AÑADIR LINEA A UN ARRAY

```
void append(Strip *st)
    {
        // Check for redundant strip, a strip is
        // redundant if it has same height or left as previous
        if (n>0 && arr[n-1].ht == st->ht)
            return;
        if (n>0 && arr[n-1].left == st->left)
        {
            arr[n-1].ht = max(arr[n-1].ht, st->ht);
            return;
        }

        arr[n] = *st;
        n++;
    }
```

ALGORITMO DIVIDE Y VENCERÁS

```
SkyLine *findSkyline(Building arr[], int l, int h)
{
    if (l == h)
    {
        SkyLine *res = new SkyLine(2);
        res->append(new Strip(arr[l].left, arr[l].ht));
        res->append(new Strip(arr[l].right, 0));
        return res;
    }

    int mid = (l + h)/2;

    // Recur for left and right halves and merge the two results
    SkyLine *sl = findSkyline(arr, l, mid);
    SkyLine *sr = findSkyline(arr, mid+1, h);
    SkyLine *res = sl->Merge(sr);

    // To avoid memory leak
    delete sl;
    delete sr;

    // Return merged skyline
    return res;
}
```

FUNCIÓN MEZCLAR

```
SkyLine *SkyLine::Merge(SkyLine *other)
{
    // Create a resultant skyline with capacity as sum of two
    // skylines
    SkyLine *res = new SkyLine(this->n + other->n);

    // To store current heights of two skylines
    int h1 = 0, h2 = 0;

    // Indexes of strips in two skylines
    int i = 0, j = 0;
    while (i < this->n && j < other->n)
    {
        // Compare x coordinates of left sides of two
        // skylines and put the smaller one in result
        if (this->arr[i].left < other->arr[j].left)
        {
            int x1 = this->arr[i].left;
            h1 = this->arr[i].ht;

            // Choose height as max of two heights
            int maxh = max(h1, h2);

            res->append(new Strip(x1, maxh));
            i++;
        }
        else
        {
            int x2 = other->arr[j].left;
            h2 = other->arr[j].ht;
            int maxh = max(h1, h2);
            res->append(new Strip(x2, maxh));
            j++;
        }
    }

    // If there are strips left in this skyline or other
    // skyline
    while (i < this->n)
    {
        res->append(&arr[i]);
        i++;
    }
    while (j < other->n)
    {
        res->append(&other->arr[j]);
        j++;
```

```
    }
    return res;
}
```