INTERPOLACIÓN

- Aplicación de la definición

$a + bx_1 + cx_1^2 + dx_1^3 \ldots = f_1$

$a + bx_2 + cx_2^2 + dx_2^3 \ldots = f_2$

$\vdots$

resolver el sistema → tantos polinomios como valores haya, de grado $\leq n-1$ (n es el nº de ptos de soporte).

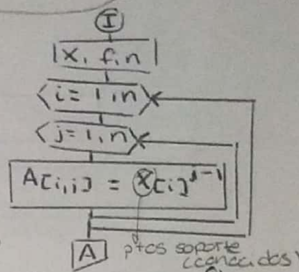
en forma de matriz: conocidos a, b, c... sustituimos en el pol por el valor a interpolar

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \ldots & x_3^{n-1} \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{pmatrix}$$

desconocida (incógnita)

puntos de soporte (conocidos)

Algoritmo para A (conocida)



- Polinomios de base

$p(x) = \sum_{i=1}^{n} f_i \cdot L_i$ (n es el nº ptos de soporte)

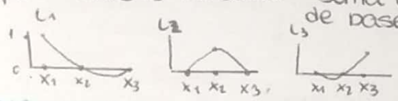
para el que se interpola

$L_i(x) = \frac{t - x_1}{x_i - x_1} \cdot \frac{t - x_2}{x_i - x_2} \cdots \frac{t - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{t - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{t - x_n}{x_i - x_n}$

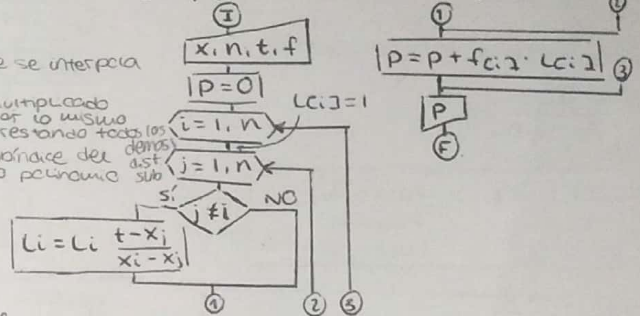
$L_i(x) = \prod_{\substack{j=1 \\ i \neq j}}^{n} \frac{t - x[j]}{x[i] - x[j]}$

para obtener el polinomio de base L_i → $\frac{t - x_1}{x_i - x_1} \ldots$ multiplicado por lo mismo (restando todos los demás sub) para el que se interpola; x con el subíndice del propio polinomio

para obtener el valor → suma de pol. de base · valor de la función.



Algoritmo para resolver todo (resultado de p(x)).



- Diferencias divididas.

tabla:

$x_1 \quad f_1 \quad f[x_1, x_2] = \frac{f_2 - f_1}{x_2 - x_1}$

$x_2 \quad f_2 \quad f[x_2, x_3] = \frac{f_3 - f_2}{x_3 - x_2}$

$x_3 \quad f_3 \quad f[x_3, x_4] = \frac{f_4 - f_3}{x_4 - x_3}$

$x_4 \quad f_4$

$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}$

$f[x_2, x_3, x_4] = \frac{f[x_3, x_4] - f[x_2, x_3]}{x_4 - x_2}$

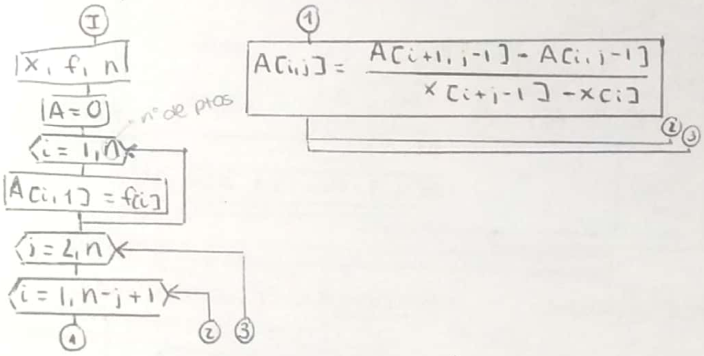
$f[x_1, x_2, x_3, x_4] = \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1}$

Fórmula de Newton solo con la primera fila:

$p(x) = f_1 + f[x_1, x_2] \cdot (x - x_1) + f[x_1, x_2, x_3] \cdot (x - x_1)(x - x_2) + f[x_1, x_2, x_3, x_4] \cdot (x - x_1)(x - x_2)(x - x_3)$.

