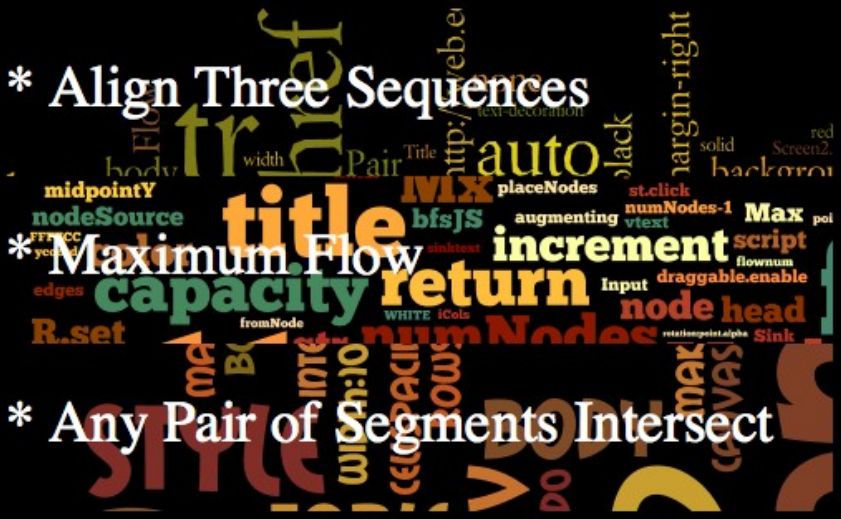
Algorithms

* Align Three Sequences



* Maximum Flow

* Any Pair of Segments Intersect

Table of Contents

Introduction: Running The Programs

Each program is an interactive web application written in PHP for the server-side, JavaScript for the client-side and marked-up using HTML for presentation to the user's browser. The programs may be run from either a menu or individually as indicated below.

Programs may be run via the web from this web address:

Menu	http://web.eecs.utk.edu/~eallen/ats_maxflow_apsi/

Also, individual programs may be run from the following web addresses:

Any Pair of Segments Intersect	http://web.eecs.utk.edu/~eallen/3apsi/
Aligning Three Sequences	http://web.eecs.utk.edu/~eallen/1ats/
Max Flow	http://web.eecs.utk.edu/~eallen/2maxflow/

Introduction: Examples

Screen-shots of each program output are shown in documentation section of each individual program.

Introduction: Documentation

Each program is documented separately and includes PHP, JavaScript, HTML and CSS mark-up.

The following tables show how each program emphasized an area of study in the study of algorithms. The table also shows pertinent functions which dove-tailed into the area of study.

Program	Area of Study	Function
Any Pair of Segments Intersect	Computational Geometry	ANY_SEGMENT_INTERSECTS calInter
Aligning Three Sequences	Dynamic Programming	traceback
Max Flow	Maximum Flow / Graph Algorithms	bfsJS

Function	Definition
ANY_SEGMENT_INTERSECTS	Manages segment endpoints and initializes a total pre-order array
calInter	Determine if two lines intersect.
traceback	Trace back through the table matrix after the table matrix has been scored.
bfsJS	Breadth-first search in Javascript

Screen Shots:

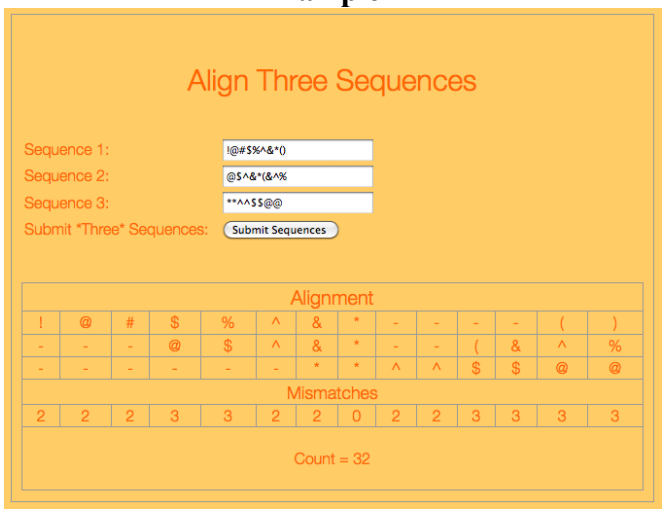
- Example 1
- Example 2
- Example 3

Code Documentation

Align Three Sequences: Screen Shots

Example 1

Example 2



Example 3

Example 4

Align Three Sequences Documentation

Align Three Sequences
PHP Class

```php
<?php

   class DEF{                    //Diagonal Edge Face

      public $dirF;
      public $val;

      function __construct($val, $dirF){
         $this->val  = $val;
         $this->dirF = $dirF;
      }

      function get_dirF() { return $this->dirF; }
      function get_val()  { return $this->val;  }

      function __destruct(){
      }
   }
?>
```

