PRÀCTICA 1: POLINOMIS

Andreu Pérez Torra

Dia 28 de Març de 2021

Universitat de Lleida

Grau en Enginyeria Informàtica

Programació II

ÍNDEX

INTRODUCCIÓ

En aquest document, trobareu l'informa realacionat amb la pràctica 1, de l'assignatura de programació II. Aquesta, consisteix en en implementar diferents funcions per a solventar petits problemes amb polinomis mitjançant arrays. Al llarg de la pràctica, veurem que treballarem amb polinomis representats en forma compressed o en forma expanded. Ho explico a continuació.

Representació d'un polinomi en format expanded: Consisteix en la represetnació del polinomi en una matriu d'una sola fila i zero columnes. Aquesta contindrà tots aquells termes del polinomi que contingui o no un zero en el coeficient. El valor de l'exponent vindrà determinat per la posició de la taula, observem-ho en el següent exemple:

0	1	2	3	4
1	6	3	0	1

$\longrightarrow \quad 1 + 6x + 3x^2 + x^4$

Representació d'un polinomi en forma compressed: Consisteix en la representació del polinomi en una matriu de de columnes i tant numeros de files com termes diferetns de zero hi hagi en el polinomi. El valor de l'exponent, vindrà determinat per els valor de la primera columna i el dels coeficients per la segona columna. En aquesta, no hi representarem cap dels valors que tingui un coeficient igual a zero.

	0	1
0	0	1
1	1	2
2	2	3
3	4	1

$\longrightarrow \quad 1 + 6x + 3x^2 + x^4$

FUNCIONS SIZE

Les primeres dues funcions que observem en el codi, consisteixen principalment a obtenir la mida d'una array representada d'una forma, mitjançant la matriu de format oposat que es passa com a paràmetre a la funció. Aquestes funcions, n'ajudaran a determinar les mides a l'hora de crear futures arrays en funcions posteriors.

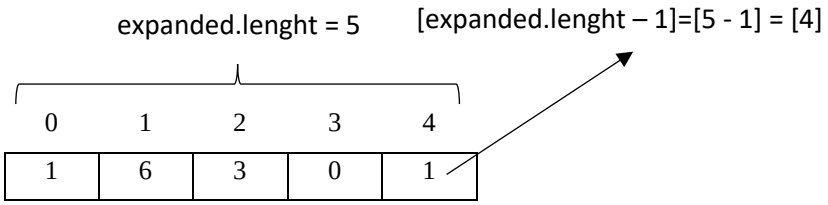
expandedSize

L'element principal que ens determinarà de longitud de la matriu expanded és el del grau més gran del polinomi.

És per això que he resolt aquesta funció mitjançant un condicional. En cas que la longitud de la matriu proporcionada sigui diferent de zero, accedirem a la casella en la qual es troba el valor que desitgem. En

aquest cas, el valor del grau més elevat, el podem trobar a la posició [compressed.lenght-1][0], que a la casella de la primera columna de l'última fila.

He utilitzat compressed.lenght, ja que aquest ens retorna la mida de columnes que té la matriu i l'hi he restat 1, ja que per exemple si la matriu és de mida 5, la seva numeració va del 0 al 4. Observem-ho en el següent exemple:



Finalment, n'hi sumem un al valor total que retornem, això és pel fet que hem de tenir en compte el terme independent del polinomi. Si solament retornàssem el valor del grau més gran, per exemple dos, a la matriu solament ens hi cabrien dos elements, quan aquest polinomi hauria d'estar format pel terme independent, exponent1 i exponent 2.

En el cas que la matriu en forma compressed fos buida, és a dir, de mida [0][2], el valor que retornaria seria 0, ja que no hi hauria cap valor a introduir a la taula del polinomi de forma expanded. Una altra manera de solucionar aquest problema, és el d'utilitzar un bucle que vagi recorrent tota la matriu i en retorni l'últim valor llegit +1, emmagatzemat a una variable de tipus enter inicialitzada a 0.

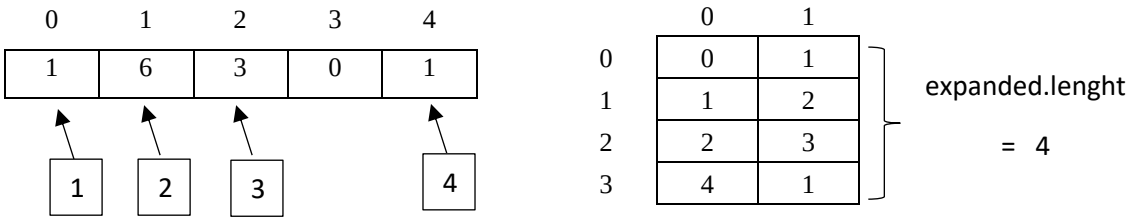
D'aquesta forma, no caldria utilitzar els condicionals per a comprovar si la longitud de l'array obtinguda és o no, diferent de zero. Això s'explica de manera que al fer la comprovació per a no sortir-se de l'array, mitjançant i < compressed.lenght, sent i la posició de la matriu en la qual volem accedir, ens trobaríem en què 0 < compressed.lenght (que en aquest cas seria 0) i per tant ja no es compliria la condició i retornaria 0 (valor el qual s'hi hauria inicialitzat la variable inicial comentada anteriorment).

