

COMS 4735 Report – By Clarence Jiang (yj2737)

This project intends to build a system that extracts similar images. The 3 main inputs are a collection of 40 images as input, a crowd.txt file that stores the “ground truth” of image similarity results, and a “mypreferences.txt” which represents my attitude towards which pairs of images are similar.

Step 1: Color distance

In the first stage of this content-based image retrieval system, I used “color” as a primary factor to determine image similarity. My first task is to convert the pixels into a 3d color histogram and decide how rich or sparse my histogram bins need. The main package I used is the PIL python package. I read images through the “Image.open” function and converted the image objects to a list of (r,g,b) pixels by the “image.getdata” function. I did not keep the image form as a 2d numpy array, because in this stage, the color comparison will compare the distance between each corresponding histogram bin between 2 images, so it is not essential to maintain a 2d form. A list format also keeps the order.

Then I set up my histogram by first following the strategy in the instruction. Since human eyes are more difficult to distinguish blueish color, I decided to take 1 leftmost bit from the blue value, and 2 leftmost bits from the red and green values. This will result in a total of $2^1 * 2^2 * 2^2 = 32$ bins. I converted each integer number of RGB value to its 8-bit binary form and take its 1 or 2 leftmost bits through the “.format” function and normal index slicing. Then, I created the histogram by counting values in each bin correspondingly. To evaluate if this choice gives a good performance, I calculated the histogram distance between each pair of images through numpy functions like np.abs and np.sum. Then I established a 40 * 40 numpy array. For each row, I applied np.argpartition and np.argsort to find the 3 images with the minimum distances with the query image. Also, the indices of the images are sorted by their distances with the query image as well. That is to say, t1 (the variable name used in the instruction) will always be the closest image to the query image. Eventually, I added up the score of the crowd(q, t1), crowd(q, t2), crowd(q, t2) for every image as my score by using “np.loadtxt” to load the txt file as a 2d numpy array. My initial design gave a final score of 9504.

To explore if there is a better design, I created a function called shrinking_explore() to test through the permutation of every r, g, and b bit from 1 to 4. In other words, I checked the score of every combination from having 1 leftmost bit of r, g, and b values to having 4 leftmost bits of RGB. Since the bit used by the 3 lenses could be different, a lot of combinations are explored. The best setting is having 1 bit for blue, 2 bits for red, and 3 bits for green. This setting gave a score of **11321**. I think this result is also intuitive, as humans are not good at seeing blue, so for the blue lens, we could use fewer bits. People are good at distinguishing green, so there could be more bits of green. Utilizing 3 digits seems like a threshold, and I found that the value does not increase above that.

Finally, to output the result, I created a string variable to store some HTML codes, where each image indices and its corresponding crowd score are incorporated into the HTML codes as

values through the “img src” tag. In addition, the sum of the crowd scores of each row is appended at the end of each row; the total system score is noted at the very beginning of the HTML file (shown at the end). With the aid of this HTML file, I could showcase my result in a PDF format. Since the parameters of my “output-HTML” function are the indices of the 3 close images for each query image, the scores of the 3 close images for each query image, the final system score, my happiness score, and a step indicator. This part of the code could be repeatedly used for the rest.

Besides, the size of overlapped set between the system and my personal results are **50**, indicating that the similar images discovered by the system overlap with about half of my personal results. I think this result is reasonable since it is hard to track metrics utilized by other students, so it is really hard to anticipate a value. I have no idea how different my thinking process is compared to other students.

Step 2: Texture

For the second stage, I first created 40 grayscale images. I originally used the function of “image.convert(“L”)” in the PIL image library, yet I then found this function used an incorrect formula ($L = R * 299/1000 + G * 587/1000 + B * 114/1000$). Thus, I decided to iterate the image pixels myself and calculated $I = (R+G+B)/3$. Then I also pursued a manual process of converting a gray image to a Laplacian image. Originally, I intended to use the ImageFitter module within the PIL library; this class has a built-in Kernel class, but it replaced the pixel values with incorrect values, so I switched to a manual “pixel update” process by using a double for-loop structure and “image.put_pixel” function. To deal with the boundary values, I also padded the images with zeros around them. To deal with negative values, I used their absolute values. Even though it seems tedious to not use a built-in function, my method does not make the program run extremely slow. Plus, I am confident that my own calculation exactly matches requirements.

With these Laplacian images, I followed a similar procedure in step 1 to explore how to reduce the bins involved in the histograms in this step. I iteratively took 1 leftmost bit to 11 bits from the left. I discovered that when less than 4 bits are taken, the system score would be really low, and there is an increasing trend that as more bits are taken, the score will be higher. Indeed, the max system score is about 6771 when 9 bits are considered, but I think that’s too many bins for histograms. I decided to follow the same mind in step 1 to find a good balance point that has relatively few bins but still maintains a relatively good score. I think this is a good strategy because maintaining so many bins for a Laplacian image, which absolutely does not involve so many colors, is unnecessary. Thus, I decided to take the 6 leftmost bits, and this gave me a final score of 6731, which is the second-highest score.

Starting from this step, I added a new parameter to the function that I used to calculate histogram distance (the function that sets up the 40*40 distance matrix). This new parameter, called step, would indicate which step I am working on, so I could adjust the methods to calculate histogram distance. I chose to use the same L1 distance norm formula in step 1 since the max difference would still be $2 * \text{height} * \text{width}$. As this distance metric could already be

used to narrow down the distance to the range [0, 1], I think there is no need to modify it. Relative distance could not be changed by the normalization technique, so using different normalized distances is unnecessary; it does not change the 3 closest images. Then I just applied the same codes in step 1 to find the 3 closest images, extract their crowd scores, calculate the final system score, count the number of overlaps in my own preferences, and output the HTML file. This pipeline from finding the 3 closest images to output the final HTML is defined as a function called “common_evaluation”. I will not repeat discussing this part in the following steps because it’s basically the same.

