**Kmean Final**

Function read\_CSV\_data (Parameters: pointer filename, double pointer

row\_name, Pointer Punts, N, D, C, pointer Centroides)

{

Set pointer fp to filename

If (not fp)

{

Print (filename)

Return (one)

}

Print (filename)

Set i, j, n\_read, n\_total\_idx to zero

Set array line [size one thousand and twenty-four]

Call function (Arguments: lines, size of(line), fp)

While (Call function (Arguments: lines, size of(line), fp))

{

Set token to line

Set j, n\_each\_in\_line to zero

While (token not equal to null)

{

If (strcmp (token) is equal to zero)

Stop

If (j is equal to zero)

{

Call function strcpy (Arguments: row\_name

[index i], token)

Set j to sum of j and one

Set token to null

Continue

}

Set array Punts [index n\_total\_idx plus one] to token

Set n\_each\_in\_line to sum of n\_each\_in\_line and one

Set j to sum of j and one

If (n\_each\_in\_line is greater than equal to D)

Stop

Set token to null

}

If (n\_each\_in\_line not equal to D)

{

Print (n\_each\_in\_line)

Print (D)

Return (two)

}

Set i to sum of i and one

Set n\_read to sum of n\_read and one

If (n\_read greater than N)

Stop

}

Call function fclose (Arguments: fb)

Call function assert (n\_read is equal to N)

Set N to n\_read

Print (N, D, filename)

Loop (set i to zero; check i less than C; Increment i with one)

{

Set k to remainder or rand and N

Loop (set j to zero; check j less than D; Increment j with one)

{

Set Centroides to Punt

}

}

Loop (set i to zero; check i less than C; Increment i with one)

{

Loop (set j to zero; check j less than D; Increment j with one

Print (Centroides)

}

Return (zero)

Function PointsToCentroides (Parameter: pointer to Punts, N, D, C, pointer to

Centroides, pointer to new\_Center, pointer to Labels, pointer to Sep)

{

Declare variable j, k, min\_arg1, min\_arg2, dist1, dist2, dist3, dist4,

min\_dist1, min\_dist2

Set r to zero

Set min\_dist1, min\_dist2 to FLT\_MAX

Loop (set i to zero; check i less than C; Increment i with one)

{

Set array Sep [index i] to zero

Loop (set k to zero; check k less than D; Increment k with one)

Set array new\_Center to zero

}

Loop (set i to zero; check i less than N; Increment i with two)

{

Set j, dist1, dist2 to zero

If (C and one is equal to one)

{

Loop (set k to zero; check k less than D; Increment k with one)

{

Set punt1, punt2 to Punts

Set center to Centroides

Set temporal1 to difference of punt1 and center

Set temporal2 to difference of punt2 and center

Set dist1 to sum of dist1 and temporal1 and product

temporal1 and temporal1

Set dist2 to sum of dist1 and temporal1 and product

temporal1 and temporal1

}

Set min-arg1 to (dist1 less than min\_dist1) J inherited

to min\_arg1

Set min-dist1 to (dist1 less than min\_dist1) dist1 inherited

to min\_arg1

Set min-arg2 to (dist2 less than min\_dist2) J inherited

to min\_arg2

Set min-dist2 to (dist2 less than min\_dist2) dist2 inherited

to min\_arg2

Set j to one

Set dist1, dist2 to zero

}

Set dist3, dist4 to zero

Loop (empty; check j less than C; Increment j with two)

{

Loop (set k to zero; check k less than D; Increment k with one)

