Brian Huntley

Design Doc 3

Part A

1) What is the abstract thing you are trying to represent?
    a) A blocked two dimensional array of values. Values inside the same block will be close to each other in memory. The outer array will contain inner arrays, which will represent blocks.
2) What functions will you offer, and what are the contracts that those functions must meet?
    a) extern T UArray2b_new (int width, int height, int size, int blocksize);
        This function returns a new Uarray2b with the given width, height, value size, and block size. Block size is the square root of the number of cells in the block.

        extern T UArray2b_new16K_block (int width, int height, int size, int blocksize);
        This function returns a new Uarray2b with the given width, height, value size, and block size. Block size as large as possible, up to 16kb.

        extern void UArray2b_free(T *Uarray2b);
        This function frees the memory used by the given Uarray2b.

        extern int UArray2b_width(T Uarray2b);
        This function returns the number of columns in the given Uarray2b.

        extern int UArray2b_height(T Uarray2b);
        This function returns the number of rows in the given Uarray2b.

        extern int UArray2b_size(T Uarray2b);
        This function returns the size in bytes of the values inside the given Uarray2b.

        extern int UArray2b_blocksize(T Uarray2b);
        This function returns the blocksize used in the specified Uarray2b.

        extern void *UArray2b_at(T Uarray2b, int row, int col);

This function returns a pointer to the values at the specified position in the given Uarray2b.

extern void UArray2b_map(T Uarray2b, void apply(int c, int r, T array2b,void *val, void *cl), void *cl);

This function calls the given apply function on each value. It does this by visiting each element of a block before moving onto the next block.

3) What examples do you have of what the functions are supposed to do?
    a) Uarray2b arr = uarray2b_new(10, 10, sizeof(int), 4)
        i) Should return an 10x10 array of integers, split into blocks of size 4.
    b) Int w = uarray2b_width(arr)
        i) w should == 10
    c) Int h = uarray2b_height(arr)
        i) h should == 10
    d) Int s = uarray2b_size(arr)
        i) s should == 4
    e) UArray_at(arr, 1, 1) = 5
        i) The values at (1, 1) should be == 5
4) What representation will you use, and what invariants will it satisfy?
    a) My uarray2b implementation will be represented as an array of arrays, where the inner arrays represent blocks. The 2d array to hold the blocks will be my UArray2 implementation from the last assignment.

        To translate from block and cell index back to pixel coordinates (i, j):
        Block index / (1+(width / blocksize)) will give me the row of the block
        Block index % (1+(width / blocksize)) will give the column of the block
        Similarly, (cell index / blocksize) will give the row of the cell
        (cell index % blocksize) will give the column of the cell
        Finally
        i = (Block col * blocksize) + cell col
        j = (block row * blocksize) + cell row
    b) It will satisfy the following invariants:
        i) The value at (i, j) can be found by first selecting which block the value is in using this equation: (i / blocksize, j / blocksize). Then, inside the block from the previous equation I can find the value at (blocksize * (j % blocksize) + i % blocksize).
        ii) The height, width, and size of the array will always be positive integers.

iii) The array will always point to a block of memory large enough to hold (width * height) elements of the specified size.

5) When a representation satisfies all invariants, what abstract thing does it represent?
    a) When all invariants are satisfied I will be left with a two dimensional blocked array, where values inside of each block are near each other in memory.
6) What test cases have you devised?
    a) Using different value types
    b) Using improper heights and widths
    c) Trying to access values that are outside of the bounds of the array
    d) Giving a value that is the incorrect size
7) What programming idioms will you need?
    a) Handling void * values
    b) Using unboxed arrays
    c) Allocating memory

Part C

1. What problem are you trying to solve?
    a. Implementing various image transformations such as rotations, horizontal and vertical flipping, and transpositions. And how locality changes the speed at which these transformations can be computed.
2. What example inputs will help illuminate the problem?
    a. Large PNM images. The images should be large in order to measure how the performance responds to each mapping function.
3. What example outputs go with those example inputs?
    a. The output will be a PNM image that has been transformed according to what the user enters.
4. Into what steps or subproblems can you break down the problem?
    a. Reading PNM images
    b. Different mapping functions to access each pixel.
    c. Each different transformation will be its own function.
    d. Writing PNM images
5. What data are in each subproblem?
    a. A PNM image to read from.
    b. An argument that specifies what to do with the image.
    c. A 2d array which represents this image.
6. What code or algorithms go with that data?
    a. Each transformation will need its own algorithm.

    b. My implementation of 2d arrays will be used for storing the PNM images.
    c. The Pnmrdr interface will be needed to read PNM images.
7. What abstractions will you use to help solve the problem?
    a. UArray2 and UArray2b will be used as a representation of images. The A2Methods abstraction will also be used, to avoid repeating code between UArray2 and UArray2b.
8. If you have to create new abstractions, what are their design checklists?
    a. Part A
9. What invariant properties should hold during the solution of the problem?
    a. For row major transformations- Every pixel directly to the left of the current pixel has been transformed, also every pixel in rows above the current pixel have been transformed.
    b. For column major transformations - Every pixel directly above the current pixel has been transformed, and all pixels to the left of the current column have been transformed.
    c. For block major transformations - Every block directly to the left of the current block has been transformed, and every block in rows above the current block have also been transformed.
10. What algorithms might help solve the problem?
    a. Algorithms to perform transformations such as rotations, flips, and transpositions.