miércoles, 13 de noviembre de 2024 09:02

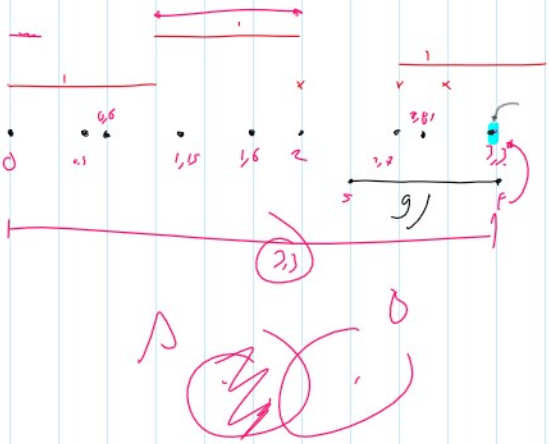
Ejercicio 1. Describa un algoritmo eficiente que, dado un conjunto $\{a_1, a_2, \ldots, a_n\}$ de puntos en la recta, determine un conjunto mínimo de intervalos de tamaño 1 que contiene a todos los puntos. Justifique que su algoritmo es correcto usando la propiedades de elección voraz y subestructura óptima. Puede suponer que $a_1 \leq a_2 \leq \cdots \leq a_n$.

Q1: $A = \langle a_1, \ldots, a_n \rangle$ set puntos. // $a_1 \leq \cdots \leq a_n$

Q2: $\mathcal{I} = \{$ intervalos $\}$ Tg $|I|=1 \wedge A \subseteq \bigcup_{I \in \mathcal{I}} I$

Q3: $|\mathcal{I}|$ mínima

Q4 $g = [a_n - 1, a_n]$ ←



Q5:
$$\begin{aligned} A' &= A - g \\ &= A - [a_n - 1, a_n] \\ &= A - \langle a_i \in A \mid a_i \in [a_n - 1, a_n] \rangle \\ &= A - \langle a_i \in A \mid a_n - 1 \leq a_i \leq a_n \rangle \\ &= A - \langle a_i \in A \mid a_i \geq a_n - 1 \rangle \\ &= \langle a_i \in A : a_i < a_n - 1 \rangle \end{aligned}$$

= Todos los puntos de A que no pertenezca al intervalo de g.c.

$Alg_0(A)$

if $(|A| == 0)$ return $\emptyset$

else

$g = [a_n - 1, a_n]$

$A' = \langle\rangle$

for $i = 1$ To n

if $(a_i < a_n - 1)$ $A'.pb(a_i)$

return $\{g\} \cup Alg_0(A')$.

G.C.P.

$\exists$ U opt sol' Tg contg al greedy

$\exists$ U opt sol Tg $g \in U$.

Prueba:

. Sea X una sol opt cualquiera. Hay 2 casos

– Caso 1: $g \in X$ Facilito $U = X$

– Caso 2: $g \notin X$ [s,f)

- Caso 1: $g \in X$. Trivial $U = X$
- Caso 2: $g \notin X$

Como X es sol del problema X contiene un intervalo J que cubre a a_n.

Además, como X no contiene al greedy choice, los extremos de J están estrictamente a la derecha de los de g.

Construimos X' como el resultado de intercambiar los objetos J y g en X.

- Claramente g es elemento de X'
- X' es un conjunto de intervalos unitarios
- Al retirar J de X, hemos descubierto algunos puntos de A, pero por lo visto arriba, como g es un desplazamiento más a la izquierda de J, entonces todos los puntos -y posiblemente más- seguirían siendo cubiertos por g. Con esto concluimos que X' sigue cubriendo a A.

$\Rightarrow \exists J \in X$ Tq $a_n \in J = [s, f]$ (con $[s,f)$) $\rightarrow s \le a_n \le f \wedge f > 1$

También tenemos: $J \neq g$

$\Rightarrow s > a_{n-1} \wedge f > a_n$

Sea $X' = X - \{J\} \cup \{g\}$

$\Rightarrow$

- $g \in X'$
- $I \in X' \rightarrow |I| = 1$
- Sea $p \in A \cap J$ gusto $\rightarrow p \in g$.

$\Rightarrow s \le p \le f$

$p \in A \rightarrow p \le a_n$ ✓

$p \ge s = f - 1 > a_n - 1 \rightarrow p > a_{n-1}$

$\Rightarrow a_{n-1} \le p \le a_n \Leftrightarrow p \in A \cap g$.

