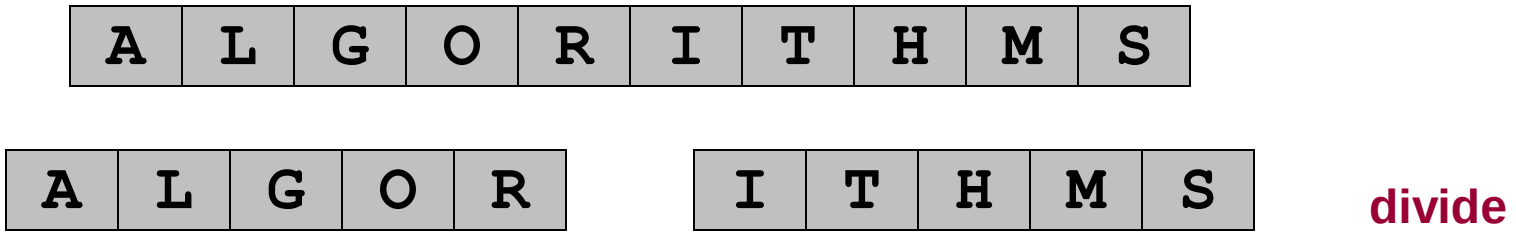
Mergesort

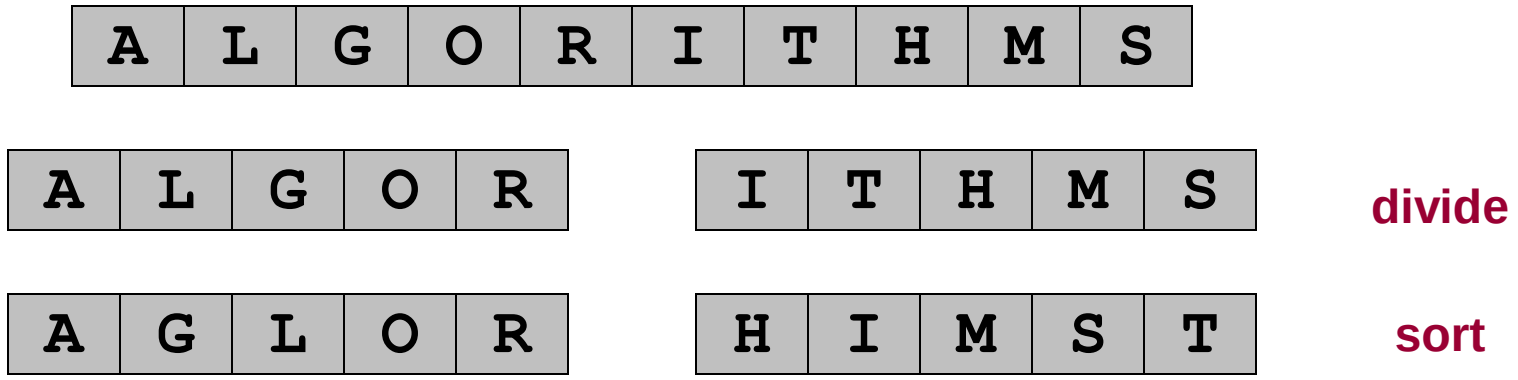
Mergesort

- Mergesort (divide-and-conquer)
  - Divide array into two halves.