Align Three Sequences
Functions

```
<?php

   /* Return number of mismatches */
   function M(){
      $argu = array();
      for ($i = 0; $i < func_num_args(); $i++) {
         $argu[$i] = func_get_arg($i);
      }

      switch (func_num_args()){

         case 3:
            if (   (strcmp( $argu[0],$argu[1]) == 0) && (strcmp($argu[0],$argu[2]) == 0) && (strcmp($argu[1],
$argu[2]) == 0) ){
               $val = 0;
            }
            elseif((strcmp( $argu[0],$argu[1]) != 0) && (strcmp($argu[0],$argu[2]) != 0) && (strcmp($argu[1],
$argu[2]) != 0) ){
               $val = 3;
            }
            else{
               $val = 2;
            }
         break;
      }
      return $val;
   }

   function traceback($OPT1, $i, $j, $k, $x, $y, $z, &$seq0Array, &$seq1Array, &$seq2Array){

      //$i, $j, $k are the index of the matrix as we index backwards
      //$x, $y, $z are the index of the sequences as ww index backwards

      if ($seq0Array[$x] == $seq1Array[$y] && $seq0Array[$x] == $seq2Array[$z] && $seq1Array[$y] ==
$seq2Array[$z]){

         $ijkDVal = $OPT1[$i-1][$j-1][$k-1];
         $ijkDO   = new DEF($ijkDVal, 'ijkD');
         $checkSeven = array($ijkDO);
      }
      elseif($i != 0 && $j != 0 && $k != 0){
         /* ************************
          * Check Diagonal Direction
          * ************************/
         $ijkDVal = $OPT1[$i-1][$j-1][$k-1];
         $ijkDO   = new DEF($ijkDVal, 'ijkD');

         /*********************************
          * Check Faces Direction - Two gaps
```

```
     *********************************/
    $ijFVal = $OPT1[$i-1][$j-1][$k]; // ij-gap
    $ijFO   = new DEF($ijFVal, 'ijF');

    $ikFVal = $OPT1[$i-1][$j][$k-1];
    $ikFO   = new DEF($ikFVal, 'ikF');

    $jkFVal = $OPT1[$i][$j-1][$k-1];
    $jkFO   = new DEF($jkFVal, 'jkF');

    /*****************************************
     * Check Edges Direction  Diagonal  One gap
    *****************************************/
    $iEVal = $OPT1[$i-1][$j][$k];
    $iEO   = new DEF($iEVal,   'iE');

    $jEVal = $OPT1[$i][$j-1][$k];
    $jEO   = new DEF($jEVal,   'jE');

    $kEVal = $OPT1[$i][$j][$k-1];
    $kEO   = new DEF($kEVal, 'kE');

    $checkSeven = array($ijkDO,$ijFO,$ikFO,$jkFO,$iEO,$jEO,$kEO);
}
/*-----------------
 * Faces - Two gaps
 -----------------*/
elseif ($i == 0 && $j != 0 && $k != 0){
    //Face - Two gap
    $jkFVal = $OPT1[$i][$j-1][$k-1];
    $jkFO   = new DEF($jkFVal, 'jkF');

    //Edges Diagonal  One gap
    $jEVal = $OPT1[$i][$j-1][$k];
    $jEO   = new DEF($jEVal, 'jE');

    $kEVal = $OPT1[$i][$j][$k-1];
    $kEO   = new DEF($kEVal, 'kE');

    $checkSeven = array($jkFO,$jEO,$kEO);
}
elseif ($i != 0 && $j == 0 && $k != 0){
    //Face - Two gap
    $ikFVal = $OPT1[$i-1][$j][$k-1];
    $ikFO   = new DEF($ikFVal, 'ikF');

    //Edges Diagonal  One gap
    $iEVal = $OPT1[$i-1][$j][$k];
    $iEO   = new DEF($iEVal, 'iE');
```

```
    $kEVal = $OPT1[$i][$j][$k-1];
    $kEO   = new DEF($kEVal, 'kE');

    //Put the moves in an array and loop thru them to find the minimum
    $checkSeven = array($ikFO,$iEO,$kEO);
}
elseif ($i != 0 && $j != 0 && $k == 0){
    //Face - Two gap
    $ijFVal = $OPT1[$i-1][$j-1][$k]; // ij-gap
    $ijFO   = new DEF($ijFVal, 'ijF');

    //Edges Diagonal  One gap
    $iEVal = $OPT1[$i-1][$j][$k];
    $iEO   = new DEF($iEVal, 'iE');

    $jEVal = $OPT1[$i][$j-1][$k];
    $jEO   = new DEF($jEVal, 'jE');

    //Put the moves in an array and loop thru them to find the minimum
    $checkSeven = array($ijFO,$iEO,$jEO);
}
/*----------------
 * Edges - One gap
----------------*/
elseif ($i != 0 && $j == 0 && $k == 0){
    //Edges Diagonal  One gap
    $iEVal = $OPT1[$i-1][$j][$k];
    $iEO   = new DEF($iEVal, 'iE');

    //Put the moves in an array and loop thru them to find the minimum
    $checkSeven = array($iEO);
}
elseif ($i == 0 && $j != 0 && $k == 0){
    //Edges Diagonal  One gap
    $jEVal = $OPT1[$i][$j-1][$k];
    $jEO   = new DEF($jEVal, 'jE');

    //Put the moves in an array and loop thru them to find the minimum
    $checkSeven = array($jEO);
}
elseif ($i == 0 && $j == 0 && $k != 0){
    //Edges Diagonal  One gap
    $kEVal = $OPT1[$i][$j][$k-1];
    $kEO   = new DEF($kEVal, 'kE');

    //Put the moves in an array and loop thru them to find the minimum
    $checkSeven = array($kEO);
}
```

```
    return  $checkSeven;
}

function moveIndex($DirF, &$countSeq0, &$countSeq1, &$countSeq2){

    /*
     * Move the matrix index according to the direction DirF.
     */

    if ($DirF     == 'ijkD'){
        //Move
        $countSeq0 = $countSeq0-1;
        $countSeq1 = $countSeq1-1;
        $countSeq2 = $countSeq2-1;
    }
    elseif ($DirF == 'ijF'){
        $countSeq0 = $countSeq0-1;
        $countSeq1 = $countSeq1-1;
    }
    elseif ($DirF == 'ikF'){
        $countSeq0 = $countSeq0-1;
        $countSeq2 = $countSeq2-1;
    }
    elseif ($DirF == 'jkF'){
        $countSeq1 = $countSeq1-1;
        $countSeq2 = $countSeq2-1;
    }
    elseif ($DirF == 'iE'){
        $countSeq0 = $countSeq0-1;
    }
    elseif ($DirF == 'jE'){
        $countSeq1 = $countSeq1-1;
    }
    elseif ($DirF == 'kE'){
        $countSeq2 = $countSeq2-1;
    }
}

function adjSeqInd($DirF, &$seqIndex0, &$seqIndex1, &$seqIndex2){

    /*
     * Adjust the index of the sequences depending on the direction DirF
     */

    if ($DirF      == 'ijkD'){
        //Move
        $seqIndex0 = $seqIndex0-1;
        $seqIndex1 = $seqIndex1-1;
        $seqIndex2 = $seqIndex2-1;
```

```
    }
    elseif ($DirF  == 'ijF'){
       $seqIndex0 = $seqIndex0-1;
       $seqIndex1 = $seqIndex1-1;
    }
    elseif ($DirF  == 'ikF'){
       $seqIndex0 = $seqIndex0-1;
       $seqIndex2 = $seqIndex2-1;
    }
    elseif ($DirF  == 'jkF'){
       $seqIndex1 = $seqIndex1-1;
       $seqIndex2 = $seqIndex2-1;
    }
    elseif ($DirF  == 'iE'){
       $seqIndex0 = $seqIndex0-1;
    }
    elseif ($DirF  == 'jE'){
       $seqIndex1 = $seqIndex1-1;
    }
    elseif ($DirF  == 'kE'){
       $seqIndex2 = $seqIndex2-1;
    }
  }

  function writeCharGap($DirF, &$alignSeq0, &$alignSeq1, &$alignSeq2, $seq0Char, $seq1Char, $seq2Char)
{

    /*
     * Write the character | an insert character
     */

    if ($DirF     == 'ijkD'){
       $alignSeq0 = $alignSeq0.$seq0Char;
       $alignSeq1 = $alignSeq1.$seq1Char;
       $alignSeq2 = $alignSeq2.$seq2Char;
    }
    elseif ($DirF == 'ijF'){
       $alignSeq0 = $alignSeq0.$seq0Char;
       $alignSeq1 = $alignSeq1.$seq1Char;
       $alignSeq2 = $alignSeq2.'-';
    }
    elseif ($DirF == 'ikF'){
       $alignSeq0 = $alignSeq0.$seq0Char;
       $alignSeq1 = $alignSeq1.'-';
       $alignSeq2 = $alignSeq2.$seq2Char;
    }
    elseif ($DirF == 'jkF'){
       $alignSeq0 = $alignSeq0.'-';
       $alignSeq1 = $alignSeq1.$seq1Char;
```

```
        $alignSeq2 = $alignSeq2.$seq2Char;
    }
    elseif ($DirF == 'iE'){
        $alignSeq0 = $alignSeq0.$seq0Char;
        $alignSeq1 = $alignSeq1.'-';
        $alignSeq2 = $alignSeq2.'-';
    }
    elseif ($DirF == 'jE'){
        $alignSeq0 = $alignSeq0.'-';
        $alignSeq1 = $alignSeq1.$seq1Char;
        $alignSeq2 = $alignSeq2.'-';
    }
    elseif ($DirF == 'kE'){
        $alignSeq0 = $alignSeq0.'-';
        $alignSeq1 = $alignSeq1.'-';
        $alignSeq2 = $alignSeq2.$seq2Char;
    }
}

function getDirF($checkSeven){

    /*************************************************
    *  Find index of the the minimum value to use a key
    *************************************************/

    $minVal     = 9999; //set minimun to an outrageous value
    for ($x = 0; $x < count($checkSeven); $x++){
        if ($checkSeven[$x]->get_val() < $minVal){
            $minVal = $checkSeven[$x]->get_val();
            $iom    = $x;           // Index of the minumun
        }
    }
    $DirF = $checkSeven[$iom]->get_dirF();

    return $DirF;
}

function displayAlign($a0, $a1, $a2){

    /*
     * Display the results on the webpage
     */

    $a0 = strrev($a0);
    $a1 = strrev($a1);
    $a2 = strrev($a2);
    $count = 0;

    $tdMax = max(strlen($a0),  strlen($a1), strlen($a2));
```

```
        $td0 = $td1 = $td2 = '';
        $table0 = "<table class = 'table0' width='100%'>";
        $alignment = "<tr> <td style='font-size:20px' align='center' colspan='".$tdMax."'>Alignment</td></tr>";
        $td0 = '<tr>';

        for ($i = 0; $i < strlen($a0); $i++){
            $td0 = $td0.'<td>'.$a0[$i].'</td>';
        }

        $td0 = $td0.'</tr>';
        $td1 = '<tr>';

        for ($i = 0; $i < strlen($a1); $i++){
            $td1 = $td1.'<td>'.$a1[$i].'</td>';
        }

        $td1 = $td1.'</tr>';
        $td2 = '<tr>';

        for ($i = 0; $i < strlen($a2); $i++){
            $td2 = $td2.'<td>'.$a2[$i].'</td>';
        }

        $td2 = $td2.'</tr>';
        $td3 = '<tr>';
        $mmTotal = "<td style='font-size:18px' align='center' colspan='".$tdMax."'>Mismatches</td></tr>";
        $td3 = $td3.$mmTotal;

        for ($i = 0; $i < $tdMax; $i++){
            $td3 = $td3.'<td>'.$mmVal = M($a0[$i],$a1[$i], $a2[$i]).'</td>';
            $count = $count+$mmVal;
        }

        $td3 = $td3.'</tr>'."<tr><td style='font-size:16px' align='center' colspan='".$tdMax."'><br>Count = ".
$count."<br><br></td></tr>";
        $table0 = $table0.$alignment.$td0.$td1.$td2.$td3.'</table>';

        echo $table0;
    }
?>
```

