

Design and Analysis of Algorithms- IT 240

Name :Sameera Madushanka Abeysekara

DIT :IT14121616

Batch :03

<html>

<head>

<link rel="stylesheet" type="text/css" href="daaCss.css">

<script type="text/javascript" src="jquery-1.4.2.js"></script>

</head>

<body>

<h2> Fixed Partition Memory Management </h2>

<h3> Design and Analysis of Algorithms </h3>

<p>Memory Segment : <input id="msegment" type="text" size="8" /> Number Of Programs : <input id="noprograms" type="text" size="8" /></p>

<table id="myTable">

<tr id="addmem"></tr>

</table>

<table id="myTable1">

<tbody id="dbody">

</tbody>

</table>

<button id="myButton" onclick="myFun()">Calculate</button>

<table id="myTable2">

<tbody id="ddbody">

</tbody>

</table>

<br>

<button id="myButton2" onclick="reloader()">Try another one</button>

<script type="text/javascript">

$(document).ready(function () {

myButton2.style.display = 'none';

});

//region size taker

$('#msegment').keyup(function (event) {

$('#addmem').empty();

var a = document.getElementById("msegment").value;

var b = a;

$('#addmem').append('<td><p>Enter Size of ' + b + ' region</p></td>');

var y = 1;

for (x = 0; x < b; x++) {

$('#addmem').append('<td> <input type="text" id="regionvalues' + x + '" placeholder="region ' + y + ' Size" > </td>');

y = y + 1;

}

});

//-------------------------------------------------------------------------------------------------------//

//program count taker

$('#noprograms').keyup(function (event) {

var a = document.getElementById("msegment").value;

if (a == "") // check if region count empty

{

alert("Memory Segment Cannot be Empty !!");

document.getElementById("msegment").focus();

document.getElementById("noprograms").value = "";

}

else {

var b = document.getElementById("noprograms").value;

var c = b;

if (a == "0" && b == "0") // if both empty end program

{

myButton.style.display = 'none';

myButton2.style.display = 'inline';

alert("Program End");

}

else // create table to input number of programs

{

$('#dbody').empty();

for (x = 0; x < c; x++) {

$("#dbody").append('<tr class="' + x + '" onclick="myFunction(this)" id="enterss' + x + '">');

$("." + x + "").append('<td > <input type="text" id="enters' + x + '" placeholder="Number of Programs"> </td>');

$("." + x + "").append('</tr>');

}

}

}

});

//-------------------------------------------------------------------------------------------------------//

//automatically create size of program and exe time slots

function myFunction(x) {

var d = x.rowIndex;

$('#enters' + d).keyup(function (event) {

var aa = document.getElementById('enters' + d).value;

var ba = aa;

$('#myTable1 #enterss' + d).find('td:eq(' + 0 + ')').nextAll().remove();

for (x = 0; x < ba; x++) {

$('#enterss' + d).append('<td > <input type="text" id="sizepro' + d + x + '" placeholder="Size of Program"> </td>');

$('#enterss' + d).append('<td > <input type="text" id="extime' + d + x + '" placeholder="Execution Time"> </td>');

}

});

}

//----------------------------------------------------------------------------------------------\*\*

var memoryarray = []; // memory size to array

var numOfPrograms = []; // number of program going to insert in to array

var programsize = []; //insert program sizes

var programtime = []; //insert program time

var counter = 0;

var programsizeUNSORT = [];

var programtimeUNSORT = [];

function myFun() {

myButton.style.display = 'none';

myButton2.style.display = 'inline';

var memory = document.getElementById("msegment").value;

var program = document.getElementById("noprograms").value;

//memory size assign to array

for (x = 0; x < memory; x++) {

memoryarray[x] = document.getElementById("regionvalues" + x).value;

}

//number of programs assing to array

for (y = 0; y < program; y++) {

numOfPrograms[y] = document.getElementById("enters" + y).value;

//alert(numOfPrograms[y]);

}

//finally add programs and time to arrays

for (z = 0; z < program; z++)

{

if (numOfPrograms[z] == 1)

{

//assing value when program is 1

programsize[counter] = document.getElementById("sizepro" + z + 0).value;

programtime[counter] = document.getElementById("extime" + z + 0).value;

programsizeUNSORT[counter] = document.getElementById("sizepro" + z + 0).value;

programtimeUNSORT[counter] = document.getElementById("extime" + z + 0).value;

// alert(programsize[counter]);

}

else if (numOfPrograms[z] > 1) {

//assing values when program is more than one

var valueoftime = 0;

var numofplace = 0;

for (w = 0; w < numOfPrograms[z]; w++)

{

//alert(numOfPrograms[z]);

if (w == 0) {

valueoftime = document.getElementById("extime" + z + 0).value;

}

var valueoftime1 = document.getElementById("extime" + z + w).value;

if (parseInt(valueoftime) > parseInt(valueoftime1))

{

valueoftime = valueoftime1;

numofplace = w;

}

}

programsize[counter] = document.getElementById("sizepro" + z + numofplace).value;

programtime[counter] = document.getElementById("extime" + z + numofplace).value;

programsizeUNSORT[counter] = document.getElementById("sizepro" + z + numofplace).value;

programtimeUNSORT[counter] = document.getElementById("extime" + z + numofplace).value;

//alert(numofplace);

}

counter = counter + 1;

}

//programtime.toString();