The final score of my texture system is **6731**, and the output results only overlap with **28** of my preferences. As for the system score, I think it is quite reasonable to be lower than using the color to distinguish, as texture is a more abstract concept than color. In common sense, humans are familiar with making decisions through colors. On the other hand, I did not consider texture too much when choosing images, so I think a low overlap with my personal preference is also acceptable.

Step 3: Shape

In the third step, I first need to convert the intensity images into binary images based on a threshold value. In other words, any pixel values greater than this threshold value should be assigned as 255, and otherwise 0. To convert a gray intensity image to a binary image, I just iterated through the pixel value to see if it is greater than a specific threshold value, and then update the pixel correspondingly to either 255 or 0 using “image.putpixel”.

To explore what is a good threshold, I first created a function called “explore_black_boundary”. In this function, I checked the pixel values of the middle row of the 18th image, since the middle row of this image has a clear jump from white to black, so checking that row is a good starting point. I discovered that generally, a pixel value around 80 might be ideal. However, some pixels with a value of 40 also appear to be not black. Thus, to explore the best threshold, I iterated through all integer threshold values between 40 and 140 and checked the optimized parameter that maximizes the final system score. I did this in the function of “explore_best_black_threshold”. This iteration process gave the optimized black threshold value of 78, which gave a system score of **6563**. It overlapped with **29** of my preferences. Again I argue that my whole process in this step is quite accurate. I first used an example image to develop an idea of what is potential range I should search. Then I did an iterative search to explore all integer threshold values within that range. Searching only integer results is sufficient, as having a difference in the decimal part does not change the system score much. Similarly, I output all the required information in HTML form.

For this stage, I think 6563 is an expected result. Shape is definitely a more “tangible” comparison criterion compared to texture for most people, so a system based on shape should have a greater score than using texture. However, it’s also not that straightforward since a lot of given images contain a lot of objects stacking together, such as tons of apples stacking together.

This will affect the system a lot. Theoretically, if it is just one apple VS one orange, I would say using shape will give a better result.

Eventually, for the comparison with my own results, it is approximately the same as step 2. I think color is definitely a more important criterion when choosing, so according to the value difference in step 1 and step 3, this overlap is likely to be correct. Again, it is hard to give feedback on the happiness value, as there is no guarantee on how other students think.

Step 4: Symmetry

In this step, I also need to convert intensity to binary images. I started with 72, the black threshold value I obtained from Step 3. For each binary image, I will calculate symmetry values by counting the number of different pixels from the left half to the right half of the image. My algorithm is to iterate through the left half of the image, and for each column, find its mirror column. For instance, column 0 will be compared with column 88. I converted each image into numpy array format, selected a pair of columns, and applied the function of `"np.count_nonzero(column1 != column2)"`. Only the pixels that are different will be evaluated to True, which is considered a nonzero number and will contribute to the `"count_nonzero"`. Then for each image pair, I calculated their distance by taking the difference between their symmetry values. I followed the same `"common_evaluation"` process to get all the desired outputs.

To explore if the best threshold value of shape also works for symmetry, I also performed an iterative search process. My search range is again from 40 to 140. I stick with this range because I believe out of this range, it is unreasonable to call a pixel just greater than 39 to be "white" or a pixel just less than 141 to be "black". This range matches more with common sense that black and white should be closer to 0 and 255 respectively. An iterative search gave me the best system score to be **4439**, a happiness value of **19**, and the best black threshold is 99.

This result matches my expectation since symmetry should definitely have the lowest score. If two completely random images are both asymmetric, they will be tagged as similar. That does not make too much sense. Likewise, in some sense, human faces and apples could be both relatively symmetrical. Grouping them together neither makes sense. Besides, the happiness value is also the lowest for the same reason.

Step 5: Overall distance

For this step, I need to figure out a weight vector. Then I multiply the weight with each histogram distance matrix returned by the previous steps. I will use the `dump` function in the `"pickle"` module to store the 4 histogram distance matrices to save running time. Each distance value, regardless of the steps, falls in the range of 0 to 1. If the 4 weights sum to 1, the final distance value in the combined matrix will also be in the range of [0,1]. With this final distance matrix, I will evaluate through the same `"common_evaluation"` process to find the system score.

The problem is to figure out the optimized weight vector. I started by setting the texture and the symmetry weights to 0.05 since I think these 2 aspects are not important. This narrows down the problem to distributing the rest 0.9 percent to color and shape. I thought the color

weight color should be as high as 0.8, so I tested different weight vectors that have color to be around 0.8. I found the vector of (0.83, 0.05, 0.07, 0.05) could give a score of 11633. The vector from left to right represents color, texture, shape, and symmetry.

Then I tried to lower the color percentage and increase the shape percentage but always keep their sum to 0.9. I discovered that when the color percentage is above 0.38, the final score is not affected much by the color weight. If there is no need to have a high color weight, it means that some other aspects need a higher score. When fixing color and shape to 0.38 and 0.52 and changing the other 2 aspects, I found there is clear evidence that the lower the texture, the better the system score, I set the texture to be fixed at 0.03. But this is the result assuming shape + color = 0.9, so I tried to make it 0.8, and the new score is 11661 (0.46, 0.03, 0.34, 0.17), which is slightly better than the previous score (11633). Then I further tried to lower the combined weights of color and shape to 0.7, the new system score is 11682. This whole process keeps going down to 0.6 for color+shape.

However, all these processes show my thinking process of how to optimize the weight vector, but it does not test through all possibilities, so I wrote a 3-layer-nested-for-loop structure. In each for-loop, I will iterate all values of one aspect. For instance, the outermost for-loop iterates through texture weight values in the range of [0, 1]. After each iteration, the weight value will add 0.01. Then for the next for-loop, it will iterate the color weight in the range of [0, 1-texture_weight]. Then for the 3rd for-loop, it will iterate the shape weight in the range of [0, 1-texture_weight-color_weight]. Then the value of symmetry weight = 1-texture_weight-color_weight-shape_weight.