Align Three Sequences
PHP Code / Logic

```
<?php
/*************************************************************************/

   /* Make sure all sequences are not blank */

   if($seq1 != "" && $seq2 != "" && $seq3 != ""){

      /* Convert the sequence strings to array */
      $seq0Array = str_split($seq1);   // string sequence along [x|i] axis
      $seq1Array = str_split($seq2);   // string sequence along [y|j] axis
      $seq2Array = str_split($seq3);   // string sequences along xyz axis

      $alignSeq0 = ''; // Alignment Sequences
      $alignSeq1 = '';
      $alignSeq2 = '';

      /* Get the count of the array elements */
      $countSeq0 = count($seq0Array);
      $countSeq1 = count($seq1Array);
      $countSeq2 = count($seq2Array);

      /* Global alignment algorithms start at the beginning of two sequences
       * and add gaps to each until the end of one is reached.
       *
       * (Local alignment algorithms finds the region (or regions) of highest
       * similarity between two sequences and build the alignment outward
       * from there.)
      */

      /* Make fill array $OPT */
      $OPT1 = array(array(array()));

      /* Initialize */
      for ($i = 0; $i < $countSeq0+1; $i++){
         for ($j = 0; $j < $countSeq1+1; $j++){
            for ($k = 0; $k < $countSeq2+1; $k++){
               $OPT1[$i][$j][$k] = 0;
            }
         }
      }

      /* Fill the fill array Row major order - fill rows from left to right */
      for($i = 1; $i < $countSeq0+1; $i++){
         for($j = 1; $j < $countSeq1+1; $j++){
            for($k = 1; $k < $countSeq2+1; $k++){

               // Scores
               $ijkM   = M($seq0Array[$i-1],$seq1Array[$j-1],$seq2Array[$k-1]);
               $ijM    = M($seq0Array[$i-1],$seq1Array[$j-1]);
```

```
            $ikM    = M($seq0Array[$i-1],$seq2Array[$k-1]);
            $jkM    = M($seq1Array[$j-1],$seq2Array[$k-1]);

            // Diagonal                          // Match
            $diLine = $OPT1[$i-1][$j-1][$k-1] + $ijkM;  // No-gap

            // Faces Diagonal                    // Two gaps
            $ijFace = $OPT1[$i-1][$j-1][$k] + $ijM + 2; // ij-gap
            $ikFace = $OPT1[$i-1][$j][$k-1] + $ikM + 2; // ik-gap
            $jkFace = $OPT1[$i][$j-1][$k-1] + $jkM + 2; // jk-gap

            // Edges Diagonal                    // One gap
            $iEdge = $OPT1[$i-1][$j][$k] + 2;          // i-gap
            $jEdge = $OPT1[$i][$j-1][$k] + 2;          // j-gap
            $kEdge = $OPT1[$i][$j][$k-1] + 2;          // k-gap

            $OPT1[$i][$j][$k] = min( $diLine, $ijFace, $ikFace, $jkFace, $iEdge,$jEdge, $kEdge);
        }
    }
}

/*
 * Match the last diagonal array indexes with the corresponding sequence
 * index
 */
$seqIndex0 = $countSeq0-1;
$seqIndex1 = $countSeq1-1;
$seqIndex2 = $countSeq2-1;

while (1){
    if ($countSeq0 == 0 && $countSeq1 == 0 && $countSeq2 == 0){
        break;
    }
    else{

        //******************Traceback*********************************
        $checkSeven = traceback($OPT1,
                $countSeq0, $countSeq1, $countSeq2,
                $seqIndex0, $seqIndex1, $seqIndex2,
                $seq0Array, $seq1Array, $seq2Array);

        //Get the variable for the direction to move/adjust the matrix and
        //sequence
        $DirF = getDirF($checkSeven);

        //Write the character or gap
        writeCharGap($DirF,
                $alignSeq0, $alignSeq1, $alignSeq2,
                $seq0Array[$seqIndex0], $seq1Array[$seqIndex1], $seq2Array[$seqIndex2]);
```

```
            //Move the matrix index
            moveIndex($DirF,
                $countSeq0, $countSeq1, $countSeq2);

            //Adjust the sequence index
            adjSeqInd($DirF,
                $seqIndex0, $seqIndex1, $seqIndex2);
        }
    }
    $_SESSION['DISPLAY'] = TRUE;
  }
?>
```

Align Three Sequences
HTML

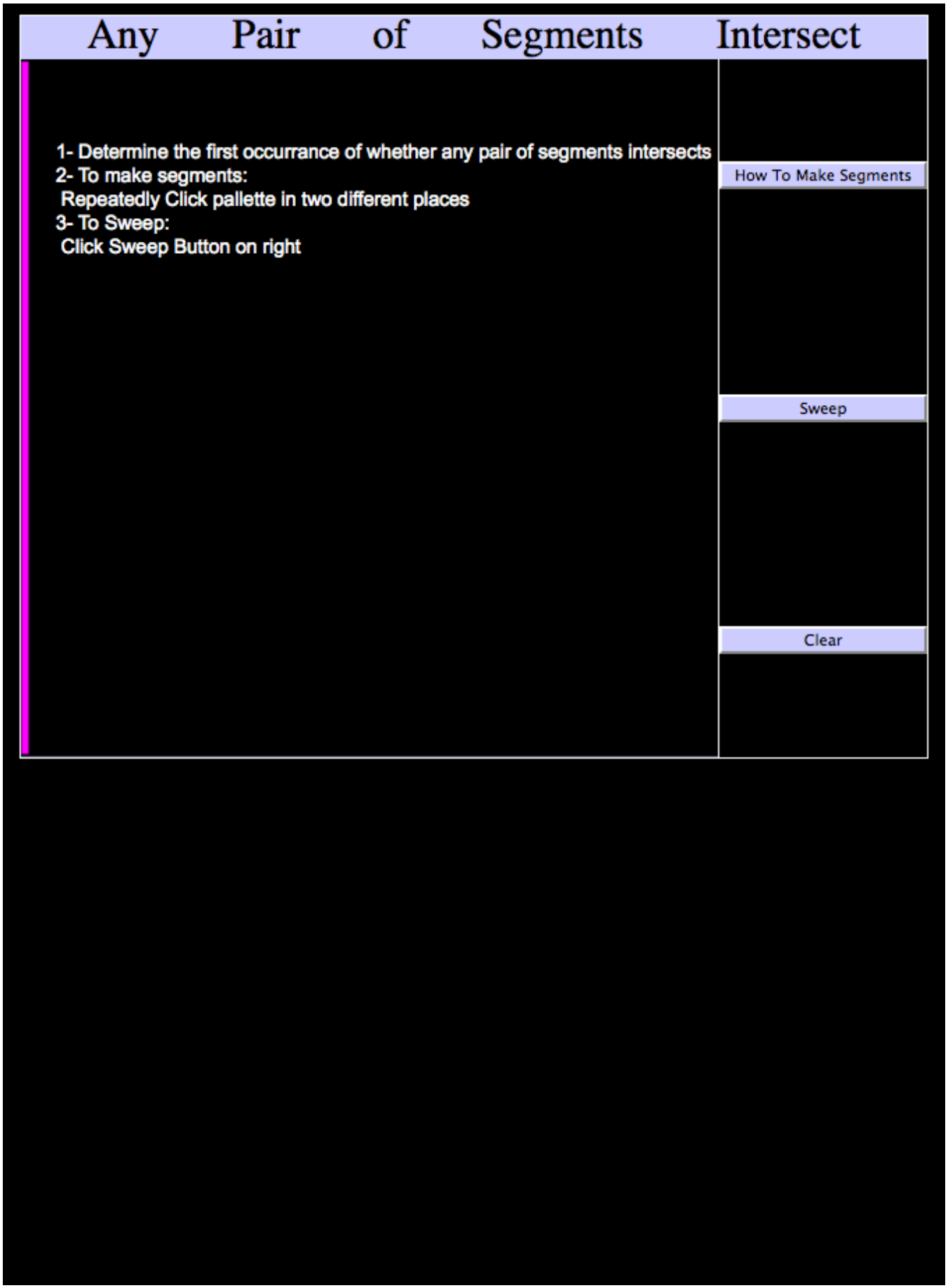
```
<?php
   session_start();             // Start a session
   error_reporting(E_ALL);      // Report all session error
   ini_set('display_errors',1);  // Display all session errors

   $_SESSION['DISPLAY'] = FALSE; //for Displaying in web page

   $seq1 = $_POST["seq1"];      //Get variables off url
   $seq2 = $_POST["seq2"];
   $seq3 = $_POST["seq3"];

   include ("Dir.php");         //Php classes which

   include ("Functions.php");   //Traceback is here

   include ("Logic.php");       //The logic which loops and moves the matrix and sequence indexes
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <title>Aligning Three Sequences</title>
      <link rel="stylesheet" href="1ats.css" type="text/css">
   </head>
   <body style="z-index: 0">
      <form method="POST" enctype="multipart/form-data" action="index.php">
         <div id="holder">
            <div id ="title"> <br>Align Three Sequences<br><br></div>
            <table  width="100%" border="0">
               <tr>
                  <td class="dwidth">Sequence 1:</td>
                  <!-- <td id="seq1"><input type="text" name="seq1" value='VSNS'></td> -->
                  <td id="seq1"><input type="text" name="seq1"></td>
               </tr>
               <tr>
                  <td>Sequence 2:</td>
                  <!-- <td id="seq2"><input type="text" name="seq2" value='SNA'></td> -->
                  <td id="seq2"><input type="text" name="seq2"></td>
               </tr>
               <tr>
                  <td>Sequence 3:</td>
                  <!-- <td id="seq3"><input type="text" name="seq3" value='AS'></td> -->
                  <td id="seq3"><input type="text" name="seq3"></td>
               </tr>
               <tr>
                  <td id="seq4">Submit *Three* Sequences:</td>
                  <td><input type="submit" value="Submit Sequences "></td>
               </tr>
```

```
            </table>
            <br><br>
               <?php
                  if ($_SESSION['DISPLAY'] == TRUE){
                     displayAlign($alignSeq0, $alignSeq1, $alignSeq2);
                  }
               ?>
         </div>
      </form>
   </body>
</html>
```

