**Q1: Felzenszwalb can already generate segmentations for the whole image, why do we need selective search at all?**

Felzenszwalb generates a static output of atomic regions after segmenting the whole image as an initial segmentation of image into smaller regions. These regions are often fragmented and lack semantic meaning as one object might be split into two regions. Selective search takes those atomic regions and does a hierarchical merging (iteratively) for coherent object like regions and gives merged regions as output. The outputs are generated based on color, texture, size and fill similarities.

**Q2: Proposal filtering is implemented in main.py. What are the criteria based on which the boxes are filtered and what might be the effect? Do you agree with the criteria? Can you think of additional ones?**

The filtering criteria in main.py are:

Duplicate removal: Identical bounding boxes from different segment merges are discarded. This reduces redundancy and computational load.

Size filtering: Excluding regions smaller than 2000 pixels. This focuses on larger, more object-like regions. However, this threshold is dataset-dependent; for high-resolution images, it might discard valid small objects

Aspect ratio filtering: Boxes with width/height >1.2 or height/width >1.2 are removed to avoid elongated shapes. While effective for most objects, this risks excluding valid proposals such as in picture of structures with pillars that have an elongated structure.

As additional criteria we can use:

Edge density: Using filters like canny or sobel, we can calculate edge density within a region. Regions with low edge density are likely not objects.

Color entropy: High entropy implies chaotic color distribution (grass texture), while low entropy suggests uniform regions (sky). By computing entropy of color histogram, we can filter overly noisy or flat regions.

**Q3: Selective Search iteratively merges regions of arbitrary shapes. How do we obtain rectangles (box proposals) from that?**

The criteria for merging would be based on similarity—like color, texture, size, or how well they fit together. The code has functions like sim\_colour and sim\_texture, which calculate similarities. So, regions that are similar enough get merged together. the Felzenszwalb algorithm produces small, atomic regions. These are the starting points. Each of these regions is a segment from the initial segmentation. As the algorithm progresses, Selective Search represents every region (original or merged) by its axis-aligned bounding box (AABB), defined by the extremal coordinates. The small regions are merged into larger ones. So the regions being merged are those that are adjacent and similar according to the similarity measures. each region has 'min\_x', 'max\_x', 'min\_y', 'max\_y' properties. These define the AABB that encloses the region. Even if the region is an irregular shape, the bounding box is determined by the extreme points (min and max coordinates). When regions are merged, the new region's bounding box is the smallest rectangle that contains both original regions. So every time two regions are merged, their combined bounding box is updated to include both. Eventually, each merged region, regardless of its shape, is represented by its enclosing rectangle. All intermediate bounding boxes from every merge step are preserved. The final proposals are all these bounding boxes from the merged regions at various stages.

In short, selective Search starts with small regions, merges similar ones step by step (iteratively), each time updating the bounding box to encompass the merged regions. The final output collects all these bounding boxes from the merged regions at various stages, resulting in rectangular proposals that cover potential objects at different scales and locations.