This algorithm makes sure the 4 weight vectors sum up to 1 and also fully explores all the possibilities with a unit length of 0.01. For each possibility, I will use the same “common_evaluation” pipeline to calculate its final system score. Eventually, the optimized weight vector is **[0.44, 0.09, 0.23, 0.24]** and the optimized score is **12852**. This weight vector overall matches my expectation. My previous exploration shows that the color weight does not need to be high, but I did not expect such a high value of symmetry weight. However, it is possible that I am an outlier. Also, this integrated system overlaps **52** of my preference, which is the highest.

Step 6: Crowd versus you

To do this step, I first need to construct the sparse matrix. I created an empty 40 * 40 numpy array and read through my_preference.txt. For each image, I will use the image index to determine the position and the order it gets extracted as the value (3, 2, or 1). Then I saved this constructed matrix as a file called “step6.txt” using np.savetxt() for convenience.

To find an optimized weight vector. I started with the optimized weight vector in Step 5 since I want to have an idea of how different my thoughts are compared to other students, and I got a score of 116. In all of the previous steps, the overlap with my preference is also around 50%, so I supposed that this is a good result. To further confirm, I also tested with some other weight vectors, and it turns out that 116 is a good result.

















































































In addition, I went through the same iterative search process in Step 5 to check all possibilities, and the optimized weight vector is **[0.31, 0.32, 0.33, 0.04]**. The optimized happiness score is **123**. I found this linear weight matches more with my preference because of the low symmetry score. As I have explained, I do not believe symmetry to play a huge role in this image retrieval system, and this idea is reflected by a low symmetry weight. There is not a drastic increase in happiness values, and I think this is because the 4 models from Step 1 to Step 4 are trained based on the crowd's intelligence, so it might not guarantee a significantly high number when evaluating my own preference file.


























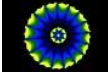

















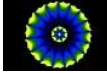




































Summary:

Overall I think my eventual image retrieval system performs a great job. While observing the HTML output, I think a lot of the chosen close images are indeed similar to the query images. There are some query images with low crowd scores, but those images serve more as distraction images that are not fruit or vegetables. As a result, I would still think my system performs well. All the HTML outputs of each step are shown below on the following pages. The orders are from Step1 to Step6, and each step takes 2 pages. The system-level information will be shown in the upper left corner of each step's opening page.

The system score is 11321.0




























The happiness value is 50














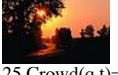






























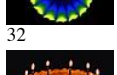



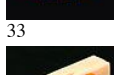























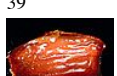



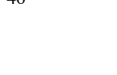
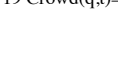
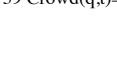
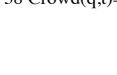
				the rows score is 540
01	10 Crowd(q,t)=88	08 Crowd(q,t)=261	03 Crowd(q,t)=191	
				the rows score is 81
02	39 Crowd(q,t)=30	09 Crowd(q,t)=21	21 Crowd(q,t)=30	
				the rows score is 462
03	04 Crowd(q,t)=272	08 Crowd(q,t)=189	36 Crowd(q,t)=1	
				the rows score is 425
04	03 Crowd(q,t)=295	08 Crowd(q,t)=121	36 Crowd(q,t)=9	
				the rows score is 376
05	06 Crowd(q,t)=309	23 Crowd(q,t)=0	11 Crowd(q,t)=67	
				the rows score is 473
06	11 Crowd(q,t)=72	05 Crowd(q,t)=259	07 Crowd(q,t)=142	
				the rows score is 284
07	40 Crowd(q,t)=48	09 Crowd(q,t)=178	11 Crowd(q,t)=58	
				the rows score is 595
08	03 Crowd(q,t)=233	01 Crowd(q,t)=260	04 Crowd(q,t)=102	
				the rows score is 312
09	07 Crowd(q,t)=239	11 Crowd(q,t)=61	40 Crowd(q,t)=12	
				the rows score is 441
10	01 Crowd(q,t)=136	16 Crowd(q,t)=272	03 Crowd(q,t)=33	
				the rows score is 451
11	09 Crowd(q,t)=98	07 Crowd(q,t)=122	06 Crowd(q,t)=231	
				the rows score is 331
12	02 Crowd(q,t)=30	09 Crowd(q,t)=75	14 Crowd(q,t)=226	
				the rows score is 258
13	39 Crowd(q,t)=10	10 Crowd(q,t)=47	14 Crowd(q,t)=201	
				the rows score is 190
14	39 Crowd(q,t)=30	02 Crowd(q,t)=13	13 Crowd(q,t)=147	
				the rows score is 72
15	40 Crowd(q,t)=0	07 Crowd(q,t)=72	20 Crowd(q,t)=0	
				the rows score is 303
16	10 Crowd(q,t)=240	01 Crowd(q,t)=39	03 Crowd(q,t)=24	
				the rows score is 342
17	25 Crowd(q,t)=8	18 Crowd(q,t)=268	35 Crowd(q,t)=66	
				the rows score is 353
18	17 Crowd(q,t)=236	25 Crowd(q,t)=0	35 Crowd(q,t)=117	
				the rows score is 371
19	38 Crowd(q,t)=76	36 Crowd(q,t)=60	24 Crowd(q,t)=235	
				the rows score is 274
20	35 Crowd(q,t)=108	40 Crowd(q,t)=156	18 Crowd(q,t)=10	

