PRACTICA 1

Programació 2

Ton Llucià Senserrich
47125160T
18/02/2020
Pralab2

- ## Apartat 1 (allBits)

En aquest apartat s'ha de dissenyar una funció per tal de comprovar si un array d'enters està format únicament per zeros o uns.
Per implementar aquesta funció necessitem recórrer cada una de les posicions de l'array i comprovar per cada una si és un 0 o ve un 1, per fer això he implementat una funció auxiliar anomenada "isBit", que comprova si un enter és un bit. Per tal de realitzar la comprovació de valor correcte, a partir d'aquí únicament s'ha de buscar un valor que no compleixi amb la condició, si el trobem retornarem fals i si no el trobem, implica la sortida del for i retornarem true.

- ## Apartat 2 (copy)

En aquest apartat es demana una funció que a partir d'un string, que representa un nombre binari, amb els bits més significatius a la part esquerra la transformi en un array d'enters amb els bits més significatius a la dreta de l'array, també es demana que si l'string conté qualsevol valor no binari, l'obviem i si tant l'array com l'string són de diferents mides que trunqui o allargui el valor segons correspongui.
Per desenvolupar aquesta funció primerament hem de tenir en compte que l'array i l'string estaran en sentits oposats, per tant he utilitzat "fromPos" per recórrer l'string i "toPos" per recórrer l'array, els quals començaran a "from.length-1" i a 0 respectivament.
També hem de tenir en compte que l'string està format per caràcters i l'array ha d'estar format per enters, per aquest motiu hem d'aplicar-li una transformació al caràcter en qüestió per tal de convertir-lo en l'enter corresponent, això s'aconsegueix amb la instrucció "num=from.charAt(fromPos)-'0';".
Seguidament hem de comprovar també que sigui un valor binari valid , aixo ho aconseguim utilitzant la funció desenvolupada en l'apartat anterior anomenada "isBit".

- ## Apartat 3 (narrow)

En aquest apartat es requereix una funció que escurci un array d'enters que conté un nombre binari, per obtenir un array de la llargada especificada, concretament menor a l'actual.
Per dissenyar aquesta funció primerament hem de tenir en compte si estem fent una crida valida, és a dir comprovar que la llargada desitjada és més gran o igual a 0 i que és més petita o igual que la del array d'origen, un cop comprovat simplement podem utilitzar la funció "copy" desenvolupada en l'apartat anterior, ja que si els dos arrays són de la mateixa mida es copiarà de l'un a l'altre i si la llargada del destí és més petita que la de l'origen, truncarà.

- ## Apartat 4 (widen)

La funció a realitzar ha de permetre allargar un nombre binari contingut en un array d'enters a una llargada especifica, concretament mes gran o igual a l'actual tenint en compte el signe del valor original, ja que en cas de ser positiu s'ha d'afegir 0 i en cas de ser negatiu s'ha d'afegir 1 en els bits de més pes, per tal de mantenir el valor numèric.
Per resoldre aquest apartat, que resulta molt semblant a l'anterior, primerament comprovem que la llargada desitjada és més gran o igual a 0 i també més gran o igual a l'actual, un cop comprovat, seguidament comprovem si el nombre és negatiu mirant si el bit de més pes és igual a 1, si es així creem un array de la mesura desitjada però ple d'1, utilitzant la funció auxiliar "oneOfSize", en cas contrari el creem ple de 0, utilitzant la funció "zeroOfSize", i només

ens queda copiar els valors ja existents, cosa que realitzem mitjançant la funció "copy" desenvolupada anteriorment i finalment retornem el resultat.

• Apartat 5 (cast)

En aquest apartat hem de desenvolupar una funció la qual rebi un array d'enters que representa un nombre binari i un enter que determinarà la llargada desitjada, a partir d'aquests paràmetres retornar un array d'enters amb la llargada desitjada i el contingut de l'array rebut, truncant o afegint valors si és necessari.
Per desenvolupar aquesta funció, que és simplement escurçar l'array si la llargada desitjada és més petita que l'actual, allargar-la si la llargada desitjada és més gran que l'actual i fer una còpia si les dues llargades són iguals, per tant la meva implementació el que fa és utilitzar les funcions "narrow" i "widen" desenvolupades en apartats anteriors, de manera que només cal fer l'if de si és més gran o més petita i cridar una funció o l'altre, i com que les dues funcions només copien l'array original si la llargada desitjada és igual a l'actual, és igual si cridem a narrow o a widen en aquest cas. Jo he utilitzat el triple condicional per tal de simplificar el codi, però també es podria utilitzar if.

