Assignment8 / {XKCompressionD}

Graphics Programming / Tristan Goodell

Methodology

- To compress an XKCD comic, the image is converted to black and white. Then, a String of 1s (white pixels) and 0s (black pixels) is constructed.
- From there, an array is created that goes through the String and counts the number of black and white pixels in a row. If there are more than 255 pixels of the same color in a row, the the array places a 0 in the next index and then resumes in the index after that.
- The array is then converted to binary and then added to *output.tzg*.
- To decompress an image, the process is the exact opposite.

BW, writeHeaders, & getPixelCount

```
def blackWhite(img, threshold):
    bw = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    bw[np.uint8(bw) < threshold] = 0
    bw[np.uint8(bw) > threshold] = 255
    return bw

def writeHeaders(w,h,f):
    f.write(b"TZG")
    aSeriesOfBytes = struct.pack("<2I", w, h)
    f.write(aSeriesOfBytes)
```

```
def getPixelCount(pixels,startIndex,count=0):
    counter=count
    index=startIndex
    val=pixels[index]
    while counter<255 and index<len(pixels)-1 and
         pixels[index]==val:
        index+=1
        counter+=1

    return counter
```

save

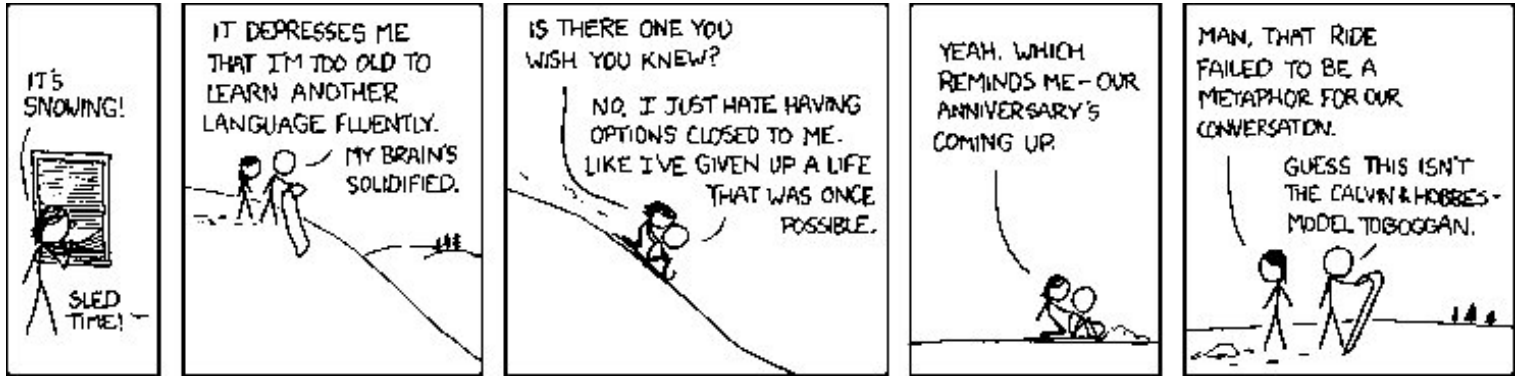
```python
def save(img,filename):
    w,h=img.shape
    f=open(filename+".tzg","wb")
    writeHeaders(w,h,f)

    pixels=""

    for row in img:
        for pixel in row:
            if pixel==255:
                pixels+="1"
            else:
                pixels+="0"

    pixelsSize=len(pixels)
    index=0
    turn=0
    npixels=[]
```

```python
    while index<pixelsSize-1:
        if turn%2==0 and pixels[index]=="0":
            num=getPixelCount(pixels,index,0)
            npixels.append(num)
            index+=num
        elif turn%2==1 and pixels[index]=="1":
            num = getPixelCount(pixels,index,0)
            npixels.append(num)
            index+=num

        turn+=1

    npixels.append(1)

    bnpixels=[]

    for pixel in npixels:
        bnpixels.append(bin(pixel))
        f.write(struct.pack("<B", pixel))

    f.close()
```