				the rows score is 6
21	34 Crowd(q,t)=3	28 Crowd(q,t)=3	25 Crowd(q,t)=0	
				the rows score is 108
22	34 Crowd(q,t)=20	36 Crowd(q,t)=84	38 Crowd(q,t)=4	
				the rows score is 77
23	35 Crowd(q,t)=53	40 Crowd(q,t)=16	28 Crowd(q,t)=8	
				the rows score is 299
24	36 Crowd(q,t)=31	38 Crowd(q,t)=46	19 Crowd(q,t)=222	
				the rows score is 23
25	17 Crowd(q,t)=16	18 Crowd(q,t)=7	21 Crowd(q,t)=0	
				the rows score is 16
26	07 Crowd(q,t)=0	40 Crowd(q,t)=16	11 Crowd(q,t)=0	
				the rows score is 267
27	32 Crowd(q,t)=207	24 Crowd(q,t)=32	30 Crowd(q,t)=28	
				the rows score is 380
28	33 Crowd(q,t)=149	21 Crowd(q,t)=3	25 Crowd(q,t)=228	
				the rows score is 338
29	30 Crowd(q,t)=311	36 Crowd(q,t)=18	38 Crowd(q,t)=9	
				the rows score is 344
30	29 Crowd(q,t)=305	36 Crowd(q,t)=29	38 Crowd(q,t)=10	
				the rows score is 322
31	28 Crowd(q,t)=16	33 Crowd(q,t)=25	32 Crowd(q,t)=281	
				the rows score is 191
32	27 Crowd(q,t)=110	24 Crowd(q,t)=71	28 Crowd(q,t)=10	
				the rows score is 353
33	28 Crowd(q,t)=149	40 Crowd(q,t)=86	25 Crowd(q,t)=118	
				the rows score is 110
34	21 Crowd(q,t)=95	28 Crowd(q,t)=14	38 Crowd(q,t)=1	
				the rows score is 331
35	17 Crowd(q,t)=97	18 Crowd(q,t)=207	20 Crowd(q,t)=27	
				the rows score is 170
36	38 Crowd(q,t)=148	24 Crowd(q,t)=21	03 Crowd(q,t)=1	
				the rows score is 345
37	36 Crowd(q,t)=262	24 Crowd(q,t)=78	30 Crowd(q,t)=5	
				the rows score is 414
38	36 Crowd(q,t)=201	24 Crowd(q,t)=84	19 Crowd(q,t)=129	
				the rows score is 6
39	02 Crowd(q,t)=6	09 Crowd(q,t)=0	08 Crowd(q,t)=0	
				the rows score is 287
40	07 Crowd(q,t)=80	23 Crowd(q,t)=24	35 Crowd(q,t)=183	

The system score is 6731.0


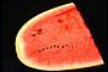















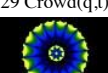



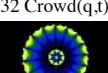



















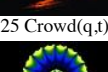






































The happiness value is 28



























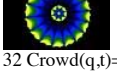



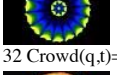










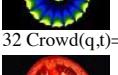


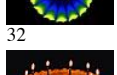



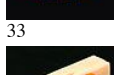




















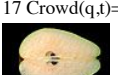


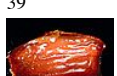



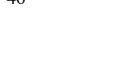
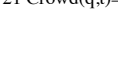
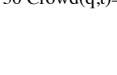
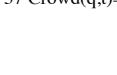
				the rows score is 540
01	08 Crowd(q,t)=261	03 Crowd(q,t)=191	10 Crowd(q,t)=88	
				the rows score is 140
02	06 Crowd(q,t)=25	14 Crowd(q,t)=46	12 Crowd(q,t)=69	
				the rows score is 338
03	08 Crowd(q,t)=189	01 Crowd(q,t)=115	10 Crowd(q,t)=34	
				the rows score is 4
04	07 Crowd(q,t)=4	31 Crowd(q,t)=0	09 Crowd(q,t)=0	
				the rows score is 208
05	15 Crowd(q,t)=72	31 Crowd(q,t)=0	07 Crowd(q,t)=136	
				the rows score is 2
06	02 Crowd(q,t)=1	14 Crowd(q,t)=0	12 Crowd(q,t)=1	
				the rows score is 178
07	13 Crowd(q,t)=0	09 Crowd(q,t)=178	16 Crowd(q,t)=0	
				the rows score is 518
08	01 Crowd(q,t)=260	03 Crowd(q,t)=233	10 Crowd(q,t)=25	
				the rows score is 249
09	16 Crowd(q,t)=5	13 Crowd(q,t)=5	07 Crowd(q,t)=239	
				the rows score is 195
10	08 Crowd(q,t)=26	01 Crowd(q,t)=136	03 Crowd(q,t)=33	
				the rows score is 105
11	16 Crowd(q,t)=7	09 Crowd(q,t)=98	13 Crowd(q,t)=0	
				the rows score is 242
12	14 Crowd(q,t)=226	15 Crowd(q,t)=12	05 Crowd(q,t)=4	
				the rows score is 60
13	07 Crowd(q,t)=6	16 Crowd(q,t)=7	09 Crowd(q,t)=47	
				the rows score is 305
14	12 Crowd(q,t)=210	15 Crowd(q,t)=92	05 Crowd(q,t)=3	
				the rows score is 122
15	05 Crowd(q,t)=93	31 Crowd(q,t)=0	14 Crowd(q,t)=29	
				the rows score is 31
16	09 Crowd(q,t)=20	13 Crowd(q,t)=4	07 Crowd(q,t)=7	
				the rows score is 12
17	25 Crowd(q,t)=8	24 Crowd(q,t)=3	28 Crowd(q,t)=1	
				the rows score is 92
18	20 Crowd(q,t)=7	36 Crowd(q,t)=0	23 Crowd(q,t)=85	
				the rows score is 311
19	40 Crowd(q,t)=0	38 Crowd(q,t)=76	24 Crowd(q,t)=235	
				the rows score is 160
20	18 Crowd(q,t)=10	36 Crowd(q,t)=8	23 Crowd(q,t)=142	