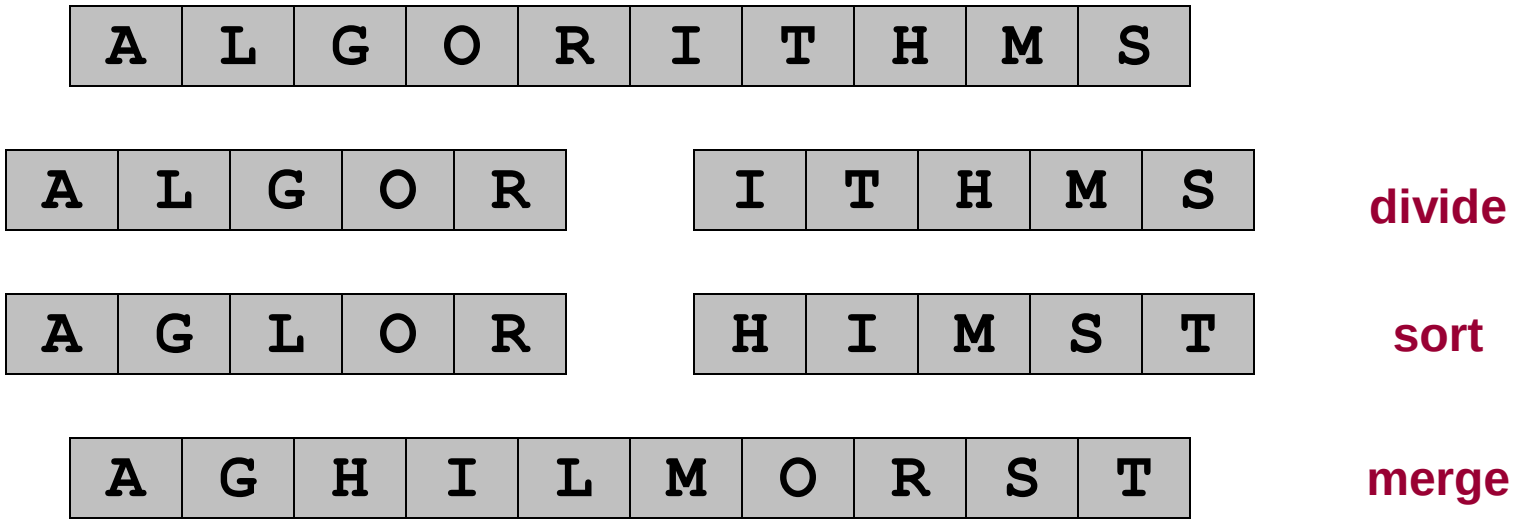
Mergesort

- Mergesort (divide-and-conquer)
  - Divide array into two halves.
  - Recursively sort each half.



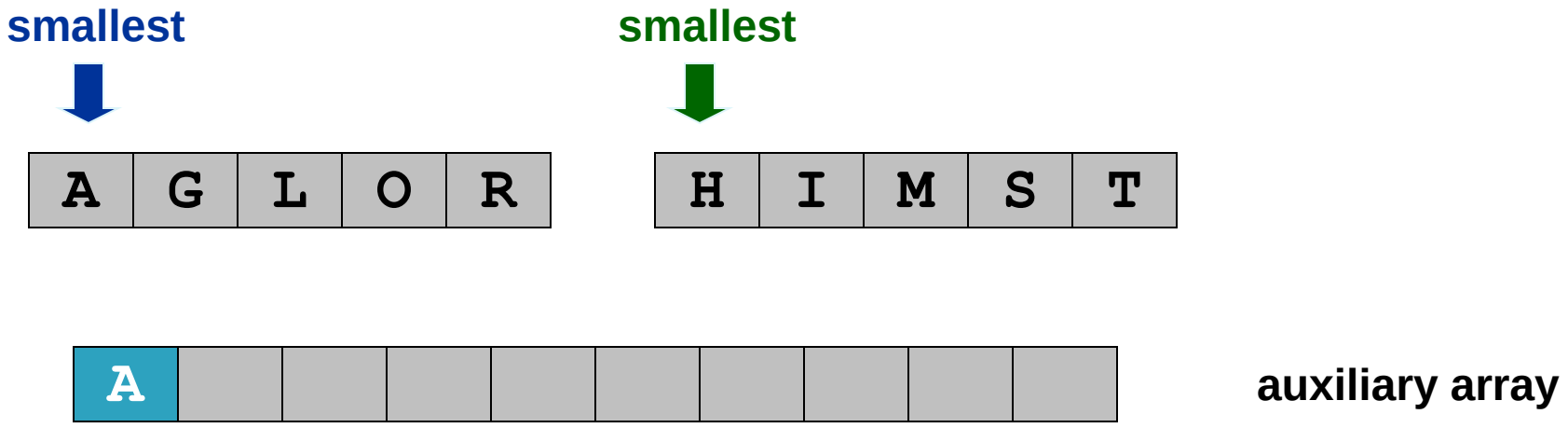
Mergesort

- Mergesort (divide-and-conquer)
  - Divide array into two halves.
  - Recursively sort each half.
  - Merge two halves to make sorted whole.



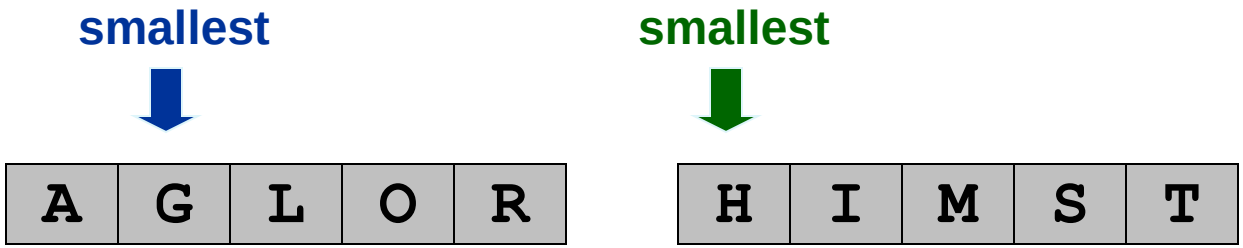
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