• Apartat 6 (and)

En aquest apartat es requereix una funció que rebi 2 arrays d'enters i realitzi l'operació and bit a bit, es vol també poder utilitzar aquesta funció per a arrays amb diferents mides, per tant es demana que si una d'elles es representa en 64 bits hem de transformar les dues a long, en cas contrari hem de transformar les dues a integer.
Per desenvolupar he creat una funció auxiliar anomenada "andSameLen", la qual realitza l'operació and a partir de dues arrays però assumint que tenen la mateixa llargada, i una altra anomenada "getResultLen" que retorna la llargada indicada per poder representar correctament les dues arrays. Amb aquestes dues funcions l'únic que he realitzat ha sigut corregir les dimensions de les arrays mitjançant la funció "getResultLen" i la funció "cast", i finalment he cridat a la funció "andSameLen" que l'unic que fa és un bucle per cada posició dels arrays i calcula l'operacio and utilitzant l'operador &.

• Apartat 7 (or)

En aquest apartat es demana una funció que, igual que l'anterior, corregeixi les llargades dels arrays que rep per paràmetre i realitzar l'operació or bit a bit.
Per a realitzar aquesta funció he desenvolupat una funció auxiliar anomenada "orSameLen" que s'utilitza per realitzar l'operació or de dues arrays suposant que tenen la mateixa llargada, per tant el que he fet, com en la funció de l'apartat anterior, ha sigut calcular la longitud ideal per interpretar ambdues arrays mitjançant la funció "getResultLen", després he cridat a la funció "orSameLen" per a realitzar l'operació or mitjançant l'operador | i retornar el resultat.

• Apartat 8 (leftShift)

En quest apartat es demana una funció que, rebent per paràmetre un array d'enters, desplaci els valors de l'array x posicions a l'esquerra segons el nombre rebut i ompli les posicions de la dreta amb zeros.
Per a desenvolupar aquesta funció he creat una funció auxiliar anomenada "leftOneShift" que serveix per desplaçar cap a l'esquerra una posició de l'array; i una altra funció per tal de saber la longitud amb la qual haurem de treballar anomenada "getResultLen", per tant, la crido, em

guardo el valor que retorna i l'utilitzo per transformar l'array mitjançant la funció "cast" i també per aconseguir el nombre de posicions a desplaçar mitjançant l'operació modul, per últim faig un bucle i crido a la funció "leftOneShift" x cops segons el nombre calculat anteriorment i retorno el resultat.

• Apartat 9 (unsignedRightShift)

En aquest apartat es demana una funció que, a partir d'un array de enters i un enter, desplaçi els valors de l'array un nombre de posicions cap a la dreta pero no sense tenir en compte el signe del nombre binari que representa l'array rebut.

Per tal de desenvolupar aquesta funció he creat una funció auxiliar anomenada "rightOneShift", que desplaça una posicio cap a la dreta i segons el valor del parametre boleà omplirar els valors de la esquerra amb uns o zeros respectant el signe del nombre original. Per tant en la funcio principal el que he fet ha estat cridar a la funció esmentada a l'apartat anterior anomenada "getResultLen" per obtenir la llargada ideal per poder treballar, despres aconseguir el nombre de posicions que he de desplaçar, transformar l'array a la llargada ideal i ultimament amb un bucle cridar la funció "rightOneShift" tants cops com el nombre calculat, i retornar el resultat.

• Apartat 10 (signedRightShift)

En aquest apartat es demana una funció que, rebent un array d'enters en representació d'un nombre binari i un enter, haurà de desplaçar el nombre original de l'array un nombre de posicions cap a la dreta respectant el signe del nombre original. Per tal de dur a terme aquesta funció he utilitzat la funció auxiliar anomenada en l'apartat anterior "rightOneShift" utilitzant el paràmetre negative de la funció per tal que en cas que el nombre sigui negatiu, ompli d'1 els espais de l'esquerra.

• Conclusions

En conclusió, en aquesta practica, com que ja disposava de coneixements previs del llenguatge java i de la programacio orientada a objectes, m'ha servit per reforçar els meus coneixements i per apendre a realitzar un informe.

• Bibliografia

- www.w3schools.com/java