GRIP-THE SPARKS FOUNDATION

Computer Vision and IOT Internship

AUTHOR - AYUSH CHHOKER

TASK 1: OPTICAL CHARACTER RECOGNITION

Import libraries

```
import tensorflow as tf
import tensorflow_hub as hub

import matplotlib.pyplot as plt
import numpy as np

from PIL import Image
from PIL import ImageColor
from PIL import ImageDraw
from PIL import ImageFont
from PIL import ImageOps
```

Preprocessing

```
def display_image(image):
  fig = plt.figure(figsize=(20, 15))
  plt.axis('off')
  plt.imshow(image)

def load_img(path):
  img = tf.io.read_file(path)
  img = tf.image.decode_jpeg(img, channels=3)
  return img
```

Draw bounding box and put class name on the image

```
def draw(image, max_boxes, min_score, boxes, class_names, scores):
  '''
  -- params --
```

```
  max boxes : maximum no of bounding boxes you want to show in the image
  min_score: threshold value for your desired detection score
  '''
  # get color values from PIL.Image.Color
  colors = list(ImageColor.colormap.values())

  # get font name from PIL.ImageFont
  try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationSansNarrow-Regular.ttf
                              25)
  except IOError:
    print("Font not found, using default font.")
    font = ImageFont.load_default()

  detected_boxes = boxes.shape[0]

  for i in range(min(max_boxes, detected_boxes)):
    if scores[i] >= min_score: # if detection score >= your desired score

      ymin, xmin, ymax, xmax = tuple(boxes[i]) # bbox coordinate values

      # decode detected object name with score, example, b'Umbrella' --> Umbrella: 92%
      display_str = "{}: {}%".format(class_names[i].decode("ascii"),
                                     int(100 * scores[i]))

      # set bbox color same to every same class
      color = colors[hash(class_names[i]) % len(colors)]

      # Convert Image to numpy type and RGB
      image_pil = Image.fromarray(np.uint8(image)).convert("RGB")

      draw_bbox_text_on_image(
          image_pil,
          ymin, xmin,
          ymax, xmax,
          color, font,
          display_str_list=[display_str]
      )
      np.copyto(image, np.array(image_pil))

  return image

def draw_bbox_text_on_image(image,
                            ymin, xmin,
                            ymax, xmax,
                            color, font, thickness=4,
                            display_str_list=()):
```

```
    """ Calculates bbox coordinates, text coordinates and draw """

    # creates PIL Draw Object
    draw = ImageDraw.Draw(image)
    im_width, im_height = image.size

    # Formula for bbox coordinate calculation
    (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                                  ymin * im_height, ymax * im_height)

    # Draw bbox with coordinates
    draw.line([(left, top), (left, bottom), (right, bottom), (right, top),
               (left, top)],
              width=thickness,
              fill=color)

    # get font size from String list
    display_str_heights = [font.getsize(ds)[1] for ds in display_str_list] # 11

    # Each display_str has a top and bottom margin of 0.05x.
    total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights) # 12


    # calculation for showing better class label position with bbox
    if top > total_display_str_height:
      text_bottom = top
    else:
      text_bottom = top + total_display_str_height

    for display_str in display_str_list:
      text_width, text_height = font.getsize(display_str)
      margin = np.ceil(0.05 * text_height)

      # draw rectangle with color
      draw.rectangle([(left, text_bottom - text_height - 2 * margin),
                      (left + text_width, text_bottom)],
                     fill=color)

      # put text in drawn colored rectangle
      draw.text((left + margin, text_bottom - text_height - margin),
                display_str,
                fill="black",
                font=font)
```

detector function

```
def run_detector(detector, path, max_boxes, max_score):
```

```
def run_detector(detector, path, max_boxes, min_score):
  '''
  -- params --
  max boxes : maximum no of bounding boxes you want to show in the image
  min_score: threshold value for your desired detection score
  '''

  img = load_img(path)

  # convert jpeg into tf.float32 fromat and add batchsize 1 as dimension
  converted_img= tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]
  print('Detecting Image ...')
  output = detector(converted_img) # Returns a dictionary containing bbox values, classnames, scores
  print('Detection Complete\n')

  for key, value in output.items():
    print(f'Key:{key}\nValue:{value}\n\n')

  output = {key:value.numpy() for key,value in output.items()}
  print("Found %d objects." % len(output["detection_scores"]))

  image_with_boxes = draw(img.numpy(),
                          max_boxes, max_score,
                          output['detection_boxes'],
                          output['detection_class_entities'],
                          output['detection_scores'])
  display_image(image_with_boxes)
```