//programsize.toString();

//alert(programtime+"program time");

//alert(programsize+"program size");

// sort arroding to time

insertionSort(programtime, programsize);

//programtime.toString();

//programsize.toString();

// alert(programtime+"program time"+" "+programsize+"program size");

//check in for best size to store

var selectone = null;

var multi = 0;

var SizeAssingToRegin = [];

var regionstore = [];

var tempTimeStore = [];

var memoryName = [];

var perValue = [];

var afterValue = [];

var averageTime = 0;

for (a = 0; a < memoryarray.length; a++) {

memoryName[a] = 0;

}

for (x = 0; x < programsize.length; x++)

{

for (y = 0; y < memoryarray.length; y++)

{

if (memoryarray[y] == memoryarray[selectone]) {

}

else if (parseInt(programsize[x]) <= parseInt(memoryarray[y]))

{

//var zz=parseInt(y)+1;

// var zzz=programsizeUNSORT.indexOf(programsize[x])+1;

SizeAssingToRegin[x] = programsizeUNSORT.indexOf(programsize[x]); //assign the place of value in ordered array to unordered array

tempTimeStore[x] = programtimeUNSORT[programsizeUNSORT.indexOf(programsize[x])];

regionstore[x] = parseInt(y) + 1;

selectone = y;

// alert(programsize[x]+" "+programtime[x]);

multi = parseInt(multi) + 1;

//time assigner

var regionNumber = y;

//time to correct region and count the time with previous region

for (b = 0; b < memoryName.length; b++)

{

if (parseInt(regionNumber) == parseInt(b))

{

//alert("memory name before "+memoryName[b]);

perValue[x] = memoryName[b];

afterValue[x] = parseInt(perValue[x]) + parseInt(tempTimeStore[x]);

memoryName[b] = afterValue[x];

// alert("memory name after "+memoryName[b]);

break;

}

}

break;

}

if (multi == memoryarray.length) {

selectone = null;

multi = 0;

}

}

}

//sorting array back to input style

insertionSort2(SizeAssingToRegin, regionstore, tempTimeStore, perValue, afterValue);

//alert(tempTimeStore.toString());

//making array for time calculate

for (xy = 0; xy < SizeAssingToRegin.length; xy++) {

var programNumber = SizeAssingToRegin.indexOf(xy) + 1;

var zzz = SizeAssingToRegin.indexOf(xy);

var regionToProgram = regionstore[zzz];

// var z=tempTimeStore[zzz];

var vperValue = perValue[xy];

var vafterValue = afterValue[xy];

// var regionNumber=parseInt(regionToProgram)-1;

/\* for(b=0;b<memoryName.length;b++)

{

if(regionNumber == b)

{

perValue=memoryName[b];

afterValue=parseInt(perValue)+parseInt(z);

memoryName[b]=afterValue;

averageTime=parseInt(averageTime)+parseInt(afterValue);

}

}

\*/

$("#myTable2").append('<tr><td > <input type="text" value="' + "&nbsp Program " + programNumber + " runs in region " + regionToProgram + " from " + vperValue + " to " + vafterValue + ' " readonly size="60"> </td></tr>');

}

//average time counter

for (a = 0; a < afterValue.length; a++)

{

averageTime = parseInt(averageTime) + afterValue[a];

}

averageTime = parseInt(averageTime) / parseInt(numOfPrograms.length);

$("#myTable2").append('<tr><td > <input type="text" value="&nbsp Average time = ' + averageTime + '" readonly size="60"> </td></tr>');

}

//sorting 2 arrays

function insertionSort(sortTime, sortProgram) {

var len = sortTime.length, i = -1, ii = -1, j, tmp, tmp2;

while (len--) {

tmp = sortTime[++i];

tmp2 = sortProgram[++ii];

j = i;

while (j-- && parseInt(sortTime[j]) > parseInt(tmp)) {

sortTime[j + 1] = sortTime[j];

sortProgram[j + 1] = sortProgram[j];

}

sortTime[j + 1] = tmp;

sortProgram[j + 1] = tmp2;

}

return sortTime, sortProgram;

}

//sorting 5 arrays

function insertionSort2(sortTime, sortProgram, sortTimer, perValue, afterValue) {

var len = sortTime.length, i = -1, ii = -1, iii = -1, iiii = -1, iiiii = -1, j, tmp, tmp2, tmp3, tmp4, tmp5;

while (len--) {

tmp = sortTime[++i];

tmp2 = sortProgram[++ii];

tmp3 = sortTimer[++iii];

tmp4 = perValue[++iiii];

tmp5 = afterValue[++iiiii];

j = i;

while (j-- && parseInt(sortTime[j]) > parseInt(tmp)) {

sortTime[j + 1] = sortTime[j];

sortProgram[j + 1] = sortProgram[j];

sortTimer[j + 1] = sortTimer[j];

perValue[j + 1] = perValue[j];

afterValue[j + 1] = afterValue[j];

}

sortTime[j + 1] = tmp;

sortProgram[j + 1] = tmp2;

sortTimer[j + 1] = tmp3;

perValue[j + 1] = tmp4;

afterValue[j + 1] = tmp5;

}

return sortTime, sortProgram, sortTimer, perValue, afterValue;

}

//--------------------------------------------------------------------------\*

function reloader() {

location.reload();

}

</script>

</body>

</html>