M'he decantat per la implementació del primer mètode, ja que el programa s'estalvia de recórrer tot l'array i anem directament a buscar el valor que desitgem.

compresedSize

Per a realitzar el desenvolupament d'aquesta funció, s'ha hagut de tenir en compte que si el valor d'algun dels termes del polinomi era igual a 0, no havia de constar en aquesta matriu.

Tenint present això, la fórmula que he dissenyat consisteix en recorre tota la matriu, de format extended, i acomular en la variable comptador la quantitat d'exponents amb terme diferent de 0. El valor final d'aquesta variable, correspon a la mida de la matriu en forma compressed. Vegem-ho:



Un petit detall a destacar, és que en el disseny inicial, havia implementat un condicional per a localitzar aquelles matrius que eren buides, és a dir de tamany 0. Però finalment em vaig adonar que la condició que utilitzava el bucle per a seguir recorrent la matriu (posició_actual (incialment 0) < compressed.lenght (que en aquest cas seria 0)), ja s'incomplia en el primer moment i retornava directament el valor inicial 0 del comptador.

CREATEEXPANDED I CREATECOMPRESSED

L'objectiu d'aquestes dues funcions, no és altre que crear les arrays per a poder representar i treballar amb els polinomis. El procediment no és gens complex, sinó que únicament crea la matriu necessària, obtenir la seva mida utilitzant els valors que s'hi passen com a paràmetre.

Una petita modificació que vaig realitzar per a concloure amb el resultat final, va ser que, en lloc de crear la matriu i després retornar-la, retornar-la directament sense guardar-la en cap espai de memòria dins de la funció createExpanded/Coompressed.

He realitzat aquest canvi, ja que el codi és mostra més simplificada i trobo que no calia guardar la matriu i després retornar-la si ho podia fer directament.

FUNCIONS COPY

Com veurem en apartats posteriors, necessitarem una eina que ens permeti copiar una matriu en un format a un altra del format oposat i és aquesta funció la que ens permetrà fer-ho.

copyTo (from expanded, to compressed)

La formula dissenyada per a dur a terme la còpia dels valors d'una matriu extended a una matriu compressed, consisteix en recorre la taula que conté els valor que volguem copiar i al mateix temps que recorrem la matriu resultant.

És a dir, s'implementa un bucle que va saltant de cassella en cassella i va comprovant, mitjançant un condicional, si el valor que hi ha és o no diferent de zero, en cas que no ho sigui es copia la posició en què ens trobem (que correspon a l'exponent) a la posició [comptador][0] i també en copia el valor que hi ha a la posició [comptador][1] de la matriu compressed.

La variable comptador, és de tipus enter i la seva funció és determinar en quina part del recorregut de la matriu compressed ens trobem. Aquesta, només s'augmenta quan col·loquem un element dins la taula compressed.

Si ens hi fixem, el variable comptador, mai farà que ens sortim de l'array, ja que el comptador s'incrementarà tantes vegades com termes diferents de zero hi hagi, valor que coincideix amb el tamany de la matriu compressed (fet explicat a l'apartat compressedSize).

copyTo (from compressed, to expanded)

El codi que he implementat en aquesta funció, ha estat el primer que vaig desenvolupar, i no he trobat manera per ha simplificat-lo més. Això és degut al fet que només necessito recorre la matriu compressed una sola vegada per a obtenir-ne el resultat.

En el llenguatge de programació Java, quan creem una matriu, aquesta s'inicialitza directament a 0. Per aquest motiu, i com que en la matriu compresses només hi ha tots aquells exponents que tenen un terme diferent de 0, ens podem posicionar directament en aquelles posicions de l'array expaneded determinades per la primera columna de la matriu compressed la qual correspon als exponents, i copiar els valor que conté la segona que correspon als termes del polinomi.

FUNCIONS COMPRESS I EXPAND

Aquestes dues funcions, tenen la principal finalitat de convertit les arrays d'un format a un altre. El desenvolupament que he dissenyat, se centra principalment en la utilització d'altres funcions creades anteriorment, ja que simplifica rotundament el codi i al mateix temps en facilita la seva comprensió.

Senzillament, consisteix en crear la matriu del format oposat al que s'ens passa com a paràmetre mitjançant les funcions createExpnded o bé createCompressed i a continuació copiar-ne el valor mitjançant les funcions copyTo.

Com es pot observar en la primera linia de la funció, en lloc de guardar el tamany en una variable

i després passar-lo com a paràmetre, el podem passar directament estalviant-nos la creació d'una variable. He optat per fer-ho així, ja que s'obté una petita simplificació en el codi.