$a_{n-1} \le p \le a_n$ ↑ $p \in g$.

$\Rightarrow A \subseteq \bigcup_{I \in X} I$ ✓

O.S.P

- X sol opt $P(A)$ que contiene a g
- $X' = X$ sin g . A' subproblema

$\Rightarrow X'$ sol opt $P(A')$

X sol op $P(A)$ Tq $g \in X$

- $X' = X - \{g\}$ y $A' = A - \{a_i : a_i \in g\}$

$\Rightarrow X'$ sol opt $P(A')$

Probar:

- Supongo x contradicción que X' no es opt. de $P(A')$
- $\exists Y'$ sí o opt $P(A')$ $\leftarrow |Y'| < |X'|$
- Sea $Y = Y' \cup \{g\}$.
- Como Y' cubre A' y g cubre $g \Rightarrow Y$ cubre $A' \cup g$ que a su vez cubre A.

$\Rightarrow Y$ es una sol.

Además: $|Y| = |Y'| + 1 < |X'| + 1 = |X|$

Esto implica que X nunca pudo ser opt!! T.T. ($\Rightarrow \Leftarrow$).

Además: $|Y| = |Y'| + 1 < |X'| + 1 = |X|$

Esto implica que X no era pue opt !! T.T. ($\Rightarrow \Leftarrow$).

Ejercicio 2. Dadas dos secuencias A y B, cada una de las cuales consiste en n enteros positivos, un *cruce* entre A y B es un conjunto de pares ordenados $\{(a_i, b_j) : a_i \in A, b_j \in B\}$, tales que todo elemento en A aparece exactamente una vez, y todo elemento en B aparece exactamente una vez. La *ganancia* de un cruce X es $\prod_{(a_i,b_j)\in X} a_i^{b_j}$. Diseñe un algoritmo voraz que maximiza la ganancia de un cruce. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.

Q_1: A, B : $|A| = n = |B|$ de enteros positivos.

Q_2: Un cruce $X = \{(a_i, b_j) : a_i \in A, b_j \in B\}$ ✓

Q_3: Max $gan(X) = \prod_{(a_i,b_j)\in X} a_i^{b_j}$

Ej: $A = \{1, 3, 4, 6, 7\}$, $B = \{3, 3, 5, 6, 10\}$

$6^{10} \times 7^{7} = 6^{6} \times 7^{10}$

Ej: (a_n, b_n), $a_1 \leq \dots \leq a_n$, $b_1 \leq \dots \leq b_n$

Q_{IH}: $A[1:n-1]$, $B[1:n-1]$

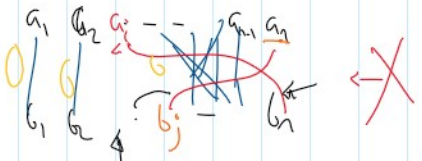
G. C. P

$\exists X$ sol opt Tq $(a_n, b_n) \in X$

Sea X un opt arbitrario

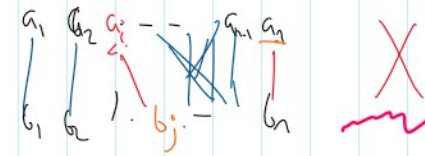
Caso 1: $(a_n, b_n) \in X$ ✓

Caso 2: $(a_n, b_n) \notin X$ ✗



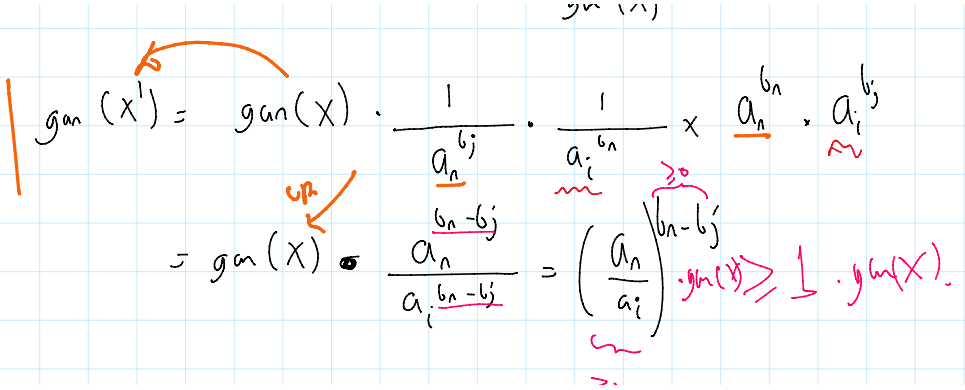
Como X es un cruce, y a_n y b_n no están emparejados