				the rows score is 192
21	29 Crowd(q,t)=27	22 Crowd(q,t)=157	33 Crowd(q,t)=8	
				the rows score is 336
22	29 Crowd(q,t)=113	34 Crowd(q,t)=20	21 Crowd(q,t)=203	
				the rows score is 150
23	38 Crowd(q,t)=0	18 Crowd(q,t)=150	25 Crowd(q,t)=0	
				the rows score is 50
24	25 Crowd(q,t)=0	28 Crowd(q,t)=4	38 Crowd(q,t)=46	
				the rows score is 178
25	28 Crowd(q,t)=178	38 Crowd(q,t)=0	24 Crowd(q,t)=0	
				the rows score is 33
26	35 Crowd(q,t)=32	34 Crowd(q,t)=1	30 Crowd(q,t)=0	
				the rows score is 17
27	04 Crowd(q,t)=5	09 Crowd(q,t)=0	03 Crowd(q,t)=12	
				the rows score is 230
28	25 Crowd(q,t)=228	38 Crowd(q,t)=0	24 Crowd(q,t)=2	
				the rows score is 499
29	21 Crowd(q,t)=63	22 Crowd(q,t)=125	30 Crowd(q,t)=311	
				the rows score is 374
30	29 Crowd(q,t)=305	21 Crowd(q,t)=59	34 Crowd(q,t)=10	
				the rows score is 0
31	05 Crowd(q,t)=0	07 Crowd(q,t)=0	15 Crowd(q,t)=0	
				the rows score is 27
32	33 Crowd(q,t)=17	21 Crowd(q,t)=10	34 Crowd(q,t)=0	
				the rows score is 39
33	34 Crowd(q,t)=10	21 Crowd(q,t)=18	32 Crowd(q,t)=11	
				the rows score is 319
34	22 Crowd(q,t)=207	33 Crowd(q,t)=17	21 Crowd(q,t)=95	
				the rows score is 4
35	30 Crowd(q,t)=0	29 Crowd(q,t)=0	22 Crowd(q,t)=4	
				the rows score is 198
36	18 Crowd(q,t)=3	20 Crowd(q,t)=4	37 Crowd(q,t)=191	
				the rows score is 14
37	30 Crowd(q,t)=5	29 Crowd(q,t)=9	21 Crowd(q,t)=0	
				the rows score is 129
38	25 Crowd(q,t)=0	28 Crowd(q,t)=0	19 Crowd(q,t)=129	
				the rows score is 111
39	40 Crowd(q,t)=9	19 Crowd(q,t)=3	38 Crowd(q,t)=99	
				the rows score is 19
40	19 Crowd(q,t)=0	39 Crowd(q,t)=13	38 Crowd(q,t)=6	

The system score is 6563.0






















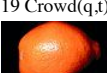





















































The happiness value is 29







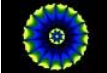
































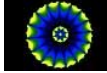








































				the rows score is 0
01	20 Crowd(q,t)=0	29 Crowd(q,t)=0	38 Crowd(q,t)=0	
				the rows score is 47
02	26 Crowd(q,t)=36	36 Crowd(q,t)=3	13 Crowd(q,t)=8	
				the rows score is 0
03	35 Crowd(q,t)=0	18 Crowd(q,t)=0	39 Crowd(q,t)=0	
				the rows score is 0
04	29 Crowd(q,t)=0	22 Crowd(q,t)=0	39 Crowd(q,t)=0	
				the rows score is 309
05	32 Crowd(q,t)=0	27 Crowd(q,t)=0	06 Crowd(q,t)=309	
				the rows score is 0
06	32 Crowd(q,t)=0	33 Crowd(q,t)=0	28 Crowd(q,t)=0	
				the rows score is 211
07	27 Crowd(q,t)=0	33 Crowd(q,t)=0	06 Crowd(q,t)=211	
				the rows score is 260
08	20 Crowd(q,t)=0	29 Crowd(q,t)=0	01 Crowd(q,t)=260	
				the rows score is 179
09	28 Crowd(q,t)=3	06 Crowd(q,t)=176	24 Crowd(q,t)=0	
				the rows score is 136
10	25 Crowd(q,t)=0	26 Crowd(q,t)=0	01 Crowd(q,t)=136	
				the rows score is 231
11	32 Crowd(q,t)=0	06 Crowd(q,t)=231	33 Crowd(q,t)=0	
				the rows score is 401
12	14 Crowd(q,t)=226	29 Crowd(q,t)=0	13 Crowd(q,t)=175	
				the rows score is 230
13	15 Crowd(q,t)=10	14 Crowd(q,t)=201	02 Crowd(q,t)=19	
				the rows score is 449
14	15 Crowd(q,t)=92	13 Crowd(q,t)=147	12 Crowd(q,t)=210	
				the rows score is 98
15	13 Crowd(q,t)=1	14 Crowd(q,t)=29	16 Crowd(q,t)=68	
				the rows score is 136
16	15 Crowd(q,t)=129	13 Crowd(q,t)=4	14 Crowd(q,t)=3	
				the rows score is 273
17	38 Crowd(q,t)=3	18 Crowd(q,t)=268	22 Crowd(q,t)=2	
				the rows score is 237
18	17 Crowd(q,t)=236	22 Crowd(q,t)=1	29 Crowd(q,t)=0	
				the rows score is 175
19	38 Crowd(q,t)=76	18 Crowd(q,t)=65	17 Crowd(q,t)=34	
				the rows score is 11
20	29 Crowd(q,t)=1	18 Crowd(q,t)=10	38 Crowd(q,t)=0	

				the rows score is 104
21	30 Crowd(q,t)=5	17 Crowd(q,t)=99	40 Crowd(q,t)=0	
				the rows score is 140
22	29 Crowd(q,t)=113	18 Crowd(q,t)=18	17 Crowd(q,t)=9	
				the rows score is 256
23	17 Crowd(q,t)=106	18 Crowd(q,t)=150	38 Crowd(q,t)=0	
				the rows score is 171
24	23 Crowd(q,t)=165	35 Crowd(q,t)=0	21 Crowd(q,t)=6	
				the rows score is 297
25	37 Crowd(q,t)=0	26 Crowd(q,t)=292	34 Crowd(q,t)=5	
				the rows score is 14
26	02 Crowd(q,t)=14	36 Crowd(q,t)=0	37 Crowd(q,t)=0	
				the rows score is 219
27	33 Crowd(q,t)=12	32 Crowd(q,t)=207	06 Crowd(q,t)=0	
				the rows score is 152
28	33 Crowd(q,t)=149	32 Crowd(q,t)=3	06 Crowd(q,t)=0	
				the rows score is 134
29	22 Crowd(q,t)=125	18 Crowd(q,t)=0	38 Crowd(q,t)=9	
				the rows score is 172
30	21 Crowd(q,t)=59	22 Crowd(q,t)=113	37 Crowd(q,t)=0	
				the rows score is 298
31	32 Crowd(q,t)=281	06 Crowd(q,t)=1	28 Crowd(q,t)=16	
				the rows score is 27
32	23 Crowd(q,t)=17	28 Crowd(q,t)=10	06 Crowd(q,t)=0	
				the rows score is 104
33	40 Crowd(q,t)=86	37 Crowd(q,t)=0	21 Crowd(q,t)=18	
				the rows score is 52
34	40 Crowd(q,t)=18	30 Crowd(q,t)=21	37 Crowd(q,t)=13	
				the rows score is 282
35	21 Crowd(q,t)=24	18 Crowd(q,t)=207	39 Crowd(q,t)=51	
				the rows score is 254
36	37 Crowd(q,t)=191	35 Crowd(q,t)=56	30 Crowd(q,t)=7	
				the rows score is 267
37	30 Crowd(q,t)=5	36 Crowd(q,t)=262	21 Crowd(q,t)=0	
				the rows score is 24
38	17 Crowd(q,t)=0	18 Crowd(q,t)=2	29 Crowd(q,t)=22	
				the rows score is 181
39	22 Crowd(q,t)=101	18 Crowd(q,t)=19	29 Crowd(q,t)=61	
				the rows score is 32
40	21 Crowd(q,t)=11	30 Crowd(q,t)=0	37 Crowd(q,t)=21	