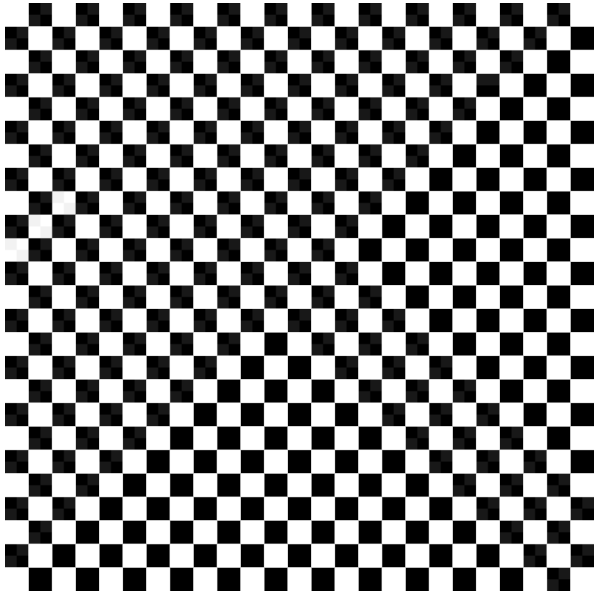
read

```python
def read(filename):
    f = open("%s.tzg" % filename, "rb")
    x = f.read(3)
    if x != b"TZG":
        print("invalid file")
    w, h = struct.unpack("<2I", f.read(8))

    pixels = []
    while 1:
        b = f.read(1)
        if not b:
            break
        pixels.append(struct.unpack("<B", b))

    npixels=[]
    index=0
    while index<len(pixels)-1:
        npixels.append(int(pixels[index][0]))
        index+=1

    npixels.append(1)

    spixels=""
    turn=0
    index=0
    while index<len(pixels)-1:
        if turn%2==1:
            spixels+=npixels[index]*"0"
        elif turn%2==0:
            spixels+=npixels[index]*"1"
        turn+=1
        index+=1

    spixels+="0"

    m = np.array(list(spixels))
    img = np.uint8(np.reshape(m[:w * h], (w, h)) == "1") * 255
    return img
```