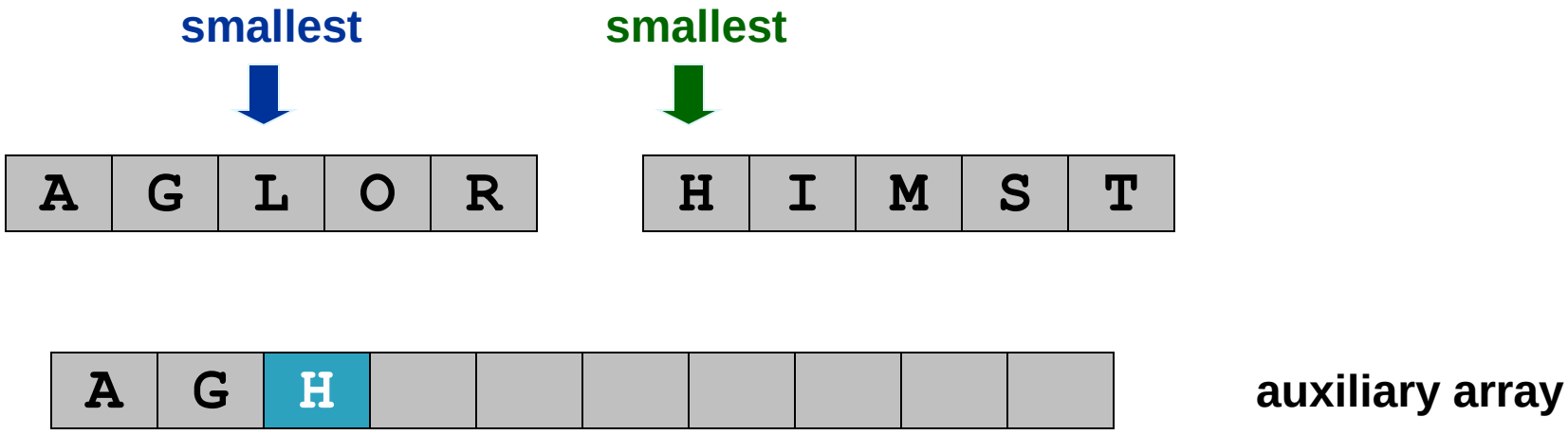
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



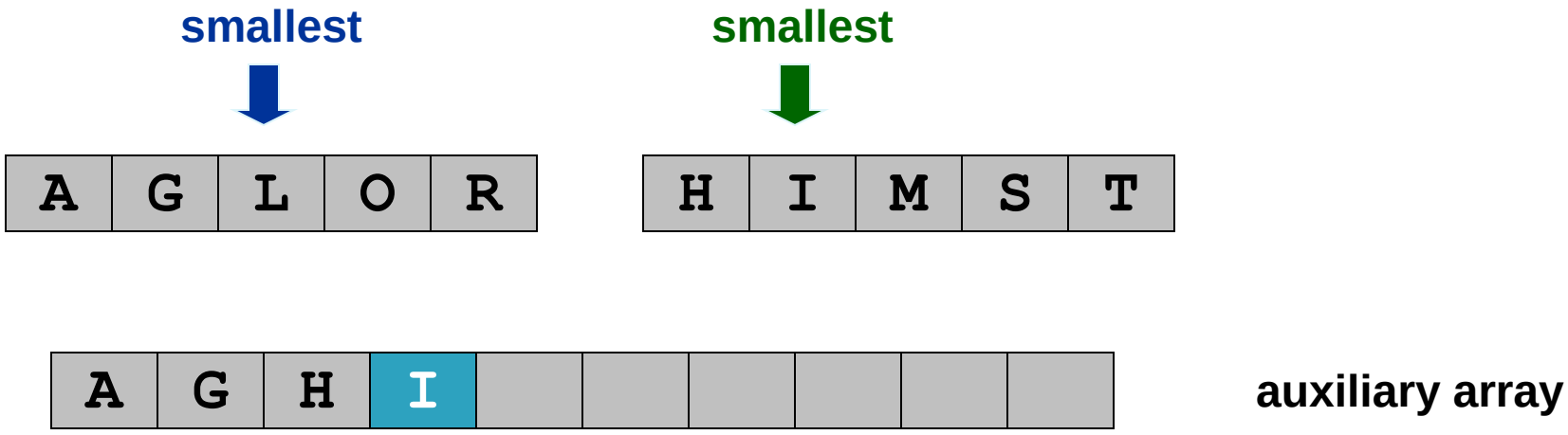
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