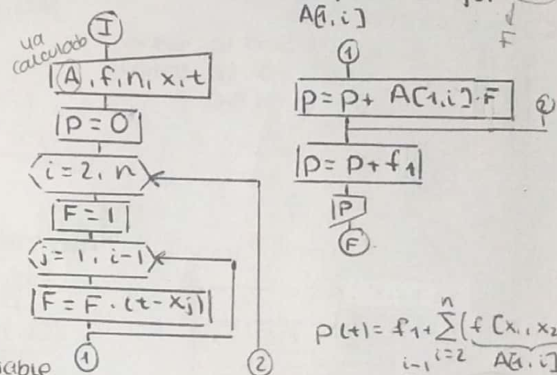
ir multiplicando la 1ª fila por x - todo lo que no esté al final en el corchete de f ($f[x_1, x_2]$ solo por $(x - x_1)$, $f[x_1, x_2, x_3]$ por $(x - x_1)(x - x_2)$)

Algoritmo tabla de diferencias divididas: (quitando x)



Algoritmo para el polinomio (fórmula de Newton):

$p = f_1 + \sum_{i=2}^{n} (\underbrace{f[x_1, x_2, \ldots x_i]}_{A[1,i]} \cdot \underbrace{\prod_{j=1}^{i-1} (t - x_j)}_{F})$



$p(t) = f_1 + \sum_{i=2}^{n} (\underbrace{f[x_1, x_2 \ldots x_i]}_{A[1,i]} \cdot \underbrace{\prod_{j=1}^{i-1} (t - x_j)}_{F})$

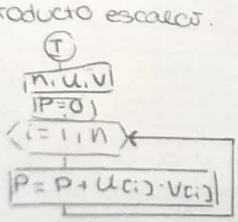
ALGORITMIA

- signo igual
  - Comprobar que todas las variables ya están
  - Operación a la derecha, se guarda en la variable
- Bucles
  - tantos como subíndices, uno debajo del otro
  - 1º el subíndice independiente (más a la izq)
- Sumatorio
  - a la izq del = → arriba = 0
  - bucle con la i del sumatorio
  - variable izq = variable izq + operación del interior
- Productorio
  - a la izq. del = → arriba = 1
  - bucle con la i del prod.
  - variable izq = variable izq * op del interior

si un bucle no depende de otro los separamos

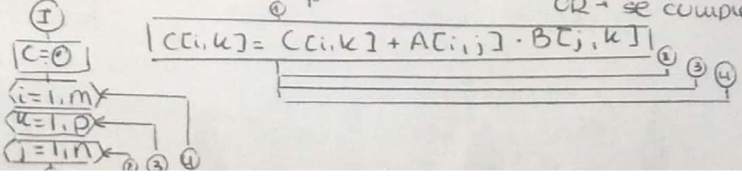
entrada · operación · salida · (rombo) no hace la operación, pregunta sí/no (no hace falta crear variable) · i = ①②③: ① inicial, ② final, ③ incremento

producto escalar.



producto de matrices.