Capture picture from webcam and save

```
from IPython.display import HTML, Audio
from google.colab.output import eval_js
from base64 import b64decode
import numpy as np
import io
from PIL import Image

VIDEO_HTML = """
<div class="video_container">
  <video autoplay
   width=%d height=%d></video>
  <div style='position: absolute;top: 40px; left: 40px; font-size: 40px; color: green;'>Click to sav
</div>
<script>
var video = document.querySelector('video')
navigator.mediaDevices.getUserMedia({ video: true })
  .then(stream=> video.srcObject = stream)
```

```
var data = new Promise(resolve=>{
  video.onclick = ()=>{
    var canvas = document.createElement('canvas')
    var [w,h] = [video.offsetWidth, video.offsetHeight]
    canvas.width = w
    canvas.height = h
    canvas.getContext('2d')
          .drawImage(video, 0, 0, w, h)
    video.srcObject.getVideoTracks()[0].stop()
    video.replaceWith(canvas)
    resolve(canvas.toDataURL('image/jpeg', %f))
  }
})
</script>
"""

def take_photo(filename=None, quality=0.8, size=(800,600)):
  """ Take photo from webcam and save it"""
  handle = display(HTML(VIDEO_HTML % (size[0],size[1],quality)), display_id='videoHTML')
  data = eval_js("data")
  binary = b64decode(data.split(',')[1])

  if filename:
    f = io.BytesIO(binary)
    Image.open(f).save(filename)
  else:
    f = io.BytesIO(binary)
    return np.asarray(Image.open(f))

take_photo('me.jpg')
```