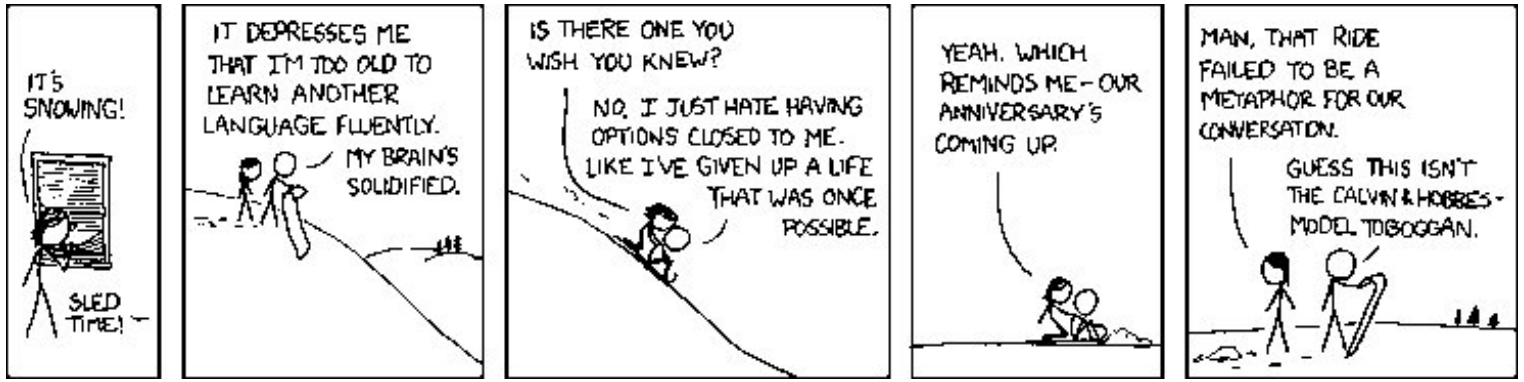
Case 1: #529



Black_and_White.png - 11.2 KB

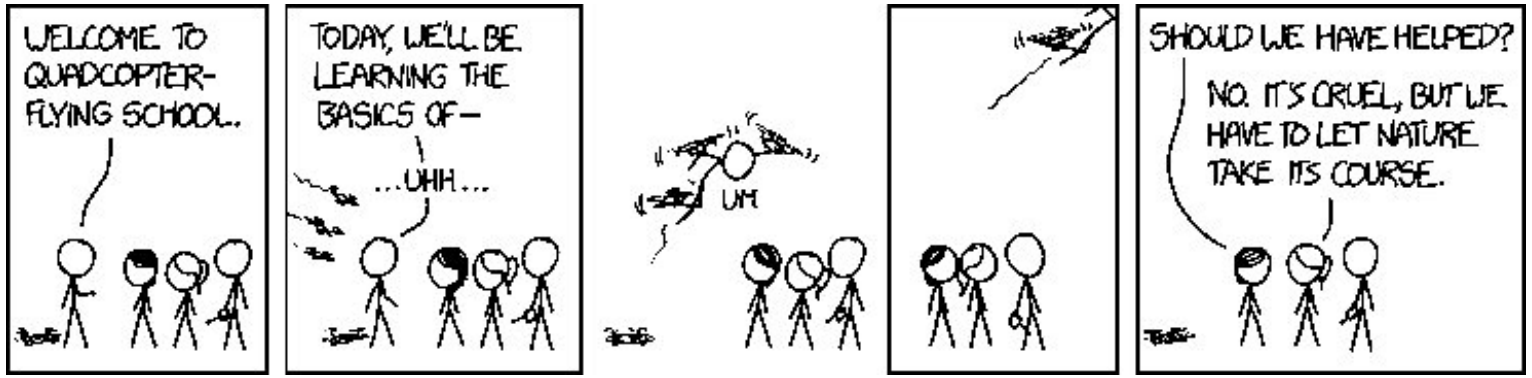


Compressed.tzg
13.9 KB
0.82 bits/pixel

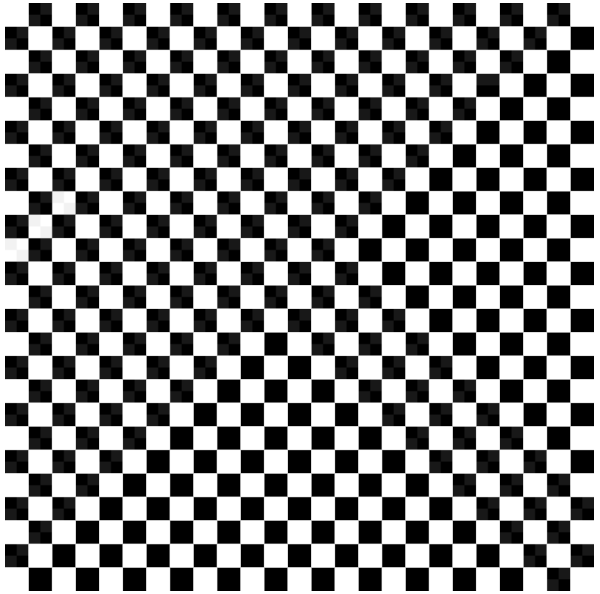


Uncompressed.png - 11.1 KB

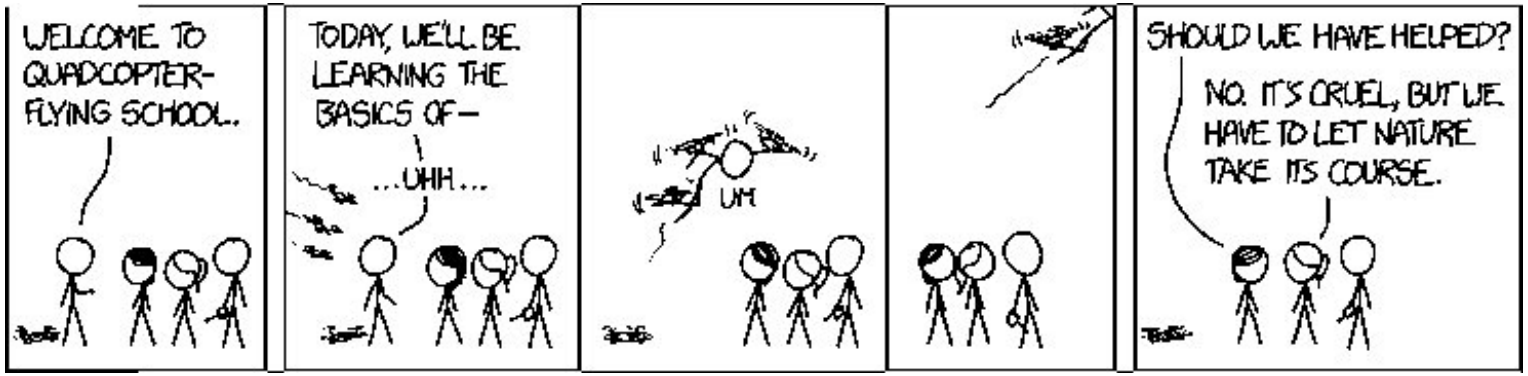
Case 2: #1630



Black_and_White.png - 10.6 KB

