Complementary Colors

Given integer n and a list of n colors in RGB format (0-255 for each color), please find how many pairs of colors are complementary colors. A pair of colors $((R_1, G_1, B_1), (R_2, G_2, B_2))$ is complementary colors when $R_1 = 255 - R_2$, $G_1 = 255 - G_2$, $B_1 = 255 - B_2$.

Modeling

Given a list of (R, G, B) values

Count all pair (i, j) that:

$R_i = 255 - R_j$

$G_i = 255 - G_j$

$B_i = 255 - B_j$

Naive Approach

```
function complementary(i, j)
    return r[i] + r[j] == 255
        and g[i] + g[j] == 255
        and b[i] + b[j] == 255
```

Naive Approach

```
for i = 0, 1, ..., n - 1
    for j = i + 1, i + 2, ..., n - 1
        if complementary(i, j)
            count += 1
```

Analysis

```
for i = <n times>
    for j = <n - i - 1 times>
        if complementary(i, j)
            count += 1
```

Analysis

```
for i = <n times>
    for j = <n - i - 1 times>
        <some stuff>
        <runs ~0.5n² times in total>
```

What if...

n = 100000?

$$n = 100000$$

$$n = 100000$$
$$0.5n^2 = 5000000000$$

Better Approach

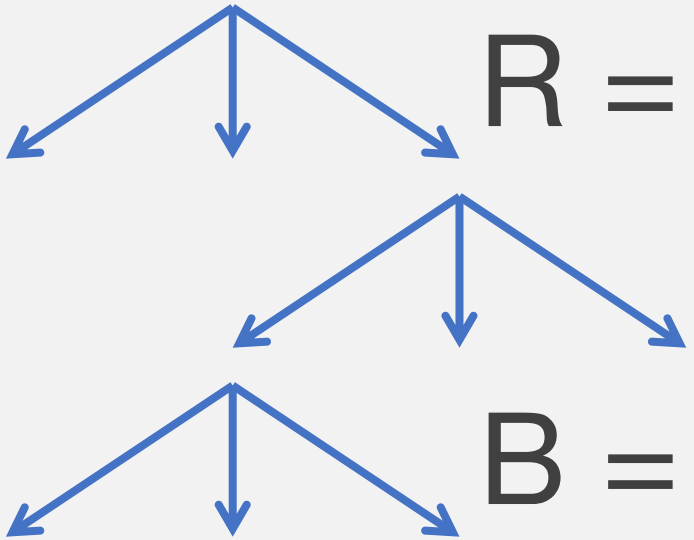
- Tree
- Buckets
- Sorting

Better Approach

- Tree
- Buckets
- Sorting

Tree

Use a three-layer tree structure:



R = 0, 1, ...

G = 0, 1, ...

B = 0, 1, ...

- 2-pass
- Store the total number of each color

Tree

```
for i = 0, 1, ..., n - 1
    leaf = root.access(r[i])
        .access(g[i]).access(b[i])
    leaf.value += 1
for i = 0, 1, ..., n - 1
    leaf = root.access(255 - r[i])
        .access(255 - g[i])
        .access(255 - b[i])
    count += leaf.value
```

Tree

```
for i = 0, 1, ..., n - 1
    leaf = root.access(r[i])
        .access(g[i]).access(b[i])
    leaf.value += 1
for i = 0, 1, ..., n - 1
    leaf = root.access(255 - r[i])
        .access(255 - g[i])
        .access(255 - b[i])
    count += leaf.value
count /= 2 // avoid counting twice!
```

Better Approach

- Tree
- Buckets
- Sorting

Buckets

Similarly, use a big array to store the total number of each color.

$$\text{Size} = 256^3 * \text{size(int)}$$

- 64MB memory consumption (x86)
- Acceptable in most ICPC problems

Buckets

```
for i = 0, 1, ..., n - 1
    buckets[r[i]][g[i]][b[i]] += 1
for i = 0, 1, ..., n - 1
    count += buckets[255 - r[i]]
        [255 - g[i]][255 - b[i]]
count /= 2
```

Better Approach

- Tree
- Buckets
- Sorting

Sorting

- Sort the color values, R, G, B in order
- Point to both ends of the sorted array
- Move toward the center
- Count complementary colors
- Stop when the two pointers meet

Sorting

```
for i = 0, 1, ..., n - 1
    v[i] = r[i] * 1000000
         + g[i] * 1000 + b[i]
sort(v)
pl = 0
pr = n - 1
while pl < pr
    scan()
```

Sorting

```
procedure scan()
    if v[pl] + v[pr] < 255255255
        pl += 1
    else if v[pl] + v[pr] > 255255255
        pr -= 1
    else
        count_local(v[pl], v[pr])
```

Sorting

```
procedure count_local(vl, vr)
    count_l = 0
    count_r = 0
    while v[pl] == vl
        count_l += 1
        pl += 1
    while v[pr] == vr
        count_r += 1
        pr -= 1
    count += count_l * count_r
```