Prob #11297 - Cencus

a) Discussion of the proposed algorithm (give the actual code as well) – how the solution was conceptualized? What solution strategy was adopted? What are the issues encountered in constructing the algorithm/solution?

Source Code:

```
#include <stdio.h>
int main(){
	int x;
	int y;
	int xPos;
	int yPos;
	int Q;
	int ctr = 0;
	char choice;
	int x1;
	int y1;
	int x2;
	int y2;
	int min;
	int max;
	int xCtr;
	int yCtr;
	int xCor;
	int yCor;
	int value;

	scanf("%d", &x);
	scanf("%d", &y);

	if((0 <= x && x <= 500) && (0 <= y && y <= 500)){
		int table [x][y];

		for(xPos = 0; xPos < x; xPos++){
			for(yPos = 0; yPos < y; yPos++){
				scanf("%d", &table[xPos][yPos]);
			}
		}

		scanf("%d", &Q);

		if(Q <= 40000){

			while(ctr < Q){
				scanf("%c", &choice);
				if(choice == 'q'){
					scanf("%d", &x1);
					scanf("%d", &y1);
					scanf("%d", &x2);
					scanf("%d", &y2);

					min = table[x1-1][y1-1];
					max = table[x1-1][y1-1];
```

```
                        for(xCtr = x1-1; xCtr < x2; xCtr++){
                              for(yCtr = y1-1; yCtr < y2; yCtr++){
                                    if(table[xCtr][yCtr] < min)
                                          min = table[xCtr][yCtr];
                                    else if(table[xCtr][yCtr] > max)
                                          max = table[xCtr][yCtr];
                              }
                        }

                        printf("%d %d\n", max, min);
                        ctr++;
                  }

                  else if(choice == 'c'){
                        scanf("%d", &xCor);
                        scanf("%d", &yCor);
                        scanf("%d", &value);
                        table[xCor-1][yCor-1] = value;
                        ctr++;
                  }
            }
      }
      return 0;
}
```

In my solution, similar to how the input of population would appear, I used a matrix to lay out the inputs and we assume that the index starts at 1. To check for the maximum and minimum population in the cities within the range of the coordinates that is user-input, (x1, y1) being the upper left and (x2,y2) being the lower right, first, the value in (x1, y1) will be assumed as the minimum and maximum. Then I check for the minimum and maximum from left-right (y1 to y2), up-down (x1 to x2). To change an element or population number in the matrix, the user will be asked to input a coordinate (xCor, yCor) and a value and the existing element in the given coordinate (xCor, yCor) will simply be over-written by the value. In constructing this algorithm, the only challenge that I faced was that I was a bit confused of the indexing thus committing minor errors.

b) Frequency count for each line code and program's overall frequency count and time complexity.

Computing for the frequency count of the algorithm:

```
int ctr = 0;                                                              (1)
scanf("%d", &x);                                                          (1)
scanf("%d", &y);                                                          (1)

if((0 <= x && x <= 500) && (0 <= y && y <= 500)){                         (1)
      int table [x][y];                                                   (0)

      for(xPos = 0; xPos < x; xPos++){                                    (2x+2)
            for(yPos = 0; yPos < y; yPos++){                              (2xy+2x)
                  scanf("%d", &table[xPos][yPos]);                        (xy)
            }
      }

      scanf("%d", &Q);                                                    (1)

      if(Q <= 40000){                                                     (1)

            while(ctr < Q){                                               (Q+1)
                  scanf("%c", &choice);                                   (Q)
                  if(choice == 'q'){                                      (Q)
                        scanf("%d", &x1);                                 (Q)
                        scanf("%d", &y1);                                 (Q)
                        scanf("%d", &x2);                                 (Q)
                        scanf("%d", &y2);                                 (Q)

                        min = table[x1-1][y1-1];                          (Q)
                        max = table[x1-1][y1-1];                          (Q)

                        for(xCtr = x1-1; xCtr < x2; xCtr++){              (2Qx2-2Qx1)
                              for(yCtr = y1-1; yCtr < y2; yCtr++){  (2Qy2-2Qy1)(Qx2-Qx1-1)
                                    if(table[xCtr][yCtr] < min)       (Qy2-Qy1)(Qx2-Qx1)
                                          min = table[xCtr][yCtr];
                                    else if(table[xCtr][yCtr] > max)  (Qy2-Qy1)(Qx2-Qx1)
                                          max = table[xCtr][yCtr];    (Qy2-Qy1)(Qx2-Qx1)
                              }
                        }

                        printf("%d %d\n", max, min);                      (Q)
                        ctr++;                                            (Q)
                  }

                  else if(choice == 'c'){
                        scanf("%d", &xCor);
                        scanf("%d", &yCor);
                        scanf("%d", &value);
                        table[xCor-1][yCor-1] = value;
                        ctr++;
                  }
            }
      }
}
return 0;                                                                 (1)
```

The total frequency count of the program is: 2(Qy2-Qy1)(Qx2-Qx1-1)+ 3(Qy2-Qy1)(Qx2-Qx1) + 2(Qx2-Qx1) + 11Q + 3xy + 4x + 9

We then assume that Q is any constant c because it is user-input dependent. We also assume that x and y and all inputs that are within that range, x2, x1, y2, and y1, are all considered as n.

Rewriting the function by substituting, we get:

$2(cn-cn)(cn-cn-1) + 3(cn-cn)(cn-cn) + 2(cn-cn) + 11c + 3n^2 + 4n + 9 = 11c + 3n^2 + 4n + 9$

Therefore the Big-Oh of the function is $O(n^2)$.

c) Discuss as well the difficulties encountered when submitting solutions to UVA – What are the problems encountered and how the group handled and solved them?

I first submitted the codes in JAVA language to UVA. After submission, I wasn't aware that the class name should be "Main" and not any other names; I first used "Driver" as my class name because this is what we practiced in our OBJECTP class. After solving this problem, we faced yet another problem, the running time of the JAVA program exceeded the time limit. After numerous attempts of submitting the same program, it still exceeded the time limit. Then I converted the JAVA program to a C program and didn't face any problems at all. After this activity, I realized that C runs faster than JAVA.