Download model and predict

```
faster_rcnn_url = "https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1"

# Download sample image using Wget
#!wget -O sample1.jpg 'https://www.istockphoto.com/photo/new-industrial-textile-factory-gm477686364-
#!wget -O sample2.jpg 'https://www.istockphoto.com/vector/back-to-school-icon-set-with-50-different-

# Load model from tensorflow hub
print('loading model...')
detector = hub.load(faster_rcnn_url).signatures['default']
print('model loaded!')
```

```
loading model...
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
model loaded!
```

```
# Detect Image from webcam
run_detector(detector, 'me.jpg',  max_boxes=8, max_score=0.2) # >=20% accuracy
```

```
Detecting Image ...
Detection Complete

Key:detection_class_entities
Value:[b'Human face' b'Man' b'Person' b'Clothing' b'Human arm' b'Human nose'
 b'Glasses' b'Human hand' b'Mobile phone' b'Man' b'Human arm' b'Person'
 b'Mobile phone' b'Clothing' b'Person' b'Toy' b'Man' b'Person'
 b'Human head' b'Human arm' b'Man' b'Computer mouse' b'Human hand'
 b'Human hair' b'Person' b'Clothing' b'Person' b'Human hand' b'Guitar'
 b'Man' b'Clothing' b'Clothing' b'Clothing' b'Human arm' b'Human hand'
 b'Human ear' b'Human arm' b'Clothing' b'Human hand' b'Human mouth'
 b'Human eye' b'Woman' b'Clothing' b'Human body' b'Human hair'
 b'Human hand' b'Human arm' b'Clothing' b'Person' b'Computer mouse'
 b'Human leg' b'Human arm' b'Human hand' b'Human mouth' b'Human mouth'
 b'Human hand' b'Laptop' b'Man' b'Human hand' b'Girl' b'Human leg'
 b'Musical instrument' b'Human hand' b'Mammal' b'Human arm'
 b'Bathroom accessory' b'Human hand' b'Clothing' b'Human beard'
 b'Human arm' b'Clothing' b'Man' b'Fashion accessory' b'Clothing'
 b'Human mouth' b'Human arm' b'Mammal' b'Clothing' b'Auto part' b'Tap'
 b'Human beard' b'Camera' b'Boy' b'Human body' b'Clothing' b'Human leg'
 b'Human body' b'Human body' b'Power plugs and sockets' b'Woman'
 b'Human arm' b'Human beard' b'Human hand' b'Clothing' b'Human mouth'
 b'Human hand' b'Human hair' b'Human arm' b'Human arm'
 b'Fashion accessory']


Key:detection_class_names
Value:[b'/m/0dzct' b'/m/04yx4' b'/m/01g317' b'/m/09j2d' b'/m/0dzf4' b'/m/0k0pj'
 b'/m/0jyfg' b'/m/0k65p' b'/m/050k8' b'/m/04yx4' b'/m/0dzf4' b'/m/01g317'
 b'/m/050k8' b'/m/09j2d' b'/m/01g317' b'/m/0138tl' b'/m/04yx4'
 b'/m/01g317' b'/m/04hgtk' b'/m/0dzf4' b'/m/04yx4' b'/m/020lf' b'/m/0k65p'
 b'/m/03q69' b'/m/01g317' b'/m/09j2d' b'/m/01g317' b'/m/0k65p' b'/m/0342h'
 b'/m/04yx4' b'/m/09j2d' b'/m/09j2d' b'/m/09j2d' b'/m/0dzf4' b'/m/0k65p'
 b'/m/039xj_' b'/m/0dzf4' b'/m/09j2d' b'/m/0k65p' b'/m/0283dt1'
 b'/m/014sv8' b'/m/03bt1vf' b'/m/09j2d' b'/m/02p0tk3' b'/m/03q69'
 b'/m/0k65p' b'/m/0dzf4' b'/m/09j2d' b'/m/01g317' b'/m/020lf' b'/m/035r7c'
 b'/m/0dzf4' b'/m/0k65p' b'/m/0283dt1' b'/m/0283dt1' b'/m/0k65p'
 b'/m/01c648' b'/m/04yx4' b'/m/0k65p' b'/m/05r655' b'/m/035r7c'
 b'/m/04szw' b'/m/0k65p' b'/m/04rky' b'/m/0dzf4' b'/m/0h8nr_l' b'/m/0k65p'
 b'/m/09j2d' b'/m/015h_t' b'/m/0dzf4' b'/m/09j2d' b'/m/04yx4' b'/m/0463sg'
