en la recursión va a quedar j=2

Si j=i para cuando j=3 → vamos a perder la pos de explorar el caso {3,1}

ⓑ Complejidad espacial = espacio que termina ocupando p → $\Theta(n)$
Complejidad temporal = # nodos = # permutaciones de un conj de n elementos = $\Theta(n!)$
1^{er} nivel n nodos
2^{do} " $n-1$ nodos
y así

ⓒ Poda x optimalidad: si la suma parcial de la permutación ya es mayor que la mínima guardada no tiene sentido completar esa solución.
Veamos que es correcto

Sup que tenemos $min = x$ y estamos explorando la sol $(v_1, \dots, v_k)$ con $k<n$ y $f(v_1, \dots, v_k) > x$. Si cortamos acá evitamos explorar los $(k-1)$ hijos de ese nodo, hijos que llevan a respuestas NO óptimas porque su suma ya supera la mínima hallada. Al no explorar esos $(k-1)$ hijos evito explorar $(k-1)!$ nodos.
Estoy optimizando mi construcción de soluciones.