Align Three Sequences
CSS

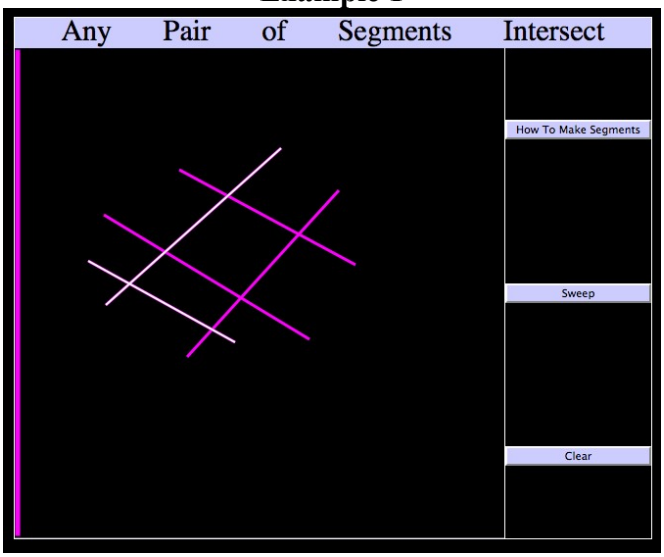
```
/*
    Document   :Align Three Sequence
    Created on : Jul 23, 2011, 12:21:38 PM
    Author     : anthonyallen
    Description:
        Purpose of the stylesheet follows.
*/

/*
   TODO customize this sample style
   Syntax recommendation http://www.w3.org/TR/REC-CSS2/
*/

root {
    display: block;
}

body {
    background-color: #ffcc66;
    color: #ff6600;
    font: 300 100.1% "Helvetica Neue", Helvetica, "Arial Unicode MS", Arial, sans-serif;

}
#holder {
    height: 480px;
    width:  640px;
    left: 320px;
    top:  125px;
    padding: 10px;

    position: absolute;
    border: 1px solid #999999;

}
.dwidth{
    width: 200px;
}
#title{
    margin-left: auto;
    margin-right: auto;
    text-align: center;
    border: 2px none red;
    margin-top: 0px;
    font-size: 30px;
}
table.table0{
        border: 1px solid #999999;
        border-collapse: collapse;
        text-align: center;
```

```
}
table.table0 td{
	border: 1px solid #999999;
}
```

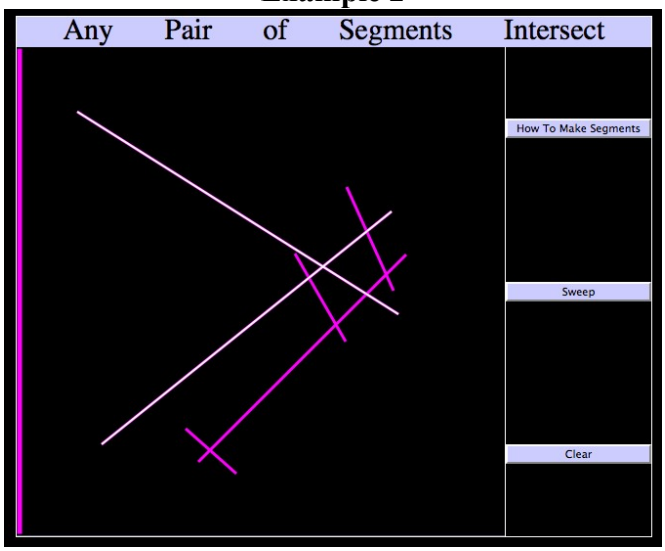

Any Pair of Segments Intersect
1- Determine the first occurrance of whether any pair of segments intersects
2- To make segments:
Repeatedly Click pallette in two different places
3- To Sweep:
Click Sweep Button on right
How To Make Segments
Sweep
Clear