FUNCIÓ POW

El seu objectiu no és altre que retornar el resultat d'elevar cada un dels coeficients x d'un polinomi amb el seu exponent. Aquesta funció ens serà força útil a l'hora de realitzar els càlculs en les funcions evaluate.

La metodologia implementada consisteix en anar multiplicant el valor guardat en la funció basememory per la base fins que el comptador sigui més petit o igual que l'exponent. He afegit aquest igual, ja que en cas que a l'inici l'exponent sigui igual que el comptador, executi el bucle una sola vegada.

Si ens hi fixem, la variable comptador està inicialitzada a 2, això és degut al fet que si algun coeficient està elevat a 1 el resultat no varia respecte al valor inicial. És a dir, $x^1 = x$. En cas que el valor exponent sigui igual a 0, retornarem directament un 1, ja que qualsevol $x^0 = 1$.

FUNCIONS EVALUATE

Ni més ni menys, el seu principal objectiu és el de calcular els polinomis a partir d'un valor x proporcionat.

El desenvolupament dissenyat per a resoldre els polinomis consisteix en recorre la matriu proporcionada i calcular un a un els termes mitjançant la funció pow explicada en l'apartat anterior.

En el cas de l'expanded, agafem com a exponent la posició de la taula en la qual es troba el terme que multiplica la x, i en el cas de la compressed els valors que es troben en la primera columna.

A continuació, apliquem el mateix procediment tant per a la forma expanded com en la compressed. Aquest consisteix en un cop obtingut el càlcul realitzat per la funció pow, el multipliquem pel terme que acompanya la x i el sumem a la variable memory, la qual és l'encarregada d'emmagatzemar i acumular els càlculs de cada un dels termes.

FUNCIONS ADD

La finalitat de les funcions add, no és altra que sumar dos polinomis. Aquest, és un dels apartats que més m'ha costat desenvolupar, sobretot el de la suma en format compressed, principalment pel fet que en

algunes ocasions la matriu pot resultar nula o bé obtenir valors zero i que aquests s'hagin d'eliminar de la matriu.

addExpanded

Inicialment, per a dur a terme el desenvolupament d'aquesta, he creat una funció externa anomenada còpia, la qual té la funció de copiar les matrius proporcionades, pel fet que a l'hora de treballar, no modifiquem les originals.

El mètode que he implementat, consisteix a sumar la matriu petita a la matriu gran mitjançant la funció sumaexpended que jo mateix he creat. Això és degut al fet que si n'afegim una dins de l'altra, ens estalviem haver de crear una nova matriu. Un cop obtingut aquest resultat, mitjançant la funció zeros comprovem si existeixen coeficients que hagin donat zero hi hagin de ser extrets. A continuació, és crea una matriu amb les mides corresponents usant la funció creació, i es copien els valors de la matriu en la qual hem emmagatzemat la suma a aquesta que hem creat posteriorment.

Si ens fixem en el codi, podem observar que en utilitzar diferents funcions i el fet d'editar les matrius sobre les quals treballem, provoca que en puguem estalviar crear diferents matrius. Inicialment, el codi que havia implementat, cada cop que una funció era modificada la guardava en una de nova, fet que finalment he pogut simplificar.

addCompressed

Incialment, igual que en la funció anterior addExpanded, he crear una funció anomenada copiacompressed que copia les matriu que es passen com a parametre per a no modificar-ne els seus valors incicals.

A continuació, implemento una serie de condicionals per a les diferentssituacions que ens podem trobar. He separat les diferents situacions, ja que tres de les quatre que ens podem trobar, ens permeten directament retornar la matriu sense haver de realitzar cap calcul.

En la última situació, és a dir, que cap de les dos matrius sigui buida, cal realitzar una sèrie d'operacions per a obtenir el resultat desitjat. El desenvolupament implementat, consisteix en crear una matriu de mida igual a al grau més gran d'entre els dos polinomis. A continuació, utilitzant la funció sumacompressed, n'he realitzat la suma de le dos matrius. El pas seguent, és retirar-ne els zeros, és a dir, descrtar totes aquelles posicions que contenen un zero en el coeficient, ja que la forma compressed no conte aquestes com ja he explicat anteriorment. Finalment, creo una matriu de la mida dient, és a dir, que només hi tinguin lloc tots aquells valor que continguin yun base diferetn de 0 i en copio els valors de la matriu que conté els resultats de la suma.

CONCLUSIÓ

Un cop ja he acabat la pràctica, he conclòs amb què els meus coneixements sobre el programa IntelliJ i la programació amb Java han millorat. He après a treballar de millor manera amb les arrays i a fer un codi que ja funcionava més eficient i més compacte perquè sigui més senzill d'entendre.

Respecte a la pràctica, penso que ha sigut un repte en moltes funcions, ja que per a algunes representacions s'han de tenir en compte molts aspectes perquè la funció faci la feina correcta per a tots els casos possibles.