**MERGE SORT**

|  |  |
| --- | --- |
| def merge\_sort(A, p, r): | **MERGE-SORT(A, p, r)** |
| if p < r:  q = (p+r)//2  merge\_sort(A, p, q)  merge\_sort(A, q+1, r)  merge(A, p, q, r) | If p < r  q = (p+r)/2  MERGE-SORT(A, p, q)  MERGE-SORT(A, q+1, r)  MERGE(A, p, q, r) |
| def merge(A, p, q, r): | **MERGE(A, p, q, r)** |
| nL = length of A[p:q] , nR= length of A[q+1: r]  copy A[p:q] into L[0,nL-1], copy A[q+1:r] into L[0,nR-1] | |
| L = A[p:q+1]; L.append(math.inf); i=0  R = A[q+1:r+1]; R.append(math.inf); j=0 | nL = q – p +1; nR = r – q  let L[0 :nL-1] and R[0 :nR-1]  for i = 0 to nL -1  L[i] = A[p+i]  for j = 0 to nR -1  R[j] = A[q+j+1] |
| As long as each of the arrays L and R contains an unmerged element,  copy the smallest unmerged element back into A[p:c] | |
| for k in range(p,r):  if L[i] <= R[j]:  A[k] = L[i]; i += 1  else:  A[k] = R[j]; j += 1 | i = 0; j = 0; k = 0  while I < nL and j < nR  if L[i] ≤ R[j]  A[k] = L[i]  i = i +1  else A[k] = L[i]  j = j +1  k = k +1 |

|  |  |
| --- | --- |
| Having gone through one of L and R entirely,  copy the remainder of the other to the end of A[p:r]. | |
|  | while I < nL  A[k] = L[i]  i = i +1  k = k +1  while j < nR  A[k] = L[j]  j = j +1  k = k +1 |