 b'/m/09j2d' b'/m/0283dt1' b'/m/0dzf4' b'/m/04rky' b'/m/09j2d'
 b'/m/08dz3q' b'/m/02jz0l' b'/m/015h_t' b'/m/0dv5r' b'/m/01bl7v'
 b'/m/02p0tk3' b'/m/09j2d' b'/m/035r7c' b'/m/02p0tk3' b'/m/02p0tk3'
 b'/m/03bbps' b'/m/03bt1vf' b'/m/0dzf4' b'/m/015h_t' b'/m/0k65p'
 b'/m/09j2d' b'/m/0283dt1' b'/m/0k65p' b'/m/03q69' b'/m/0dzf4' b'/m/0dzf4'
 b'/m/0463sg']


Key:detection_boxes
Value:[[0.         0.46397373 0.35973904 0.7199089 ]
 [0.01097549 0.25570604 0.96445316 0.93365705]
 [0.00826085 0.08073997 1.         0.93251586]
 [0.28505173 0.2262797  1.         0.86321855]
```

```
[0.40262124 0.06789196 1.         0.6614067 ]
[0.10166661 0.57562757 0.20628834 0.6371525 ]
[0.05074981 0.46532348 0.19022253 0.7095651 ]
[0.36666045 0.1626109  0.9135353  0.7182787 ]
[0.29185805 0.15854669 0.7739682  0.54551995]
[0.         0.33642387 0.6839122  0.879372  ]
[0.5734339  0.03268821 0.996341   0.34067565]
[0.         0.34950605 0.91060144 0.89675015]
[0.2543137  0.17915644 0.87958926 0.6391011 ]
[0.2889431  0.13530669 0.94967335 0.662192  ]
[0.         0.3985356  0.5125058  0.76624477]
[0.06143069 0.19462006 0.13556382 0.24492854]
[0.         0.39195532 0.50169194 0.77132064]
[0.30419827 0.         0.9942969  0.84449637]
[0.         0.42860308 0.38662228 0.7426132 ]
[0.34033078 0.21215592 1.         0.7880099 ]
[0.11636534 0.10317837 0.9586428  0.780737  ]
[0.33033922 0.15663196 0.7412917  0.68887925]
[0.44170156 0.28738198 0.9855679  0.7825173 ]
[0.         0.460772   0.15549375 0.7294535 ]
[0.         0.51101464 0.9939516  0.9591209 ]
[0.49542654 0.04823513 0.9993323  0.73955595]
[0.5532282  0.01444784 0.994514   0.3542266 ]
[0.3556584  0.1633829  0.8366173  0.5402405 ]
[0.2913136  0.11568704 0.9802343  0.7333263 ]
[0.         0.5243813  0.98757    0.9632736 ]
[0.24922992 0.3749868  0.5295318  0.7818688 ]
[0.5932844  0.13665938 0.8263139  0.32481265]
[0.24825977 0.40438402 0.9926236  0.96143734]
[0.37684017 0.00973026 0.99516463 0.41547614]
[0.43006584 0.37793517 0.6094953  0.69914234]
[0.07509428 0.4456628  0.19905488 0.4808932 ]
[0.77748954 0.16757368 1.         0.36046714]
[0.27599698 0.40381622 0.48399475 0.72657186]
[0.5814104  0.3034975  0.98859274 0.7004861 ]
[0.22925042 0.5769078  0.28317884 0.6549757 ]
[0.06461106 0.51622826 0.13442859 0.5904942 ]
[0.02248561 0.1481459  1.         0.95254636]
[0.62752336 0.01051226 1.         0.34548935]
[0.0367896  0.10602943 1.         0.9103624 ]
[0.         0.4332994  0.25445157 0.7366423 ]
[0.2657307  0.09602299 0.92318577 0.5758901 ]
[0.5079464  0.10983242 0.9803452  0.8517204 ]
[0.7436851  0.         1.         0.24931693]
[0.         0.279217   0.6697991  0.82675266]
[0.26002204 0.1482704  0.8143369  0.54117554]
[0.6007323  0.01489536 0.96539825 0.3611615 ]
[0.32704583 0.3432827  0.984302   0.9518415 ]
[0.80670375 0.1748133  0.9975756  0.3383975 ]
[0.24154662 0.5688278  0.29230028 0.6462401 ]
[0.22416809 0.5743149  0.27049807 0.6443888 ]
[0.3784207  0.27073675 0.7672889  0.75292474]
[0.2954773  0.10242054 0.9449376  0.6079161 ]
[0.55133396 0.00868412 1.         0.35438496]
```