Any Pair Of Segments Intersect

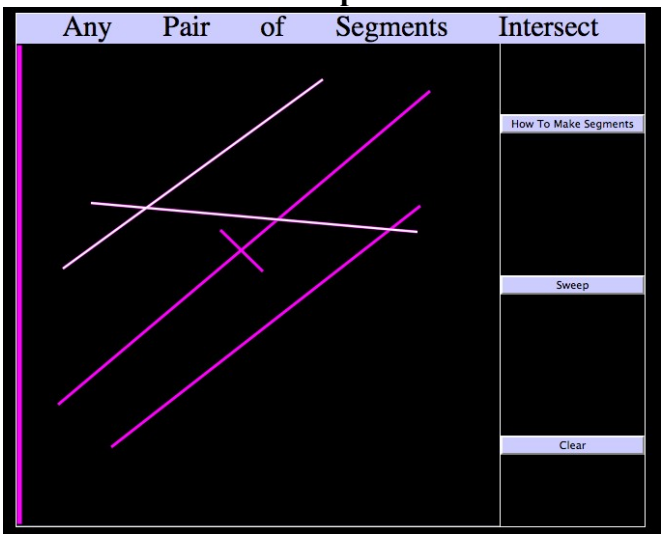
Example 1



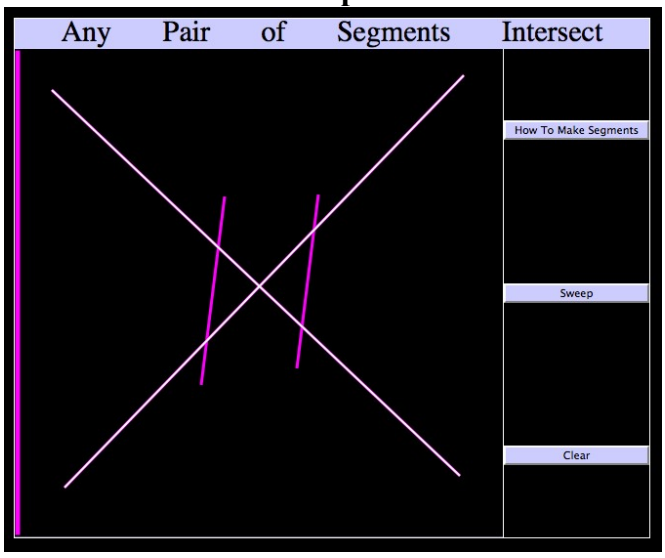
Example 2



Example 3



Example 4



Any Pair Of Segments Intersect Documentation

Any Pair Of Segments Intersect
Javascript Logic and Functions

```
<script type="text/javascript" charset="utf-8">

    //Globals - Yikes!

    var count = 0;
    var segIndex = 0;
    var data     = '';
    var data1    = '';
    var Mx = 0;
    var My = 0;
    var Lx = 0;
    var Ly = 0;

    var paper      = null;
    var scanLine   = null;
    var init_instr = null;

    var getMPFlag  = 1;

    var S;
    var segArray;
    var ans;

    window.onload = function(){

        S = new Array();        //Endpoints of segments in S
        segArray = new Array(); //Segment Array

        //Make a pallette
        paper = new Raphael(document.getElementById('canvas_container'),500,500);

        //Make shapes
        //Rectangle
        var linePallette = paper.rect(0, 0, 500,500,0);
        linePallette.attr({fill: '#000', stroke: 'none'});

        //Instructions
        init_instr = paper.text(25, 100, '1- Detrmine the first occurrance of whether any pair of segments
intersects\n2- To make segments:\n   Repeatedly Click pallette in two different places\n3- To Sweep:\n   Click
Sweep Button on right');
        init_instr.attr({opacity: 1, 'font-size': 14,'fill':'#FFF','text-anchor':'start'}).toFront();

        //Scan line
        scanLine = paper.path('M 3 2 L 3 498').attr({stroke:'#f0f','stroke-width':5});

        //Mouse over to get cursor style
        linePallette.node.onmouseover = function(){
           this.style.cursor = 'pointer';
        }
```

```
    init_instr.node.onmouseover = function(){
        this.style.cursor = 'pointer';
    }

    scanLine.node.onmouseover = function(){
        this.style.cursor = 'pointer';
    }

    //Click on line pallette or instructions to remove instructions
    linePallette.node.onclick = function(event) {
        if (getMPFlag == 1){
            getMousePos(event);
            init_instr.animate({opacity: 0}, 2000,function(){
            init_instr.hide();
            });
        }
    }

    init_instr.node.onclick = function(){
        init_instr.animate({opacity: 0}, 2000,function(){
            this.hide();});
    }

    scanLine.node.onclick = function(){
    //no getting mouse position while scanning
        getMPFlag = 0;
        init_instr.animate({opacity: 0}, 2000,function(){
            this.hide();});
        scanLine.animate({translation:'500 0'},4000,function(){
            ans = ANY_SEGMENTS_INTERSECTS(S,segArray);
            scanLine.animate({translation:'-500 0'},1,function(){
                getMPFlag = 1;
                drawinter(ans);
            });})
    }
}

function ANY_SEGMENTS_INTERSECTS(S,segArray){

    var T = new Array()//Total Order Array
    for (var i=0;i<S.length;i++){
        if (S[i].e == 0){  //left endpoint
            //tag it to
            S[i].tag = '*'; //tag it to obtain locatin in T
            Insert(T,S[i]);

            //Get index of *
            var star = getStarIndex(T);
```

```
            var aboveArray = Above(T,star,segArray);
            if (aboveArray != -1){

               //Calculate intersection
               if (calInter(aboveArray,segArray[S[i].segIndex])) return (aboveArray
+"|"+segArray[S[i].segIndex]);
            }
            var belowArray = Below(T,star,segArray);
            if (belowArray != -1){

               //Calculate intersection
               if (calInter(belowArray,segArray[S[i].segIndex]))return
(belowArray+"|"+segArray[S[i].segIndex]);
            }
            S[i].tag = '';
         }
         else if (S[i].e == 1){ // right endpoint
            var segIndex = S[i].segIndex;
            showQ(T);
            showQ(S);
            for(var k=0; k < T.length; k++){
               if (T[k].segIndex == segIndex){
                  T.splice(k,1);
                  showQ(T);
               }
            }

            //tag it to
            S[i].tag = '*'; //tag it to obtain locatin in T
            Insert(T,S[i]);
            showQ(T);
            //Get index of *
            var star = getStarIndex(T);
            var aboveArray = Above(T,star,segArray);
            var belowArray = Below(T,star,segArray);

            if(((aboveArray != -1) && (belowArray != -1)) && (calInter(belowArray,aboveArray))){
               return (belowArray+"|"+aboveArray);
            }
            else{
               star = getStarIndex(T);
               T.splice(star,1);
            }
            S[i].tag = '';
         }
      }
      return false;
   }
```

```
function showQ(T){

    var data = '';
    for(var n=0; n<T.length; n++){
        data = data + T[n].x+" "+T[n].y+" "+T[n].e+" "+T[n].segIndex+" "+T[n].tag+'\n';
    }
}

function calInter(abA,segA){

    // http://www.johanvanmol.org/content/view/39/37/1/2/

   //get the points
    var S1 = abA.split(" ");
                        //   0   1    2    3   4     5
    var S1x = S1[4] - S1[1]; //   M   x    y    F   x     y
    var S1y = S1[5] - S1[2]; // S1    xp1  yp1      xp2   yp2
                        // S2    xp3  yp3      xp4   yp4
    var S2 = segA.split(" ");
    var S2x = S2[4] - S2[1];
    var S2y = S2[5] - S2[2];

      //s = (-S1y * (xp1 -   xp3)   + S1x * (yp1   - yp3))   / (-S2x*S1y + S1x*S2y)
    var s = (-S1y * (S1[1] - S2[1]) + S1x * (S1[2] - S2[2])) / (-S2x*S1y + S1x*S2y);

    //t   = (S2x * (yp1   - yp3)   - S2y * (xp1   - xp3))   / (-S2x*S1y + S1x*S2y)
    var t = (S2x * (S1[2] - S2[2]) - S2y * (S1[1] - S2[1])) / (-S2x*S1y + S1x*S2y);

    if (((0 < s) && (s < 1)) && ((0 < t) && (t < 1)))
        return true;
    else
        return false;
}

function getStarIndex(T){

    for (var i=0;i<T.length;i++){

        if (T[i].tag =="*") break;
    }
    return i;
}
function makeline(){

    init_instr.animate({opacity: 1}, 200,function(){
        init_instr.show();
    });
}
```

```
    function scan(){
       //no getting mouse position while scanning
          getMPFlag = 0;
          init_instr.animate({opacity: 0}, 2000,function(){
             this.hide();});
          scanLine.animate({translation:'500 0'},4000,function(){
             ans = ANY_SEGMENTS_INTERSECTS(S,segArray);
             scanLine.animate({translation:'-500 0'},1,function(){
                getMPFlag = 1;
                drawinter(ans);
             });});
    }

    function drawinter(ans){

       line = new Array();
       if(ans == false){
          alert("No Intersections Found");
       }
       else{
          line = ans.split("|");
          var linecolor='#FFF';
          paper.path(line[0]).attr({'stroke-width': '2', stroke:linecolor}); // Do not set stroke-width:1 makes
phantom vertical line! when scan line
          paper.path(line[1]).attr({'stroke-width': '2', stroke:linecolor}); // Do not set stroke-width:1 makes
phantom vertical line! when scan line
       }
    }

    function getMousePos(e){
       if (!e) var e = window.event;
       //alert("// W3C - FF, Safari, IE 9 and most of the world");
       if (e.pageX || e.pageY){
          posx = e.pageX;
          posy = e.pageY;

          var tableMargin = document.getElementById("tab0");
          var tableMarginAttr = document.defaultView.getComputedStyle(tableMargin, null);
          tableMarginH = tableMarginAttr.marginTop;

          tableMarginL = tableMarginAttr.marginLeft;

          var titleTd = document.getElementById("td0");
          var titleTdAttr = document.defaultView.getComputedStyle(titleTd, null);
          titleTdHeight  = titleTdAttr.height;

          posx = parseInt(posx)-parseFloat(tableMarginL);
          posy = parseInt(posy) - parseInt(titleTdHeight) - parseFloat(tableMarginH);
       }
```

```
else if (e.clientX || e.clientY){
   posx = e.clientX + document.body.scrollLeft + document.documentElement.scrollLeft;
   posy = e.clientY + document.body.scrollTop  + document.documentElement.scrollTop;

   //need to subtract the body margin
   bodyleft = document.getElementById('bod').style.marginLeft;
   bodytop =  document.getElementById('bod').style.marginTop;

   //convert margins into numbers margin is for example 100px not 100
   bodyleft = pxToNum(bodyleft);
   bodytop  = pxToNum(bodytop);
   posx -= bodyleft;
   posy -= bodytop;
}

if (count==0){
   count++;
   Mx = posx;
   My = posy;
}
else{
   count=0;
   Lx = posx;
   Ly = posy;

   if(Mx > Lx){
      //Swap the points
      var tempx;
      var tempy
      tempx = Mx;
      Mx    = Lx;
      Lx    = tempx;
      tempy = My;
      My    = Ly;
      Ly    = tempy;
   }

   segment = new lineObj();
   segment.x = Mx;
   segment.y = My;
   segment.e = 0; //startpoint - left endpoint
   segment.segIndex = segIndex;
   S.push(segment);


   segment = new lineObj();
   segment.x = Lx;
   segment.y = Ly;
   segment.e = 1; //endpoint - right endpoint
```

```
        segment.segIndex = segIndex;
        S.push(segment);
        segIndex++;

        // Array of line objects
        segArray.push("M "+ Mx+" "+My+" L "+Lx+" "+Ly);

       drawLine();
       S.sort(function(a,b){return a.x - b.x});
    }
}

function Insert(T,S){

    var data = "";
    var test = "";
    T.push(S);
    T.sort(function(a,b){return a.y - b.y});
}

function Delete(){
}

function Above(T,index,segArray){

    if (index-1 != -1){
        showQ(T);

        return segArray[T[index-1].segIndex];
    }
    else{
        showQ(T);
        return -1;
    }
}

function Below(T,index,segArray){
    if ( index+1 <= T.length-1 ){
        showQ(T);

        return segArray[T[index+1].segIndex];
    }
    else{
        showQ(T);
        return -1;
    }
}

function lineObj(x, y, e, segIndex,tag){
```

```
        this.x = x; this.y = y;
        this.e = e; this.segIndex = segIndex;
        this.tag = tag; // *
    }

    function pxToNum(px){
        var temp = new Array();
        temp = px.split('px');
        return temp[0];
    }

    function drawLine(){
        var linecolor='#F0F';
        var line = "M " + Mx + " " + My + " L " + Lx + " " + Ly;
        paper.path(line).attr({'stroke-width': '3', stroke:linecolor}); // Do not set stroke-width:1 makes phantom
vertical line! when scan line
    }

</script>
```

Any Pair Of Segments Intersect
HTML

Any Pair Of Segments Intersect
CSS

```
/*
    Document   : 3apsi
    Created on : Jul 13, 2011, 2:46:28 AM
    Author     : anthonyallen
    Description:
        Purpose of the stylesheet follows.
*/

/*
   TODO customize this sample style
   Syntax recommendation http://www.w3.org/TR/REC-CSS2/
*/

root {
    display: block;
}

#canvas_container{
    width: 500px;
    height: 500px;
    background-color:    #ccccff;
    border-right:  1px solid white;
}

#instructions{
     width: 900px;
     height: 500px;
     border: 2px red solid;
}

button{
    background-color: #ccccff;
}

#heading{
    text-align: left;
    width: 500px;
    color: white;
    font-size: X-large;
    background-color: black;
}

#bod{
    background-color: black;
    margin-top: 0px;
    margin-left: 0px;
}

td.td1{
```

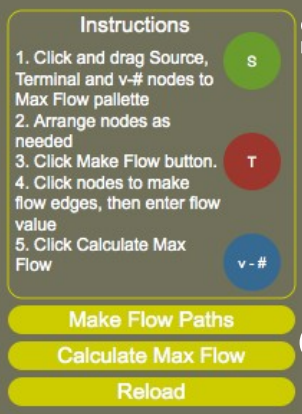
```
    height: 166px;
    border-bottom: none;
    border-left: none;
    border-top-color:white;
}

td.td2{
    height: 166px;
    border-top: none;
    border-bottom: none;
    border-left: none;
}

td.td3{
    height: 166px;
    border-top: none;
    border-left: none;
    border-bottom-color: white;

}

#td0{
    word-spacing: 1.5em;
    background-color: #ccccff;
    font-size: 30px;
    text-align: center;
    border-right-color:white;
    border-left-color:white;
}

#tab0{
    margin-left: 25%;
    margin-top: 5%;
    border: 1px solid white;
    border: 1px solid white;
}
```