The system score is 4439.0












































































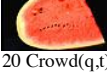




The happiness value is 19


























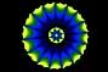

















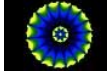




































				the rows score is 0
01	12 Crowd(q,t)=0	07 Crowd(q,t)=0	09 Crowd(q,t)=0	
				the rows score is 2
02	23 Crowd(q,t)=0	40 Crowd(q,t)=2	27 Crowd(q,t)=0	
				the rows score is 189
03	08 Crowd(q,t)=189	13 Crowd(q,t)=0	15 Crowd(q,t)=0	
				the rows score is 81
04	16 Crowd(q,t)=26	10 Crowd(q,t)=55	25 Crowd(q,t)=0	
				the rows score is 0
05	34 Crowd(q,t)=0	19 Crowd(q,t)=0	31 Crowd(q,t)=0	
				the rows score is 3
06	36 Crowd(q,t)=0	35 Crowd(q,t)=2	40 Crowd(q,t)=1	
				the rows score is 179
07	01 Crowd(q,t)=0	09 Crowd(q,t)=178	12 Crowd(q,t)=1	
				the rows score is 233
08	03 Crowd(q,t)=233	13 Crowd(q,t)=0	15 Crowd(q,t)=0	
				the rows score is 288
09	15 Crowd(q,t)=49	07 Crowd(q,t)=239	01 Crowd(q,t)=0	
				the rows score is 338
10	16 Crowd(q,t)=272	04 Crowd(q,t)=66	25 Crowd(q,t)=0	
				the rows score is 0
11	28 Crowd(q,t)=0	27 Crowd(q,t)=0	24 Crowd(q,t)=0	
				the rows score is 249
12	01 Crowd(q,t)=0	07 Crowd(q,t)=23	14 Crowd(q,t)=226	
				the rows score is 0
13	03 Crowd(q,t)=0	08 Crowd(q,t)=0	19 Crowd(q,t)=0	
				the rows score is 210
14	25 Crowd(q,t)=0	12 Crowd(q,t)=210	01 Crowd(q,t)=0	
				the rows score is 156
15	09 Crowd(q,t)=84	07 Crowd(q,t)=72	01 Crowd(q,t)=0	
				the rows score is 283
16	10 Crowd(q,t)=240	04 Crowd(q,t)=43	25 Crowd(q,t)=0	
				the rows score is 378
17	26 Crowd(q,t)=6	18 Crowd(q,t)=268	21 Crowd(q,t)=104	
				the rows score is 348
18	21 Crowd(q,t)=109	17 Crowd(q,t)=236	26 Crowd(q,t)=3	
				the rows score is 2
19	13 Crowd(q,t)=0	05 Crowd(q,t)=0	03 Crowd(q,t)=2	
				the rows score is 19
20	38 Crowd(q,t)=0	36 Crowd(q,t)=8	33 Crowd(q,t)=11	

				the rows score is 277
21	18 Crowd(q,t)=151	17 Crowd(q,t)=99	29 Crowd(q,t)=27	
				the rows score is 70
22	39 Crowd(q,t)=67	32 Crowd(q,t)=1	37 Crowd(q,t)=2	
				the rows score is 69
23	02 Crowd(q,t)=0	40 Crowd(q,t)=16	35 Crowd(q,t)=53	
				the rows score is 4
24	24 Crowd(q,t)=0	28 Crowd(q,t)=4	11 Crowd(q,t)=0	
				the rows score is 1
25	14 Crowd(q,t)=0	12 Crowd(q,t)=1	01 Crowd(q,t)=0	
				the rows score is 34
26	17 Crowd(q,t)=15	18 Crowd(q,t)=19	21 Crowd(q,t)=0	
				the rows score is 62
27	24 Crowd(q,t)=32	28 Crowd(q,t)=30	11 Crowd(q,t)=0	
				the rows score is 15
28	11 Crowd(q,t)=1	27 Crowd(q,t)=12	24 Crowd(q,t)=2	
				the rows score is 65
29	21 Crowd(q,t)=63	18 Crowd(q,t)=0	17 Crowd(q,t)=2	
				the rows score is 172
30	22 Crowd(q,t)=113	39 Crowd(q,t)=42	32 Crowd(q,t)=17	
				the rows score is 177
31	11 Crowd(q,t)=1	28 Crowd(q,t)=16	27 Crowd(q,t)=160	
				the rows score is 1
32	39 Crowd(q,t)=1	37 Crowd(q,t)=0	22 Crowd(q,t)=0	
				the rows score is 8
33	38 Crowd(q,t)=1	37 Crowd(q,t)=0	20 Crowd(q,t)=7	
				the rows score is 0
34	05 Crowd(q,t)=0	31 Crowd(q,t)=0	19 Crowd(q,t)=0	
				the rows score is 123
35	06 Crowd(q,t)=1	40 Crowd(q,t)=59	36 Crowd(q,t)=63	
				the rows score is 58
36	06 Crowd(q,t)=0	35 Crowd(q,t)=56	40 Crowd(q,t)=2	
				the rows score is 3
37	32 Crowd(q,t)=1	39 Crowd(q,t)=2	33 Crowd(q,t)=0	
				the rows score is 36
38	33 Crowd(q,t)=0	20 Crowd(q,t)=0	37 Crowd(q,t)=36	
				the rows score is 103
39	32 Crowd(q,t)=0	22 Crowd(q,t)=101	37 Crowd(q,t)=2	
				the rows score is 203
40	35 Crowd(q,t)=183	06 Crowd(q,t)=4	36 Crowd(q,t)=16	