$\Rightarrow \exists\, i, j < n$ Tq $(a_n, b_j), (a_i, b_n) \in X$



$X' = X - \{(a_n, b_j), (a_i, b_n)\} \cup \{(a_n, b_n), (a_i, b_j)\}$

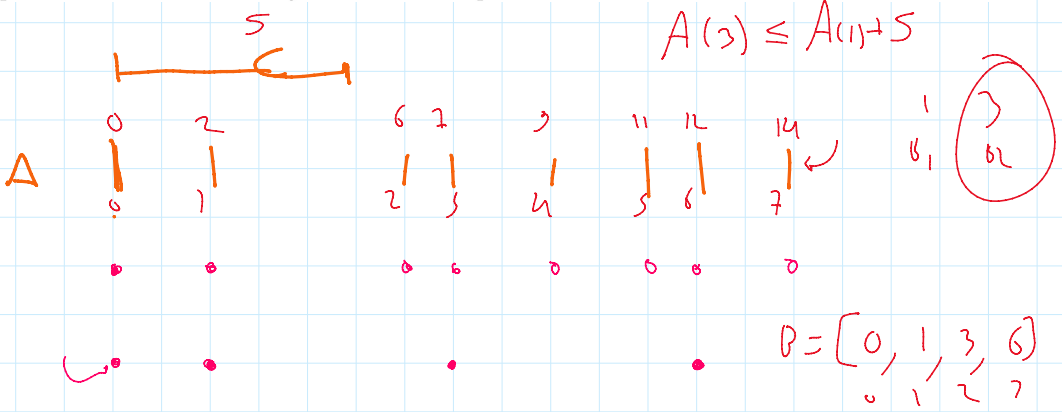
- $g \in X'$ ✓
- X' es un cruce ✓
- ¿ $gan(X') \geq gan(X)$? $\Leftrightarrow$ Si $\frac{gan(X')}{gan(X)} \geq 1$?



Ejercicio 3. Quiero dirigir un carro de una ciudad a otra a lo largo de una carretera. El tanque de combustible del carro tiene capacidad suficiente para cubrir c kilómetros, El mapa de la carretera indica la localización de los puestos de combustible. Queremos encontrar un algoritmo que garantize el viaje con el menor número de abastecimientos.

Más formalmente, usted recibe un arreglo $A[0..n]$ de números reales. El primer punto es el punto de partida, que además es un punto de recarga. Cada uno de los siguientes $n-1$ números indica los puntos de posibilidad de recarga y el último número indica el lugar de destino. Además recibe un número c. Usted empieza en el punto con coordenada $A[0]$. Debe encontrar un arreglo ordenado de índices $B[0..k]$ con el menor tamaño posible, que cumpla que nunca se le va a agotar la gasolina, es decir, $B[0] = 0$ y $A[B[i+1]] \leq A[B[i]] + c$ para $0 \leq i < k$.

Diseñe un algoritmo voraz. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.



jueves, 14 de noviembre de 2024 11:07

Q1

Ejercicio 4. Dado un arreglo A de n números naturales, queremos encontrar un arreglo B de tamaño n, que tenga a los elementos de A permutados y que minimize la suma $\sum_{i=1}^{n} iB[i]$.

Diseñe un algoritmo voraz. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.

Q1.- Array A (Tam n) · A = [1, 2]

Q2.- Array B, permutación de A, $val(B) = \sum_{i=1}^{n} i \cdot B(i)$

Q3.- Aquel B con mínimo valor. A = [1, 2, 3]

Q4.- g: Asignar el máximo elemento de A en B(1)

g: (Asumiendo A ordenado de forma decreciente) B(1) = A(1)

g: (Asumiendo A ordenado de forma creciente) B(1) = A(n)

g: Push back(a) el max elemento de A en B.