$A = m\{(\underbrace{\quad}_{n})$ $\quad B = n\{(\underbrace{\quad}_{p})$



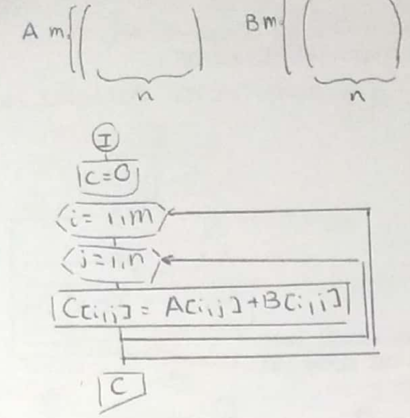
si queremos poner dos igual? ==

AND → se tienen que cumplir las dos condiciones

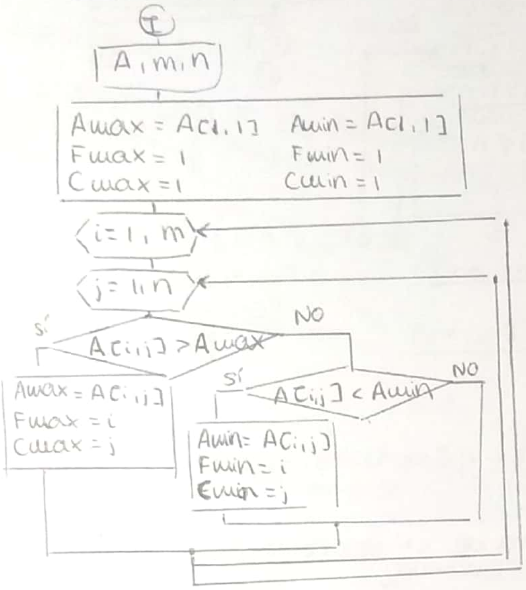
OR → se cumple una de las dos

Escaneado con CamScanner

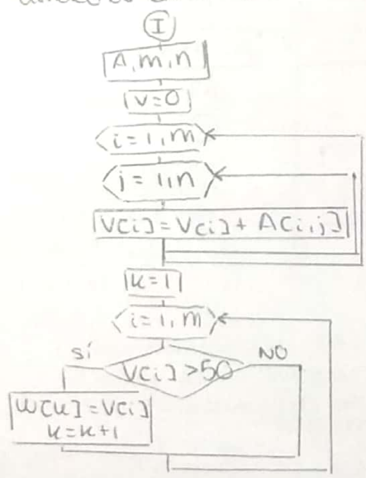
suma de matrices:



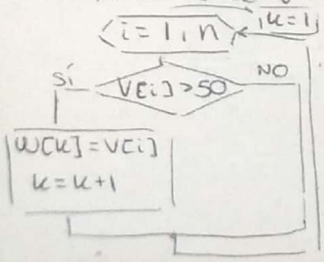
valor máx y mín de matriz



m aminoácidos, n muestras, [ ] en matriz m,n
V=sus elementos sean la suma de [ ] de cada uno (m comp)
almacenar en w [ ] > 50.



Almacenar en w las componentes de v > 50

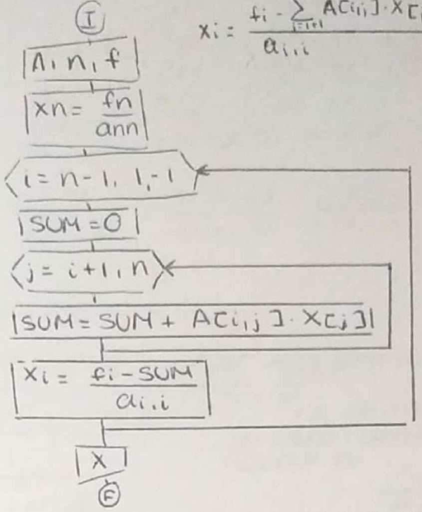


sistema triangular superior por remonte:

