CSC346/646 Parallel Computation Spring 2015

K-means Clustering – Implementation Notes

For our first project, we will develop a parallel program using OpenMP to do K-means clustering. Your main program should accept the name of a data file and a positive integer k. E.g. `% cluster datafile 12`

Output:

Let $m \times n$ denote the size of the input data. Let us agree to name the clusters using the integers $\{0, 1, 2, 3, ..., k\}$. All input pixels which are 0 (i.e., nothing there in the scene) belong to cluster number 0. Clusters numbered $1, 2, 3, ..., k$ correspond to the input pixels with value 1.

Your program output must include:

- A file named `cluster.idat` containing an $m \times n$ matrix C, where $C_{i,j}$ is an integer in the set $\{0, 1, 2, 3, ..., k\}$. The format of the file must be the same as the input file.
- A file named `means.txt` containing the k points representing the final means (cluster centroids).
- The run-time (wall clock) time taken by your code, excluding time taken by input and output.

Suggestions:

- Use a simple representation of data points. E.g.,

```
class point {
   public:
      float x ;
      float y ;
} ;
```

- Use a 1-D array length $k + 1$ of points to represent the means (centroids). Let's call this array `means[]`.
- Use a 2-D array of integers size $m \times n$ representing the cluster number for each pixel.
- To update the array of means, you will need some way to accumulate the appropriate sums. You can re-initialize the `means[]` array to zero, and accumulate the new sums in the `means[]` array.
- You will also need some way to keep track of the number of points that have been added. You will most likely need an auxiliary array of integers length $k + 1$ to keep track of how many points have been added for each cluster index.
- To help with timing: see *http://menehune.opt.wfu.edu/csc346* for C/C++ source file `etime.h` and `etime.c` (or `etime.cc`).