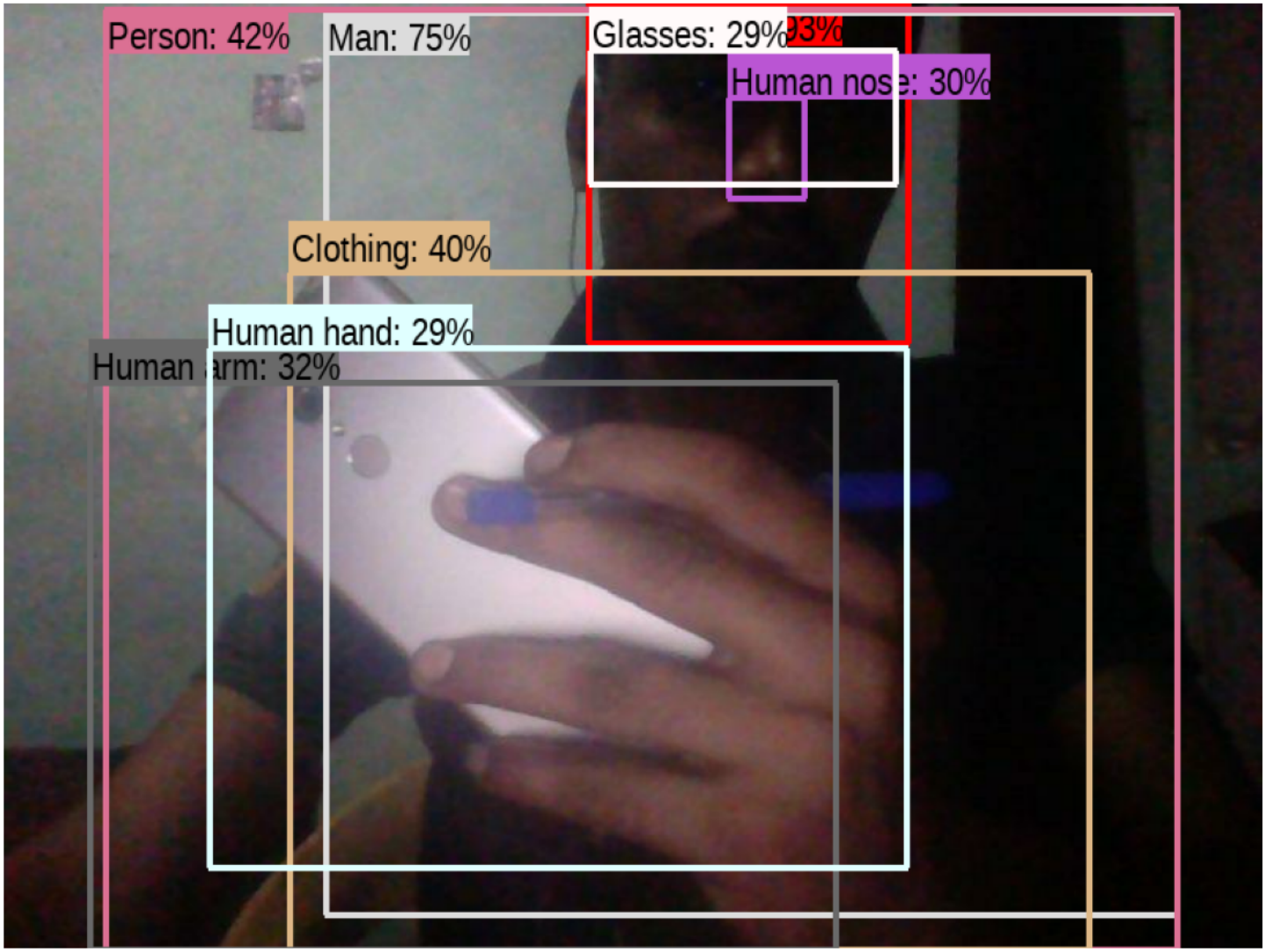
```
 [0.59315497 0.13668203 0.8151526  0.29882523]
 [0.04762095 0.06193992 1.         0.79995674]
 [0.7459426  0.         0.9932282  0.23368824]
 [0.17604457 0.10131203 0.9576734  0.7509682 ]
 [0.7624532  0.01024613 0.9907618  0.21639988]
 [0.         0.29314086 0.9919783  0.9185594 ]
 [0.67758685 0.01498982 0.9859803  0.2637816 ]
 [0.06009274 0.2288569  0.07832292 0.2589912 ]
 [0.6926873  0.01193768 0.9861669  0.29951066]
 [0.6137066  0.09889648 0.96874684 0.5070261 ]
 [0.15028915 0.46153614 0.3705838  0.7017468 ]
 [0.5207449  0.08509181 1.         0.5089561 ]
 [0.81478703 0.16213697 1.         0.34343964]
 [0.42516035 0.04863048 1.         0.9805835 ]
 [0.61471677 0.14076488 0.8083263  0.3209907 ]
 [0.68968606 0.17955102 1.         0.67820096]
 [0.25360554 0.56726164 0.3027644  0.64241654]
 [0.6009737  0.09630316 0.85857344 0.31686005]
 [0.14443807 0.08221488 0.98261    0.7912264 ]
 [0.13211372 0.3413913  0.8762702  0.8712438 ]
 [0.07045615 0.19562268 0.12832431 0.24216253]
 [0.06147436 0.22490694 0.08026613 0.25907776]
 [0.23949108 0.43344787 0.4270272  0.68468237]
 [0.26160234 0.17861287 0.8841449  0.6949439 ]
 [0.06120204 0.13824734 0.9994855  0.81779206]
 [0.         0.34611627 0.9371499  0.8667874 ]
 [0.09819695 0.40277907 0.50309604 0.7634517 ]
 [0.67178965 0.19395396 0.9975855  0.35161772]
 [0.07214025 0.5319551  1.         0.96461743]
 [0.3433409  0.02472736 0.99150485 0.81734145]
 [0.27531323 0.12313707 0.7793395  0.54918486]
 [0.         0.3952445  0.5070644  0.7681323 ]
 [0.6348995  0.15138516 0.99305993 0.679588  ]
 [0.17748874 0.38535106 0.47610402 0.72065747]
 [0.47688377 0.13959663 0.90663874 0.50477844]
 [0.09226146 0.5859069  0.9770512  0.9553472 ]
 [0.23638527 0.55631757 0.2811867  0.63121057]
 [0.5879484  0.01770035 0.95220613 0.32368156]