Max Flow

Screen Shots:
- Example 1
- Example 2
- Example 3

Code Documentation

Max Flow



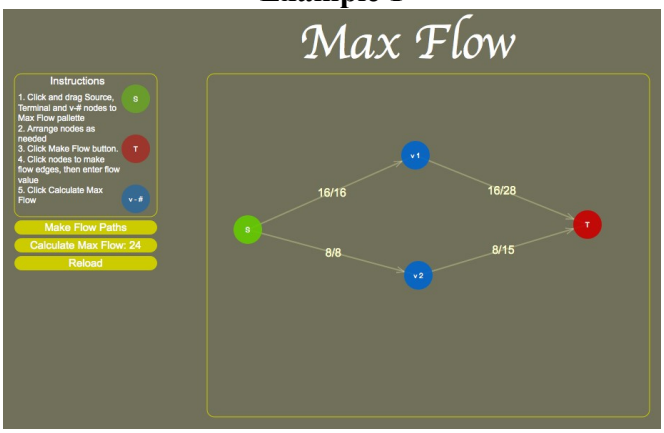
Screen Shots:

- Example 1
- Example 2
- Example 3

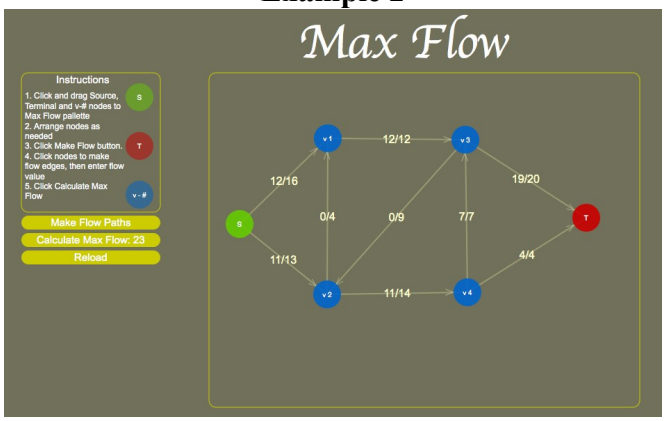
Code Documentation

Max Flow

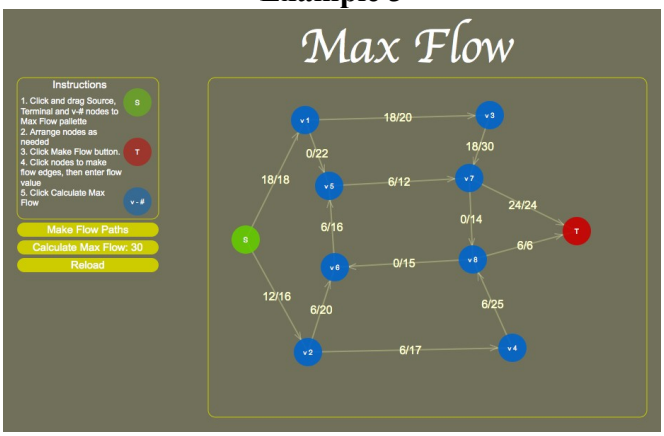
Example 1



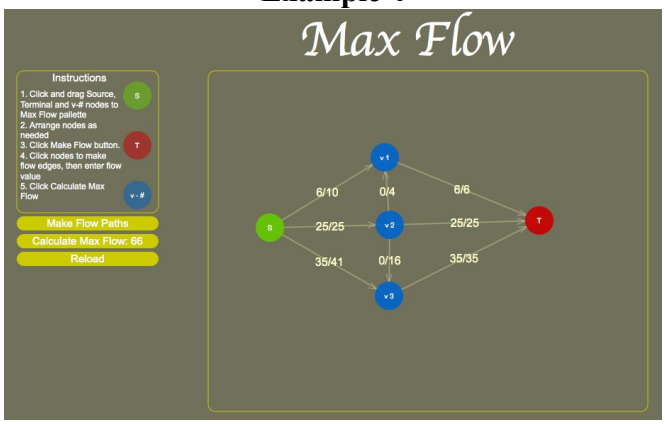
Example 2



Example 3



Example 4



Max Flow Documentation

Max Flow
Javascript Logic and Functions
and HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <title>2maxflow</title>
      <link rel="stylesheet" href="2maxflow.css" type="text/css">
      <script src="raphael-min.js"       type="text/javascript" charset="utf-8"></script>
      <script src="raphael.draggable.js" type="text/javascript" charset="utf-8"></script>
      <script type="text/javascript" charset="utf-8">

         var nodeSet = new Array();
         //var circles = [];
         var i = 0;
         var sourceflag = false; // only one source
         var sinkflag = false;   // only one sink
         //var R = null;
         var g = null;
         var r = null;
         var p = null;
         var obj1 = null;
         var vtext = null;
         var sourcetext = null;
         var sinktext = null;
         var startSet = null;
         var makeFlowflag = false;
         var pointCount = 0;

         var faketext = null;
         var fake = null;
         var fakeSet = null;
         var obj = null;

         var numEdges = 0;    //Number of edges
         var numNodes = 0;    //Number of nodes
         var nodeCheck = '';  //String of Nodes checked T:S:v-1:v-2: ...

         var toNode,fromNode; // for input file

         var capacityFromTo = []; //array hold from node to node capacity

         var midpointFromTo = []; //array to hold from to midpoint x y



         //var inputFile = [];

         var inputfile=[];
         var nodeSink;              //sink
         var nodeSource=0;
```

```
var arrayAugPath = [];
var WHITE     = 0; //False
var GRAY      = 1; //True
var BLACK     = 2; //True
var MAX_NODES = 1000;

var e;  // See numEdges
//var capacity[MAX_NODES][MAX_NODES]; // capacity matrix
var flow = [];     // flow matrix

var color =  [];  // needed for breadth-first search
var pred  =  [];  // array to store augmenting path augPath

var head;
var tail;

var q = [];

var yes;

var pathArray = [];

window.onload = function () {

   var corners = 10;

   R = Raphael(0, 0, "100%", "100%"); // Make the canvas
   R.draggable.enable();              // enable object to be dragged

   var instructions1 = "1. Click and drag Source,\nTerminal and v-# nodes to\nMax Flow pallette\n";
   var instructions2 = "2. Arrange nodes as\nneeded\n";
   var instructions3 = "3. Click Make Flow button.\n";
   var instructions4 = "4. Click nodes to make\nflow edges, then enter flow\nvalue\n";
   var instructions5 = "5. Click Calculate Max\nFlow";
   var instructions = instructions1+instructions2+instructions3+instructions4+instructions5;

   var title = R.text(600,80,"Max Flow").attr({'font-size':'70px','font-family':'cursive',fill:"#FFF","text-
anchor":"Title"});

   var rectinst = R.rect(50,125,200,200,corners).attr({stroke:"#cccc00"});
   var pallette = R.rect(320,125,620,480,corners).attr({stroke:'#cccc00'});
   var textinst = R.text(100,135,"Instructions").attr({'font-size':'15px',fill:"#FFF","text-anchor":"start"});
   var textdirections = R.text(55,230,instructions).attr({'font-size':'12px',fill:"#FFF","text-
anchor":"start"});

   var makeFlows    = R.rect(50,330,200,20,corners).attr({fill:'#cccc00',stroke:'none'});
   var makeFlowsText = R.text(150,340,"Make Flow Paths").attr({'font-
size':'15px',fill:"#FFF",title:"Make Flows"});
```

```
        var makeInputFileButton    = R.rect(50,355,200,20,corners).attr({fill:'#cccc00',stroke:'none'});
        var makeInputFileText      = R.text(150,365,"Make Input File").attr({'font-
size':'15px',fill:"#FFF",title:"Make Input File"});

        var readInputFileButton    = R.rect(50,380,200,20,corners).attr({fill:'#cccc00',stroke:'none'});
        var readInputFileText      = R.text(150,390,"Read Input File").attr({'font-
size':'15px',fill:"#FFF",title:"Read Input File"});

        //var oldcal = 'rect 50,405,200,20: text 150,415';
        //var newcal = '';
        var calculateMaxFlowButton = R.rect(50,355,200,20,corners).attr({fill:'#cccc00',stroke:'none'});
        var calculateMaxFlowText   = R.text(150,365,"Calculate Max Flow").attr({'font-
size':'15px',fill:"#FFF",title:"Calculate Max Flow"});

        var reloadButton = R.rect(50,380,200,20,corners).attr({fill:'#cccc00',stroke:'none'});
        var reloadText   = R.text(150,390,"Reload").attr({'font-size':'15px',fill:"#FFF",title:"Reload Page"});



        // Make initial templates for source, terminal, and nodes
        //
        g = R.circle(220, 160, 20).attr({fill: "#66CC00", stroke: "none", opacity: .5,title:"S-Source"});   //
Green
        r = R.circle(220, 230, 20).attr({fill: "#CC0000", stroke: "none", opacity: .5,title:"T-Terminal"}); // Red
        p = R.circle(220, 300, 20).attr({fill: "#0066CC", stroke: "none", opacity: .5,title:"Nodes - #"});