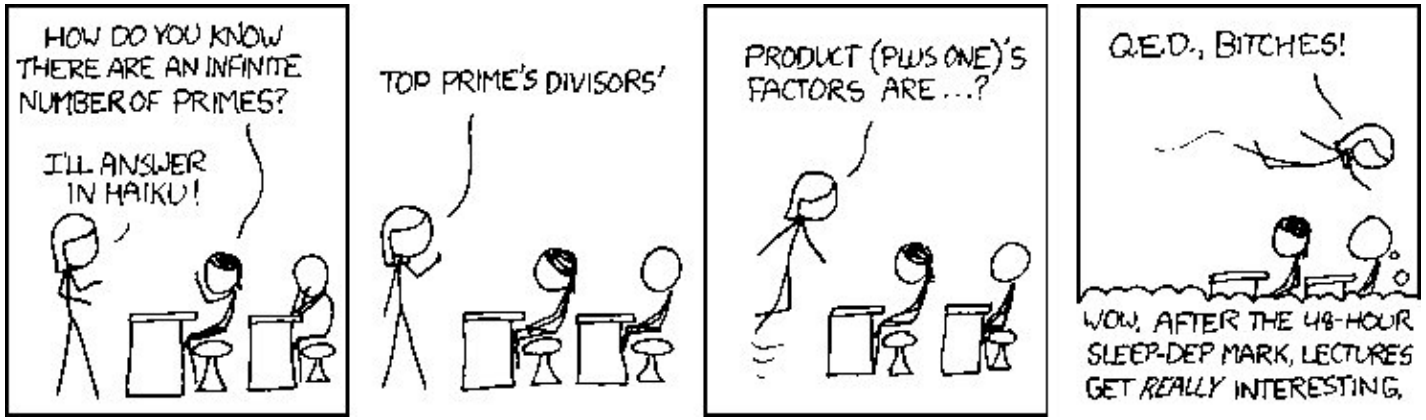


Compressed.tzg
11.7 KB
0.7 bits/pixel

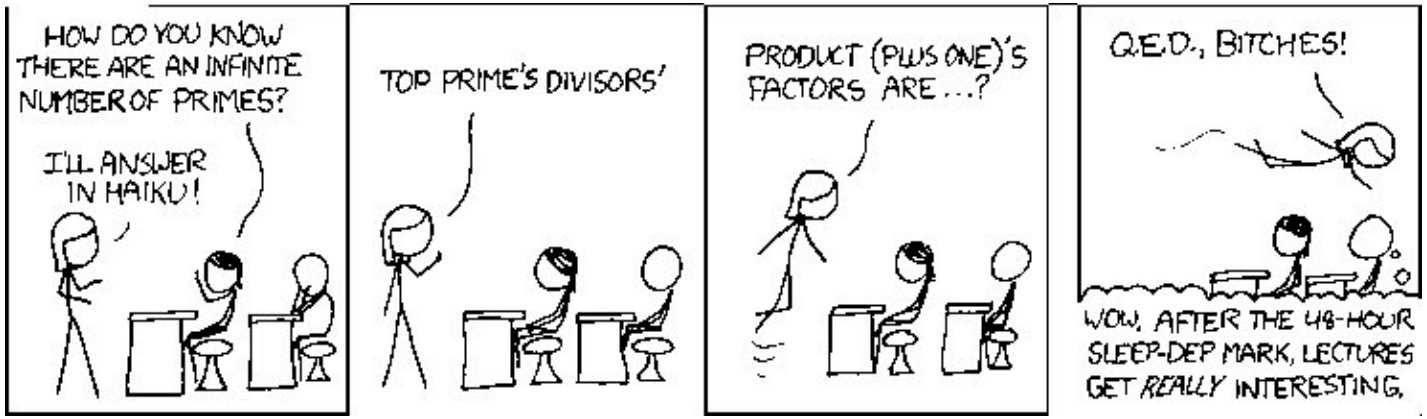


Uncompressed.png - 10.6 KB

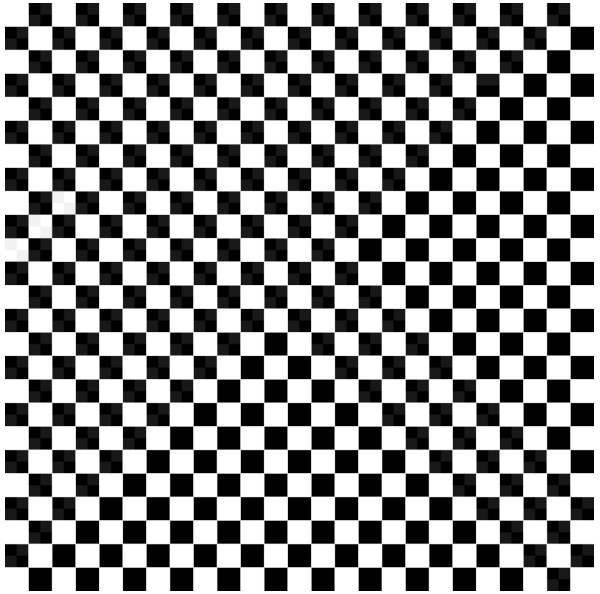
Case 3: #622



Black_and_White.png - 11.6 KB



Uncompressed.png - 11.5 KB



Compressed.tzg
12.6 KB
0.63 bits/pixel