 [0.         0.42711413 0.41039172 0.7429698 ]
 [0.48409206 0.68660116 0.9502212  0.9662467 ]
 [0.20428602 0.19484746 0.82773197 0.79009026]
 [0.03345291 0.4640313  0.18480775 0.72315985]]


Key:detection_scores
Value:[0.9326563  0.7552265  0.4280183  0.40656832 0.32248774 0.3085529
 0.299116   0.29038474 0.2855914  0.2207314  0.21089062 0.19040574
 0.15480717 0.14660122 0.1240598  0.1164424  0.11439021 0.10611716
 0.08868709 0.07722838 0.06987529 0.06240588 0.06177038 0.05859418
 0.05017703 0.03986519 0.03935039 0.03715076 0.03567476 0.03239058
 0.03227748 0.03115889 0.03079082 0.02950672 0.02760667 0.02653383
 0.0222375  0.02153493 0.02074484 0.02053564 0.02035031 0.02012464
 0.01873795 0.01867978 0.01840124 0.01699001 0.01642441 0.01630761
 0.01605608 0.01600582 0.0152415  0.01521605 0.01486678 0.01464443
```

```
 0.01425982 0.0140967  0.01343626 0.0130267  0.012383   0.0114439
 0.01017415 0.009919   0.00942837 0.00914982 0.00907725 0.00842864
 0.00808864 0.00805795 0.00767387 0.00764175 0.00744786 0.00645922
 0.00600854 0.00588918 0.00579542 0.00576715 0.00564763 0.00541424
 0.0054118  0.00499863 0.00483668 0.00479189 0.00464877 0.00462893
 0.00445887 0.00433985 0.00428872 0.00421755 0.0041766  0.00415169
 0.00414767 0.0041327  0.00413268 0.00395287 0.0039488  0.00390943
 0.00369735 0.00336281 0.00333962 0.0033155 ]


Key:detection_class_labels
Value:[502 308  69 433 503 568 567 573 313 308 503  69 313 433  69  11 308  69
 292 503 308 128 573 253  69 433  69 573 219 308 433 433 433 503 573 224
 503 433 573 148  15 228 433 177 253 573 503 433  69 128 221 503 573 148
 148 573  55 308 573 333 221 301 573 298 503 548 573 433  21 503 433 308
 277 433 148 503 298 433 414 172  21 499  51 177 433 221 177 177 225 228
 503  21 573 433 148 573 253 503 503 277]


Found 100 objects.
```



✓ 44s completed at 1:30 PM