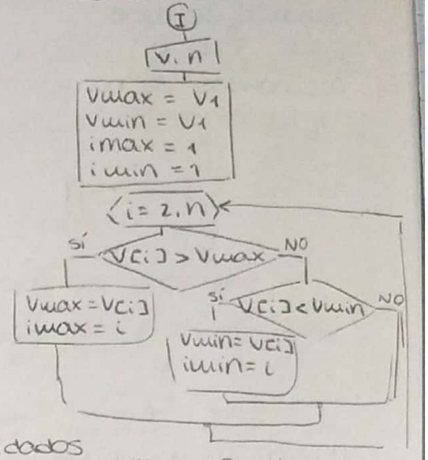
$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & a_{24} \cdots & a_{2n} \\ 0 & 0 & a_{33} & a_{34} \cdots & a_{3n} \\ 0 & 0 & 0 & 0 & a_{nn} \end{pmatrix} \quad x: \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

conocidos　　　　conocidos

$$x_i = \frac{f_i - \sum_{j=i+1}^{n} A[i,j] \cdot x[j]}{a_{i,i}}$$

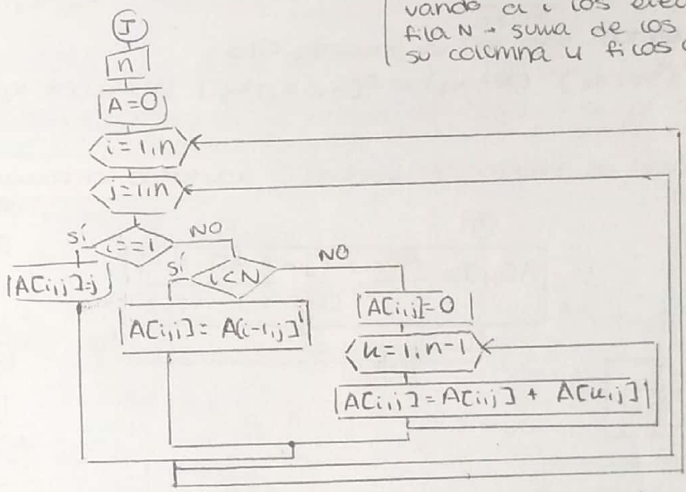


valor máx y mín de un vector.

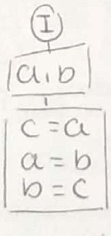


para sumatorios / productorios anidados
- empezar con el + externo, poner fuera nombre = 0 o 1, seguido de su bucle. Correspondiente operación al final.
- poner lo que haya dentro en el espacio.
- el bucle siempre se cierra una vez hecha la operación.

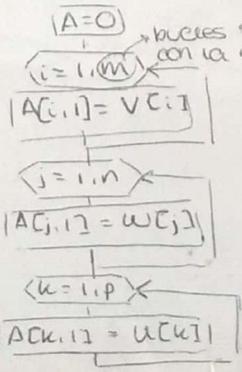
matriz cuadrada n,n:
1ª fila formada por nºs naturales
fila i (i = 2, 3, ... n-1) se obtiene elevando a i los elementos de i-1
fila N → suma de los elementos de su columna y filas anteriores



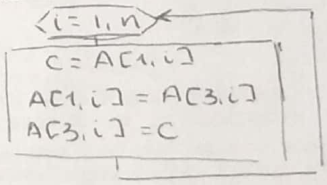
intercambios variables:



construir matriz a partir de vectores dist longitud

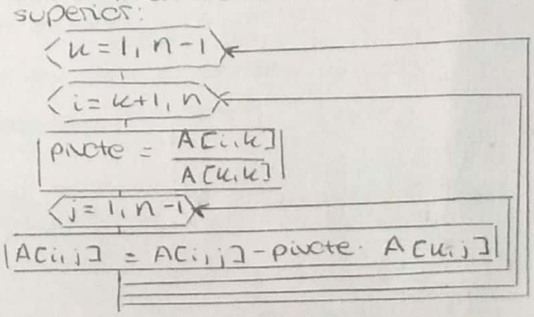


intercambios columnas/filas



pivote = $\frac{A_{i,k}}{A_{kk}}$
k = 1, 2, n-1
i = k+1, k+2, n
A_{ij} = A_{ij} - pivote · A_{kj}
(j, k): 1, ... n
i = k+1, .. n

convertir en matriz triangular superior:



inicializar matrices y vectores a 0
* sumatorio
* productorio