        // Text for initial templates for source, terminal, and nodes
        //
        vtext      = R.text(220,300,"v - #").attr({fill: "#FFF",title:"Nodes"});
        sourcetext = R.text(220,160,"S").attr({fill:"#FFF",title:"Source"});     //Source green
        sinktext   = R.text(220,230,"T").attr({fill:"#FFF",title:"Sink"});       //Sink red

        sourceSet = R.set();
        readInputFileButtonSet = R.set()
        startSet              = R.set();
        inputFileButtonSet    = R.set();
        calculateMaxFlowSet   = R.set();
        reloadSet             = R.set();

        calculateMaxFlowSet.push(
           calculateMaxFlowButton.attr({cursor:"pointer"}),
           calculateMaxFlowText.attr({cursor:"pointer"})
        )

       reloadSet.push(
           reloadButton.attr({cursor:"pointer"}),
           reloadText.attr({cursor:"pointer"})
        )
```

```
        sourceSet.push(
            g,
            sourcetext
        )

        readInputFileButtonSet.push(
            readInputFileButton.attr({cursor:"pointer"}).hide(),
            readInputFileText.attr({cursor:"pointer"}).hide()
        )

        inputFileButtonSet.push(
            makeInputFileButton.attr({cursor:"pointer"}).hide(),
            makeInputFileText.attr({cursor:"pointer"}).hide()
        )

        startSet.push(
            makeFlows.attr({cursor:"pointer"}),
            makeFlowsText.attr({cursor:"pointer"})
        );

        // Bind templates graphics to mouse events
        //

        reloadSet.click(function(){
            window.location.reload();

        });

        calculateMaxFlowSet.click(function(){
            var inputFile = [];
            var capacity  = [];
            var max_flow  = [];
            //
            inputFile = makeInputFile(numNodes, numEdges, capacityFromTo);  // Changes nodes text to node
numbers. source->0 etc

            capacity = readInputFile(numNodes, numEdges, inputFile);        // capacityFromTo now consist of
indexes not text such as S

            max_flow = calculateMaxFlow(nodeSource, nodeSink, capacity);    // and it is any array of From
To Capacity, From To Capacity

            updateWithFlow(numNodes, max_flow, capacity, midpointFromTo,calculateMaxFlowSet );
        });

        readInputFileButtonSet.click(function(){
            readInputFile(numNodes,numEdges,capacityFromTo);
        });
```

```
    inputFileButtonSet.click(function(){
        makeInputFile(numNodes, numEdges,capacityFromTo);

    });

    startSet.click(function(){
        makeFlowflag = true;
        return;
    });

    r.mousedown(function(){
        makeSink();
        return;
    });

    g.mousedown(function(){
        makeSource();
        return;
    });

    p.mousedown(function(){
        return makeNode();
    });
}

function updateWithFlow(numNodes, flow, capacity, midpointXY,calculateMaxFlowSet){

    //alert ('updateWithFlow\nMax Flow: '+flow[0]);

    //alert(calculateMaxFlowSet[1].attr('text')+': '+flow[0]);

    calculateMaxFlowSet[1].attr('text',calculateMaxFlowSet[1].attr('text')+': '+flow[0]);

    for(var p in midpointXY){
        objR = midpointXY[p];
        //alert('p: '+p);
        newKey = p.replace(/T/g,numNodes-1);
        midpointXY[newKey] = objR;

        //alert('newKey: '+newKey+' objR: '+ objR);
        //delete midpointXY[p];
    }


    for (var i = 0;i < numNodes; i++){
        for (var j = 0; j < numNodes; j++){
            if(capacity[i][j] > 0){
                var capFlow = flow[1][i][j]+'/'+capacity[i][j];
```

```
                    var fromto = i+''+j;
                    midStr  = midpointXY[fromto];
                    //alert('i j '+i+' '+j+' flow: '+flow[1][i][j]+'\ncapacity '+capacity[i][j]+'\ncapFlow '+capFlow+'
midStr: '+midStr);
                    midStr.attr('text',capFlow);
                    //midStr.attr({text: 'nifty new text here'});
                }
            }
        }
    }

    function min ( x, y) {
        return x < y ? x : y;  // returns minimum of x and y
    }


    //Queue for for bfsJS just uses one queue
    //Putting in the queus
    function enqueue (x) {
        q[tail] = x;        //put the parent's children at the end of the queue
        tail++;             //increment the pointer / index
        color[x] = GRAY;    //color the element Gray - unvisited children
        return tail;
    }

    //List for bfsJS
    //Removing from the queue
    function dequeue () {
        var x = q[head];     //get the element from the front head of the queue/list
        head++;              //increment the pointer
        color[x] = BLACK;    //color the element Black - visited
        return x;            //return that element - what is it
    }

    function remEle(arrayName,arrayElement){
        //http://www.roseindia.net/java/javascript-array/javascript-remove-an-element.shtml

        for(var i=0; i<arrayName.length;i++ ){
            if(arrayName[i]==arrayElement){
                arrayName.splice(i,1);
                //alert('remEle '+arrayName[i]);
            }
        }
        return arrayName;
    }

    function eliDup(arr) {
      // http://dreaminginjavascript.wordpress.com/2008/08/22/eliminating-duplicates/
```

```
    var i,
    len=arr.length,
    out=[],
    obj={};

    for (i=0;i<len;i++) {
      obj[arr[i]]=0;
    }

    for (i in obj) {
      out.push(i);
    }

    return out;
}

function calculateMaxFlow(nodeSource, nodeSink, capacity){

    var maxFlow_flowArray = [];
    var u;
    // Initialize empty flow.
    var max_flow = 0;
    var flow = [];
    // Initialize the flow array with zeros.
    flow = MultiDimensionalArray(numNodes, numNodes);

    // While there exists an augmenting path,
    // increment the flow along this path.

    while ( bfsJS(nodeSource, nodeSink, capacity, flow) ){
      // Determine the amount by which we can increment the flow.
      var increment = 1000000; // this a very large value infinity
      for (u = numNodes-1; pred[u] >= 0; u = pred[u]) {          //pred[u] is the augmenting path
        increment = min(increment, capacity[pred[u]][u] - flow[pred[u]][u]); // the capcity
      }

      // Now increment the flow.
      for (u = numNodes-1; pred[u] >= 0; u = pred[u]) {

        flow[pred[u]][u] += increment;
        flow[u][pred[u]] -= increment;

      }

      max_flow += increment;

    }

    maxFlow_flowArray.push(max_flow,flow);
```

```
    return maxFlow_flowArray;

}

function bfsJS(start, target, capacity, flow ){

  //alert('Hello BFS');

  //alert('start '+start+' target '+target+' numNodes '+numNodes);

   var u; // The from node
   var v; // The to node

   for (u = 0; u < numNodes; u++) { // initialize the coloring array White. Note: Black = unvisited
      color[u] = WHITE;           // Gray = visited
   }

   head = tail = 0;      // Iitialize the queue The head and tail start together
   enqueue(start);       // Put the start node / source node in the queue
   pred[start] = -1;     // pred - array for the augumenting path

   while (head != tail){ // as long as the head and tail don't meet keep going
      u = dequeue();    // get the node from the queue
                        // Search all adjacent white nodes v. If the capacity
                        // from u to v in the residual network is positive,
                        // enqueue v.

      for (v = 0; v < numNodes; v++){
         if (color[v] == WHITE  &&  capacity[u][v]-flow[u][v] > 0) { //if the color is White
            enqueue(v);  // put the node in the queue
            pred[v] = u; // create the augmenting path

            // Don't need the paths when flows < 0;
            if(flow[u][v] >= 0){
               //alert ('**path: '+u+"+v+' '+(capacity[u][v]-flow[u][v])+' capacity: '+capacity[u][v]+' flow:
'+flow[u][v]);
               pathArray.push(u+"+v);
            }
         }
      }
   }

   // If the color of the target node is black now,
   // it means that we reached it.
   return color[target] == BLACK;
}

function makeFake(obj){
```

```
            var st = R.set();
            var v     = R.circle(obj[0].attr('cx'), obj[0].attr('cy'), obj[0].attr('r')).attr({fill: obj[0].attr('fill'),opacity:
obj[0].attr('opacity'), stroke: obj[0].attr('stroke'),title:obj[0].attr('title')});
            var vtext = R.text  (obj[1].attr('x'),  obj[1].attr('y'),  obj[1].attr('text')).attr({fill: obj[1].attr('fill'),title:
obj[1].attr('title')});

            // make the new st object
            st.push(
               v,
               vtext.insertAfter(v)
            );
            st.toFront(); // place st in front
            obj.remove(); //remove the old obj which was movable. st is not movable

            // Bind the st event to a mouse click
            st.click(function(){
               makeFake(st);
            });
            // Make flow lines between st
            makeFlowlines(st);
         }

         function makeSink(){
            if (sinkflag == false){
               var sinktext   = R.text(220, 230,"T").attr({fill:"#FFF",title:"Sink"}).draggable.enable();       //Sink
red
               var sink = R.circle(220, 230, 20).attr({fill: "#CC0000", stroke: "none", opacity: .9,title:"T-
Terminal"}).draggable.enable();   // Green
               var st = R.set();
               st.push(
                  sink.attr({cursor:"pointer"}),
                  sinktext.insertAfter(sink).attr({cursor:"pointer"})
               );
               sinkflag = true;
               st.click(placeNodes(st));
               return;
            }
         }

         function makeSource(){
            if (sourceflag == false){
               var sourcetext = R.text(220,160,"S").attr({fill:"#FFF",title:"Source"}).draggable.enable();
//Source green
               var source = R.circle(220, 160, 20).attr({fill: "#66CC00", stroke: "none", opacity: .9,title:"S-
Source"}).draggable.enable()   // Green
               var st = R.set();
               st.push(
                  source.attr({cursor:"pointer"}),
                  sourcetext.insertAfter(source).attr({cursor:"pointer"})
```