Q5.- A' = El array obtenido al eliminar un valor igual al g.c.y del array A

A' = A.erase(B(1))

A' = (asumiendo orden decreciente) A[2:n) A.pop_front()

A' = (asumiendo orden creciente) A(1..n-1) A.pop_back()

```
Algo(A)
  if(A.size() == 0) return []
  else
    B = []
    g = A(1)
    for i = 2 To n
      if (g < A(i)) g = A(i)
    A' = A.erase(g)
    B = Algo(A')
    B = B.push_front(g)
    return B.
```

```
// Asumir orden decreciente
Algo(A)
  if(|A| < 1) return []
  return Algo(A[2:n)).push_front(A(1))

// Asumir orden creciente
Algo(A)
  if(|A| ≤ 0) return []
  g = A[n]
  A' = A(1:n-1)
  return Algo(A').push_back(g)
```

G.C.P

Tq $\max(A) \in X$

Existe un opt X Tq $X(1) = \max(A)$

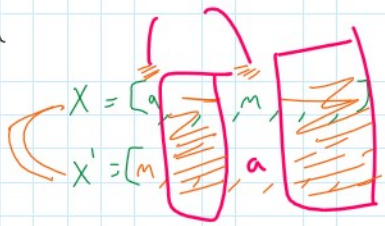
Prop: Definimos $m = \max(A)$. ← en particular $A(i) \leq m$

Prop: Demos $m = \max(A)$. ← en particular $A[i] \le m$

Sea X una sol opt cualquiera. Hay 2 casos

Caso 1: $X[1] = m$ ✓ (nada que probar)

Caso 2: $X[1] \neq m$



$\Rightarrow$ m está en alguna otra pos de X: pos j $1 < j \le n$. ($X[j] = m$)

Sea X' construido de la sgt manera:

- Swapeando las posiciones 1 y j del array X
- $X' = X$ luego del $swap(X_{[1]}, X_{[j]})$

$$X'[i] = \begin{cases} X[i] & i \neq 1, j \\ X[1] & i = j \\ X[j] & i = 1 \end{cases}$$

- Claramente $X'[1] = X[j] = m$ ✓
- X' es una perm A y es el swapeo de otra permutación.
- $val(X')$ vs $val(X)$

$$val(X') = val(X) - 1 \cdot X[1] - j \cdot X[j] + 1 \cdot X'[1] + j \cdot X'[j]$$

$$= val(X) - a - j \cdot m + m + j a$$

$$= val(X) + \underbrace{(a - m)}_{g^{(-)}}\overbrace{(j-1)}^{(+)} < val(X) \qquad val(X') < val(X)$$

up

$\Rightarrow$ X' es opt. ✓

O. S. P

Sea X sol opt Tq $X[1] = \max(A)$

$\therefore A' = A.erase(\max(A))$

Pq $X' = X_{[2:n]}$ sol opt A'.

Prop: Sg X' no es op $P(A')$

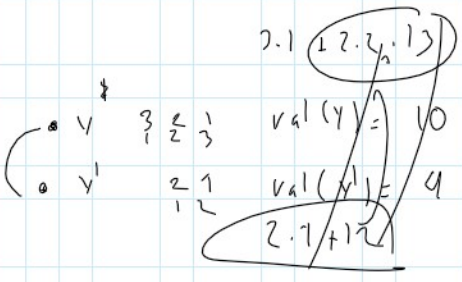
$\Rightarrow \exists Y'$ que sí es opt. $\Leftrightarrow$ $val(Y') < val(X')$

- $Y = Y'.push\_front(X_{[1]})$
- Y' permutación de A' y $A = A' \cup X_1$

$\Rightarrow$ Y permutación de A

$val(Y)$ vs $val(X)$

$$val(Y) = 1 \cdot Y_1 + \sum_{j=2}^{n} j \cdot Y[j] \qquad 2 \le j \le n \leftarrow (j = i+1) \rightarrow 2 \le i+1 \le n \quad 1 \le i \le n-1$$



$$val(Y) = \sum_{i=1}^{n} i \cdot Y[i] = 1 \cdot Y_1 + \sum_{i=2}^{n} i \cdot Y[i] = \sum_{i=2}^{n} i \, Y'[i-1]$$

$$= X_1 + val(Y') \neq \sum_{j=1}^{n-1} (j+1) \, Y'[j]$$

$$val(Y) = 1 \cdot Y_1 + \sum_{j=2}^{n} j \cdot Y(j) \quad \leftarrow (j = i+1) \quad 1 \le i \le n-1$$