{

Set punt1, punt2 to Punts

Set center1, center2 to Centroides

Set temporal1 to difference of punt1 and center1

Set temporal2 to difference of punt1 and center2

Set temporal3 to difference of punt2 and center1

Set temporal3 to difference of punt2 and center2

Set dist1 to sum of dist1 and temporal1 and product

temporal1 and temporal1

Set dist2 to sum of dist2 and temporal2 and product

temporal2 and temporal2

Set dist3 to sum of dist3 and temporal3 and product

temporal3 and temporal3

Set dist4 to sum of dist4 and temporal4 and product

temporal4 and temporal4

}

Set min-arg1 to (dist1 less than min\_dist1) J inherited min\_arg1

Set min-dist1 to (dist1 less than min\_dist1) dist1 inherited

to min\_dist1

Set min-arg2 to (dist3 less than min\_dist2) J inherited min\_arg2

Set min-dist2 to (dist3 less than min\_dist2) dist3 inherited

to min\_dist2

Set min-arg1 to (dist2 less than min\_dist1) J+ one inherited

to min\_arg1

Set min-dist1 to (dist2 less than min\_dist1) dist1 inherited

to min\_dist1

Set min-arg2 to (dist4 less than min\_dist2) J+ one inherited

to min\_arg2

Set min-dist2 to (dist4 less than min\_dist2) dist4 inherited

to min\_dist2

}

Set array Labels [index i] to min\_arg1

Set array Sep [index min\_arg1] to sum of Sep [index min\_arg1] and one

Loop (set k to zero; check k less than D; Increment k with one)

Set array new\_Center [index min\_arg1 pointer D plus k] to new\_Center

[index min\_arg1 pointer d plus k] and Punts [index i pointer D plus

Set array Labels [index i plus one] to min\_arg2

Set array Sep [index min\_arg2] to sum of Sep [index min\_arg2] and one

Loop (set k to zero; check k less than D; Increment k with one)

Set array new\_Center [index min\_arg2 pointer D plus k] to new\_Center

[index min\_arg2 pointer D plus k] and Punts [index i plus

One pointer D plus K]

Loop (set j to zero; check j less than C; Increment j with one)

{

Set k to zero

Loop (set k to zero; check k less than D; Increment k with one)

{

Set value to division of new\_Center and Sep

If (value not equal to Centroides)

{

Set pointer pR to one

Set array Centroides to value

}

}

}

Return (r)

Start

Function main (Parameters: argc, pointer to argv)

{

Declare N, D, C

Set pointer file to malloc [size one thousand and twenty-four]

If (value of argc greater than five)

{

Call function (Argument: file)

Set C to four

Set N to one million

Set D to sixty-eight

Print (argv [index zero])

Print (C, N, D)

}

Else {

Call function (Argument: file, argv [index one])

Set C to call function atoi (Argument: argv [index two]

Set N to call function atoi (Argument: argv [index three])

Set D to call function atoi (Argument: argv [index four])

}

Set pointer Punts to malloc (size of pointer N and D)

Set pointer Labels to malloc (size of pointer N)

Set pointer Centroides to malloc (size of pointer C and D)

Set pointer new\_Center to malloc (size of pointer C and D)

Set pointer Sep to malloc (size of pointer C)

Set cont and final to zero

Set k to zero

Set double pointer row\_name to malloc (size of N)

Loop (set k to zero; check k less than N; Increment k with one)

Set array row\_name [index k] to malloc (size of ten)

Print (N, D, C)

If (call function read\_CSV\_data (Arguments: file, row\_name, Punts, N, D,

C, Centroides) is greater than to zero)

Do

{

Set final to Call function PointsToCentroides (Arguments: Punts, N, D, C,

Centroides, new\_Center, Labels, Sep)

Set cont to sum of cont and one

}

While (final or cont less than to one)

{

Print (cont)

Print (final)

Set i and j to zero

Loop (set i to zero; check i less than C; Increment i with one)

{

Loop (set j to zero; check j less than D; Increment j with one)

{

Print (Centroides)

}

}

}

}

Set array line [size one thousand and twenty-four]

Set pointer fp to file

Call function fgets (Arguments: line, size of (line), fp)

Call function fclose (Arguments: fp)

Set pointer result\_file to result\_csv

If (not result\_file)

Return (one)

fprint (result\_file, C)

Loop (set k to zero; check k less than to C; Increment k with one)

fprint (result\_file, k plus one, Sep [index k])

Loop (set i to zero; check i less than to C; Increment i with one)

{

fprint (result\_file, i plus one)

fprint (result\_file, line)

Loop (set j to zero; check j less than to N; Increment j with one)

{

If (Labels [index i] is equal to i)

fprint (result\_file, row\_name)

Loop (set k to zero; check k less than to D; Increment k with one)

{

fprint (result\_file, Punts)

}

}

}

Call function fclose (Arguments: file)

Call function free (Argument: Punts);

Call function free (Argument: Centroides);

Call function free (Argument: new\_Center);

Call function free (Argument: Labels);

Call function free (Argument: Sep);

Call function free (Argument: row\_name);

Out