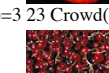







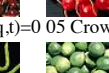



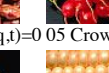



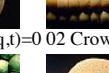











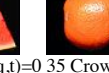

The system score is 12852.0









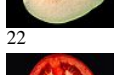


































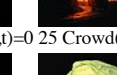

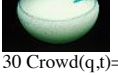









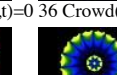











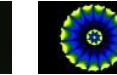












The happiness value is 52

				the rows score is 540
01	08 Crowd(q,t)=261	10 Crowd(q,t)=88	03 Crowd(q,t)=191	
				the rows score is 81
02	39 Crowd(q,t)=30	21 Crowd(q,t)=30	09 Crowd(q,t)=21	
				the rows score is 576
03	08 Crowd(q,t)=189	04 Crowd(q,t)=272	01 Crowd(q,t)=115	
				the rows score is 442
04	03 Crowd(q,t)=295	08 Crowd(q,t)=121	16 Crowd(q,t)=26	
				the rows score is 376
05	06 Crowd(q,t)=309	11 Crowd(q,t)=67	23 Crowd(q,t)=0	
				the rows score is 333
06	11 Crowd(q,t)=72	23 Crowd(q,t)=2	05 Crowd(q,t)=259	
				the rows score is 284
07	09 Crowd(q,t)=178	40 Crowd(q,t)=48	11 Crowd(q,t)=58	
				the rows score is 595
08	03 Crowd(q,t)=233	01 Crowd(q,t)=260	04 Crowd(q,t)=102	
				the rows score is 312
09	07 Crowd(q,t)=239	11 Crowd(q,t)=61	40 Crowd(q,t)=12	
				the rows score is 441
10	01 Crowd(q,t)=136	16 Crowd(q,t)=272	03 Crowd(q,t)=33	
				the rows score is 451
11	09 Crowd(q,t)=98	06 Crowd(q,t)=231	07 Crowd(q,t)=122	
				the rows score is 476
12	14 Crowd(q,t)=226	09 Crowd(q,t)=75	13 Crowd(q,t)=175	
				the rows score is 230
13	14 Crowd(q,t)=201	39 Crowd(q,t)=10	02 Crowd(q,t)=19	
				the rows score is 413
14	13 Crowd(q,t)=147	12 Crowd(q,t)=210	09 Crowd(q,t)=56	
				the rows score is 185
15	07 Crowd(q,t)=72	14 Crowd(q,t)=29	09 Crowd(q,t)=84	
				the rows score is 322
16	10 Crowd(q,t)=240	04 Crowd(q,t)=43	01 Crowd(q,t)=39	
				the rows score is 438
17	18 Crowd(q,t)=268	21 Crowd(q,t)=104	35 Crowd(q,t)=66	
				the rows score is 360
18	17 Crowd(q,t)=236	35 Crowd(q,t)=117	20 Crowd(q,t)=7	
				the rows score is 371
19	38 Crowd(q,t)=76	24 Crowd(q,t)=235	36 Crowd(q,t)=60	
				the rows score is 274
20	35 Crowd(q,t)=108	18 Crowd(q,t)=10	40 Crowd(q,t)=156	

				the rows score is 253
21	17 Crowd(q,t)=99	18 Crowd(q,t)=151	34 Crowd(q,t)=3	
				the rows score is 109
22	38 Crowd(q,t)=4	30 Crowd(q,t)=21	36 Crowd(q,t)=84	
				the rows score is 77
23	35 Crowd(q,t)=53	40 Crowd(q,t)=16	28 Crowd(q,t)=8	
				the rows score is 299
24	36 Crowd(q,t)=31	38 Crowd(q,t)=46	19 Crowd(q,t)=222	
				the rows score is 199
25	28 Crowd(q,t)=178	17 Crowd(q,t)=16	34 Crowd(q,t)=5	
				the rows score is 63
26	17 Crowd(q,t)=15	40 Crowd(q,t)=16	35 Crowd(q,t)=32	
				the rows score is 269
27	32 Crowd(q,t)=207	28 Crowd(q,t)=30	24 Crowd(q,t)=32	
				the rows score is 151
28	33 Crowd(q,t)=149	34 Crowd(q,t)=0	24 Crowd(q,t)=2	
				the rows score is 445
29	30 Crowd(q,t)=311	38 Crowd(q,t)=9	22 Crowd(q,t)=125	
				the rows score is 334
30	29 Crowd(q,t)=305	36 Crowd(q,t)=29	37 Crowd(q,t)=0	
				the rows score is 457
31	28 Crowd(q,t)=16	27 Crowd(q,t)=160	32 Crowd(q,t)=281	
				the rows score is 198
32	27 Crowd(q,t)=110	33 Crowd(q,t)=17	24 Crowd(q,t)=71	
				the rows score is 253
33	28 Crowd(q,t)=149	40 Crowd(q,t)=86	21 Crowd(q,t)=18	
				the rows score is 109
34	28 Crowd(q,t)=14	21 Crowd(q,t)=95	24 Crowd(q,t)=0	
				the rows score is 331
35	17 Crowd(q,t)=97	18 Crowd(q,t)=207	20 Crowd(q,t)=27	
				the rows score is 360
36	38 Crowd(q,t)=148	37 Crowd(q,t)=191	24 Crowd(q,t)=21	
				the rows score is 345
37	36 Crowd(q,t)=262	30 Crowd(q,t)=5	38 Crowd(q,t)=78	
				the rows score is 414
38	36 Crowd(q,t)=201	24 Crowd(q,t)=84	19 Crowd(q,t)=129	
				the rows score is 389
39	22 Crowd(q,t)=101	38 Crowd(q,t)=99	21 Crowd(q,t)=189	
				the rows score is 297
40	23 Crowd(q,t)=24	35 Crowd(q,t)=183	33 Crowd(q,t)=90	