$$= 1 \cdot Y_1 + \sum_{i=1}^{n-1} (i+1)\, Y(i+1)$$

$$= 1 \cdot Y_1 + \sum_{i=1}^{n-1} (i+1)\, Y'(i) = Y_1 + \sum_{i=1}^{n-1} Y'(i) \cdot i + \sum_{i=1}^{n-1} Y'(i) \cdot 1 = Y_1 + val(Y') + S(A')$$

$$= val(Y') + S(A)$$

$val(Y) = val(Y') + S(A)$ ∧ $val(X) = val(X') + S(A)$

$val(Y) = val(Y') + S(A) < val(X') + S(A) = val(X)$ $\Leftrightarrow$

Alternativa

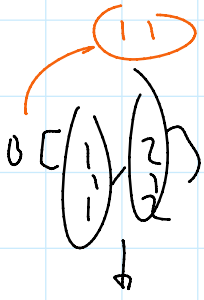
g := select $B(n) = \min(A)$.

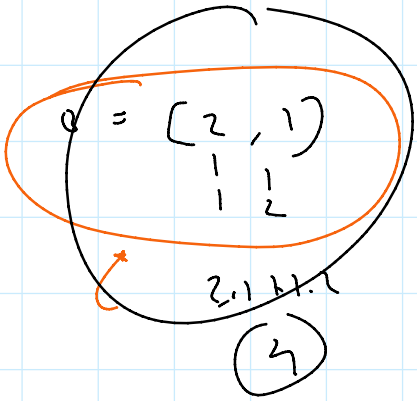
↳ $val(X) = val(X') + a_i \cdot X_n$

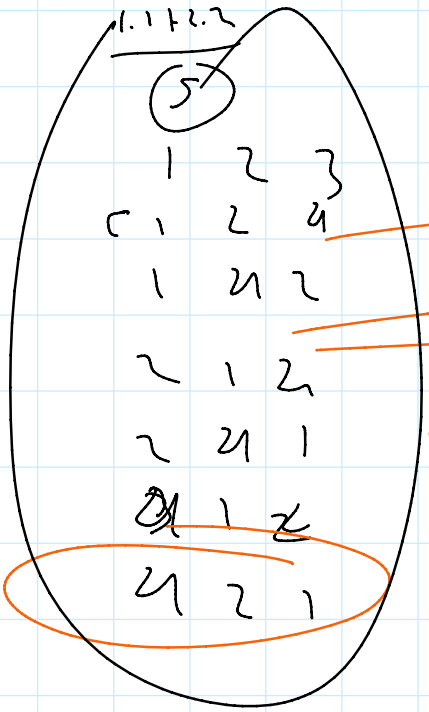
jueves, 14 de noviembre de 2024 11:29

$A = [1, 2)$

$(2, 1]$







$P = [1, 2, 4)$

$Q = [4, 2, 1]$

$1 + 4 + 12$

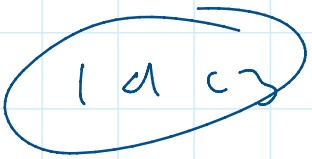
$1 + 8 + 6$

$2 + 2 + 12$

$2 + 8 + 3$ 13

$4 + 2 + 6$ 12

$4\ 4\ 3$ 11



	1	2	3	4
Y	4	3	0	1
Y'		3	0	1
		1	2	3

$val(Y) = 4 \cdot 1 + 3 \cdot 2 + 0 \cdot 3 + 1 \cdot 4$

$val(Y') = \_\_ + 3 \cdot 1 + 0 \cdot 2 + 1 \cdot 3$

$val(Y) - val(Y') = 4 \cdot 1 + 3 \cdot 1 + 0 \cdot 1 + 1 \cdot 1$

$= 4 + 3 + 0 + 1 = S(A)$

$2 \le i \le n \quad - \quad 1 \le j - 1 \le n - 1$

$j = i - 1 \quad (1 \le j \le n - 1)$

$$val(y) = \sum_{i=1}^{n} i \cdot y_i = y_1 + \sum_{i=2}^{n} i\, y_i = \sum_{i=2}^{n} i\, y'_{(i-1)} + y_1$$

$$= y_1 + \sum_{j=1}^{n-1} (j+1)\, y'(j) = y_1 + \sum_{j=1}^{n-1} j\, y'_j + \sum_{j=1}^{n-1} y'(j)$$

$$= y_1 + val(y') + s(A') = val(y') + s(A)$$

$$val(y) = val(y') + s(A)$$