```
            );
            sourceflag = true;
            st.click(placeNodes(st));
            return;
        }
    }

    function makeNode(){
        var nodename = 'v '+(i+1);
        var vtext    = R.text(220,300,nodename).attr({fill: "#FFF",title:"Node: "+nodename});
        var v        = R.circle(220, 300, 20).attr({fill: "#0066CC",opacity: .9, stroke: "none",title:"Node:
"+nodename});
        var st = R.set();

        st.push(
            v.draggable.enable().attr({cursor:"pointer"}),
            vtext.insertAfter(v).attr({cursor:"pointer"})
        );
        i++;
        st.click(placeNodes(st));
        return;
    }

    function placeNodes(obj){
        return function(){
            var xcoord = obj[0].attr('cx');
            var ycoord = obj[0].attr('cy');
            if (makeFlowflag == false){
                if( (xcoord >= 320 && xcoord <= 960) && (ycoord >= 125 && ycoord <= 605) ){
                }
                else{
                    alert("Please, drag to blue pallette");
                }
            }
            else if(makeFlowflag == true){

                obj.click=makeFake(obj);
            }
        }
    }

    function makeFlowlines(obj1){
        var factor;
        var str = obj1[1].attr('text');
        countNodes(str);
        switch(pointCount){
            case 0:
                Mx = obj1[0].attr('cx');
                My = obj1[0].attr('cy');
```

```
                fromNode = str; //for making the input file from Node
                pointCount++;
                break;

            case 1:
                Lx = obj1[0].attr('cx');
                Ly = obj1[0].attr('cy');
                var slope = (My - Ly)/(Mx - Lx);

                toNode = str; //for making the input file to Node

                if (slope < 0){
                    factor = -1.0;
                    if (Mx <= Lx && My >= Ly){
                        var midpointX = Math.round(Mx) - ((factor)*Math.round(Math.abs(((Mx - Lx)/2.0)))); //
Round to nearest integer
                        var midpointY = Math.round(My) + factor*Math.round(Math.abs(((My - Ly)/2.0)));
                    }
                    else{
                        var midpointX = Math.round(Lx) - ((factor)*Math.round(Math.abs(((Mx - Lx)/2.0))));
                        var midpointY = Math.round(Ly) + factor*Math.round(Math.abs(((My - Ly)/2.0)));
                    }
                }
                else{
                    factor = 1.0;
                    if (Mx <= Lx && My <= Ly){
                        var midpointX = Math.round(Mx) + ((factor)*Math.round(Math.abs(((Mx - Lx)/2.0))));
                        var midpointY = Math.round(My) + (factor)*Math.round(Math.abs(((My - Ly)/2.0)));
                    }
                    else{
                        var midpointX = Math.round(Lx) + ((factor)*Math.round(Math.abs(((Mx - Lx)/2.0))));
                        var midpointY = Math.round(Ly) + factor*Math.round(Math.abs(((My - Ly)/2.0)));
                    }
                }

                var midpointXY = midpointX+'|'+midpointY;
                var flowline    = R.path('M '+ Mx+' '+My+' L '+Lx+' '+Ly).attr({stroke:'#FFFFB2','stroke-
width':'2px',opacity:".3"}).toBack();
                var line_length = flowline.getTotalLength();
                var point       = flowline.getPointAtLength(line_length - 25);

                if((Mx > Lx) && (My > Ly)){
                    R.text(point.x,point.y,'V').attr({fill:'#FFFFCC','font-
size':'15px',opacity:".3",rotation:point.alpha + 270}).toBack();
                }
                else{ // OK
                    R.text(point.x,point.y,'V').attr({fill:'#FFFFCC','font-
size':'15px',opacity:".3",rotation:point.alpha - 270}).toBack();
                }
```

```
                //alert("Mx= "+ Mx +" My= " + My+"\nLx "+Lx+" Ly "+Ly+"\nSlope "+slope+'\nmidX
'+midpointX+' midY '+midpointY+'\nPath Length '+line_length+'\nPoint '+ point.x+' ' + point.y+' ' +
point.alpha);
                var flownum = prompt("Please enter flow","0");
                //count edges

                var capflowText = R.text(midpointX,midpointY,flownum).attr({fill:'#FFFFCC','font-
size':'16px'}).toFront();

                midpointMatrix(fromNode, toNode, numNodes, capflowText);

                capacity(fromNode, toNode, flownum); // Function capacity

                pointCount = 0;
                //alert('numNodes: '+numNodes+' numEdges: '+numEdges);
                numEdges++;
                break;
            }

            return true;
        }

        function MultiDimensionalArray(iRows,iCols){

        //http://www.developerfusion.com/code/2310/building-a-multidimensional-array-in-javascript/

            var i;
            var j;
            var a = new Array(iRows);
            for (i=0; i < iRows; i++){
              a[i] = new Array(iCols);
              for (j=0; j < iCols; j++){
                 a[i][j] = 0;
              }
            }
            return a;
        }

        function readInputFile(numNodes, numEdges, capacityFromTo){

            var inputFileStr = '';
            inputFileStr = capacityFromTo.toString();
            inputFileStr = inputFileStr.replace(/:/g,' ');
            capacityFromTo = inputFileStr.split(',');

            // Zero capacity matrix

            var capacity = MultiDimensionalArray(numNodes,numNodes);
```

```
    // Populate The capacity matrix with edges capacity
    var str = '';
    var strarray = [];
    for (var i=0; i< numEdges; i++) {
       str = capacityFromTo[i];
       strarray = str.split(' ');
       capacity[strarray[0]][strarray[1]] = strarray[2];
       str = '';
       strarray = [];
    }

    return capacity;
}


 function countNodes(str){

    if(nodeCheck == ''){
       nodeCheck = str;
       numNodes++;
    }
    else{
       if(nodeCheck.search(str) < 0){ //not found, add it
          nodeCheck = nodeCheck+':'+str;
          numNodes++;
       }
    }

    nodeSink = numNodes-1;
}

function makeInputFile(numNodes, numEdges, inputFile){
    //alert('Num Edges '+numEdges+'\nNum Nodes '+numNodes+'\nNodes '+nodeCheck);
    //Convert nodeCheck to numeric values
    //inputFile = [];
    numericNodes = nodeCheck.split(':');

    //change S->0; T->last number; v nn-> nn

    for (var i = 0; i < inputFile.length; i++){

       inputFile[i] = inputFile[i].replace(/S/g,'0');
       inputFile[i] = inputFile[i].replace(/T/g,numNodes-1);
       inputFile[i] = inputFile[i].replace(/v /g,'');
    }
    return inputFile;
}

function capacity(from, to, cap){
```

```
            capacityFromTo.push(from + ' ' + to + ' ' + cap +':');
            return;
          }
          function midpointMatrix(from, to, numNodes, midpointxy){

            //Create a hash table or associative array

            //alert ('***midpontMatrix\nfrom '+from+'\nto '+to+'\nmidpointxy '+midpointxy)

            //Translate from to

            from = from.replace(/S/g,'0');
            //from = from.replace(/T/g,numNodes-1);
            from = from.replace(/v /g,'');

            to = to.replace(/S/g,'0');
            //to = to.replace(/T/g,numNodes-1);
            to = to.replace(/v /g,'');

            //alert('From: '+from+' To: '+to+' numNodes: '+(numNodes-1)+' midpointxy: '+midpointxy);

            var fromto = from+to;

            midpointFromTo[fromto]=midpointxy;

            //alert(' midpointFromTo: '+ midpointFromTo[fromto]);

           return;
          }

        </script>
    </head>
    <body>
        <div id="holder">

        </div>
    </body>
</html>
```

Max Flow
CSS

```
/*
    Document   : 2maxflow
    Created on : Jul 23, 2011, 12:21:38 PM
    Author     : anthonyallen
    Description:
        Purpose of the stylesheet follows.
*/

/*
   TODO customize this sample style
   Syntax recommendation http://www.w3.org/TR/REC-CSS2/
*/

root {
    display: block;
}

body {
    background: #70705B;
    color: #fff;
    font: 300 100.1% "Helvetica Neue", Helvetica, "Arial Unicode MS", Arial, sans-serif;
}
#holder {
    height: 480px;
    width: 640px;

    left: 320px;
    top: 125px;

    position: absolute;
    border-right-color: #999999;
    border-left-color: #999999;
    border-bottom-color: #999999;
    border-top-color: #999999;
}
#holder1 {
    height: 480px;
    border-right-color: #cccc00;
    border-left-color: #cccc00;
    border-bottom-color: #cccc00;
    border-top-color: #336600;
    border-top-style: solid;
    border-top-width: 1px;
    border-bottom-width: 1px;
    border-left-width: 1px;
    border-bottom-style: solid;
    border-right-width: 1px;
    border-left-style: solid;
    border-right-style: solid;}
```