The system score is 123.0

				the rows score is 5
01	08 Crowd(q,t)=3	10 Crowd(q,t)=0	03 Crowd(q,t)=2	
				the rows score is 0
02	39 Crowd(q,t)=0	14 Crowd(q,t)=0	25 Crowd(q,t)=0	
				the rows score is 6
03	08 Crowd(q,t)=2	04 Crowd(q,t)=3	01 Crowd(q,t)=1	
				the rows score is 5
04	03 Crowd(q,t)=3	08 Crowd(q,t)=2	16 Crowd(q,t)=0	
				the rows score is 4
05	06 Crowd(q,t)=3	11 Crowd(q,t)=0	07 Crowd(q,t)=1	
				the rows score is 3
06	05 Crowd(q,t)=3	23 Crowd(q,t)=0	28 Crowd(q,t)=0	
				the rows score is 2
07	09 Crowd(q,t)=2	11 Crowd(q,t)=0	40 Crowd(q,t)=0	
				the rows score is 6
08	01 Crowd(q,t)=3	03 Crowd(q,t)=2	04 Crowd(q,t)=1	
				the rows score is 4
09	07 Crowd(q,t)=3	11 Crowd(q,t)=0	05 Crowd(q,t)=1	
				the rows score is 4
10	01 Crowd(q,t)=1	03 Crowd(q,t)=0	16 Crowd(q,t)=3	
				the rows score is 3
11	09 Crowd(q,t)=0	07 Crowd(q,t)=0	05 Crowd(q,t)=3	
				the rows score is 3
12	14 Crowd(q,t)=3	39 Crowd(q,t)=0	02 Crowd(q,t)=0	
				the rows score is 2
13	14 Crowd(q,t)=2	16 Crowd(q,t)=0	39 Crowd(q,t)=0	
				the rows score is 3
14	12 Crowd(q,t)=3	13 Crowd(q,t)=0	02 Crowd(q,t)=0	
				the rows score is 2
15	14 Crowd(q,t)=0	07 Crowd(q,t)=2	13 Crowd(q,t)=0	
				the rows score is 3
16	04 Crowd(q,t)=0	10 Crowd(q,t)=3	01 Crowd(q,t)=0	
				the rows score is 3
17	18 Crowd(q,t)=3	21 Crowd(q,t)=0	38 Crowd(q,t)=0	
				the rows score is 3
18	17 Crowd(q,t)=3	20 Crowd(q,t)=0	35 Crowd(q,t)=0	
				the rows score is 3
19	38 Crowd(q,t)=0	24 Crowd(q,t)=3	36 Crowd(q,t)=0	
				the rows score is 0
20	18 Crowd(q,t)=0	17 Crowd(q,t)=0	38 Crowd(q,t)=0	

				the rows score is 0
21	17 Crowd(q,t)=0	34 Crowd(q,t)=0	30 Crowd(q,t)=0	
				the rows score is 1
22	29 Crowd(q,t)=0	30 Crowd(q,t)=0	21 Crowd(q,t)=1	
				the rows score is 5
23	17 Crowd(q,t)=2	40 Crowd(q,t)=0	18 Crowd(q,t)=3	
				the rows score is 3
24	38 Crowd(q,t)=0	19 Crowd(q,t)=3	36 Crowd(q,t)=0	
				the rows score is 3
25	17 Crowd(q,t)=0	28 Crowd(q,t)=2	33 Crowd(q,t)=1	
				the rows score is 3
26	35 Crowd(q,t)=0	02 Crowd(q,t)=0	25 Crowd(q,t)=3	
				the rows score is 2
27	32 Crowd(q,t)=2	28 Crowd(q,t)=0	24 Crowd(q,t)=0	
				the rows score is 4
28	33 Crowd(q,t)=1	32 Crowd(q,t)=0	25 Crowd(q,t)=3	
				the rows score is 5
29	30 Crowd(q,t)=3	22 Crowd(q,t)=2	38 Crowd(q,t)=0	
				the rows score is 4
30	29 Crowd(q,t)=3	37 Crowd(q,t)=0	36 Crowd(q,t)=1	
				the rows score is 5
31	27 Crowd(q,t)=2	28 Crowd(q,t)=0	32 Crowd(q,t)=3	
				the rows score is 0
32	33 Crowd(q,t)=0	28 Crowd(q,t)=0	21 Crowd(q,t)=0	
				the rows score is 1
33	28 Crowd(q,t)=1	21 Crowd(q,t)=0	32 Crowd(q,t)=0	
				the rows score is 3
34	21 Crowd(q,t)=3	30 Crowd(q,t)=0	33 Crowd(q,t)=0	
				the rows score is 5
35	18 Crowd(q,t)=3	17 Crowd(q,t)=2	21 Crowd(q,t)=0	
				the rows score is 5
36	37 Crowd(q,t)=3	38 Crowd(q,t)=2	30 Crowd(q,t)=0	
				the rows score is 3
37	36 Crowd(q,t)=3	30 Crowd(q,t)=0	21 Crowd(q,t)=0	
				the rows score is 4
38	19 Crowd(q,t)=0	36 Crowd(q,t)=3	24 Crowd(q,t)=1	
				the rows score is 3
39	38 Crowd(q,t)=0	22 Crowd(q,t)=0	18 Crowd(q,t)=3	
				the rows score is 0
40	23 Crowd(q,t)=0	17 Crowd(q,t)=0	33 Crowd(q,t)=0	