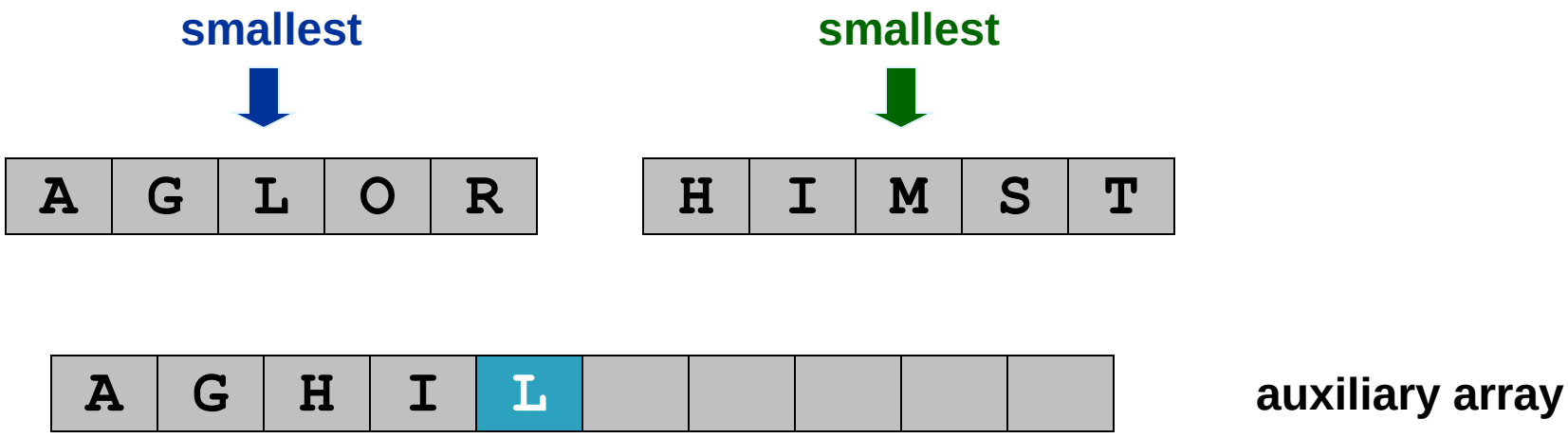
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



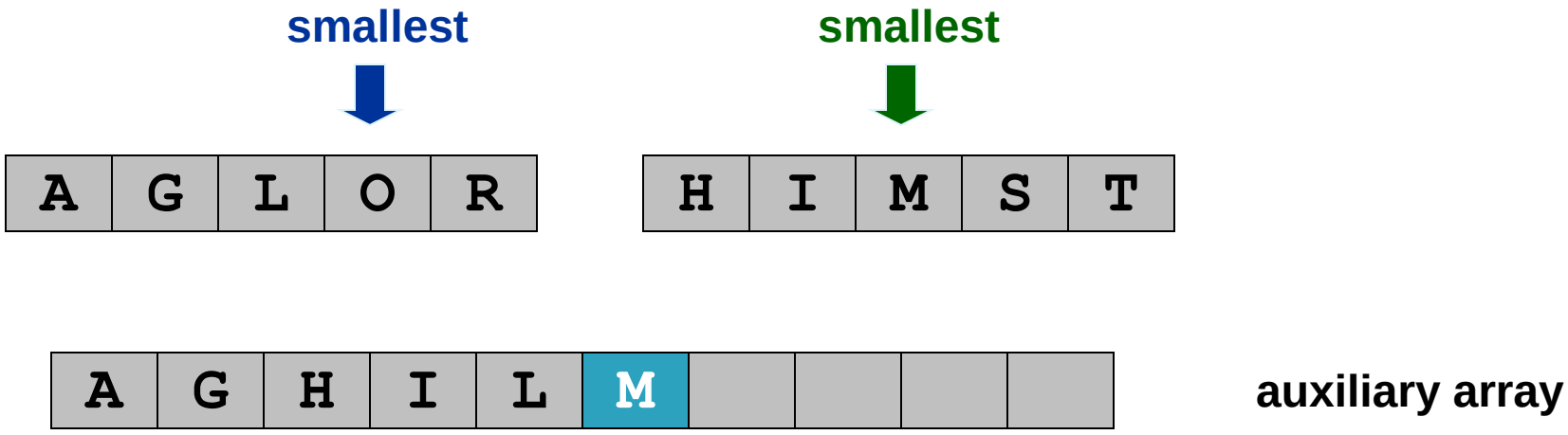
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



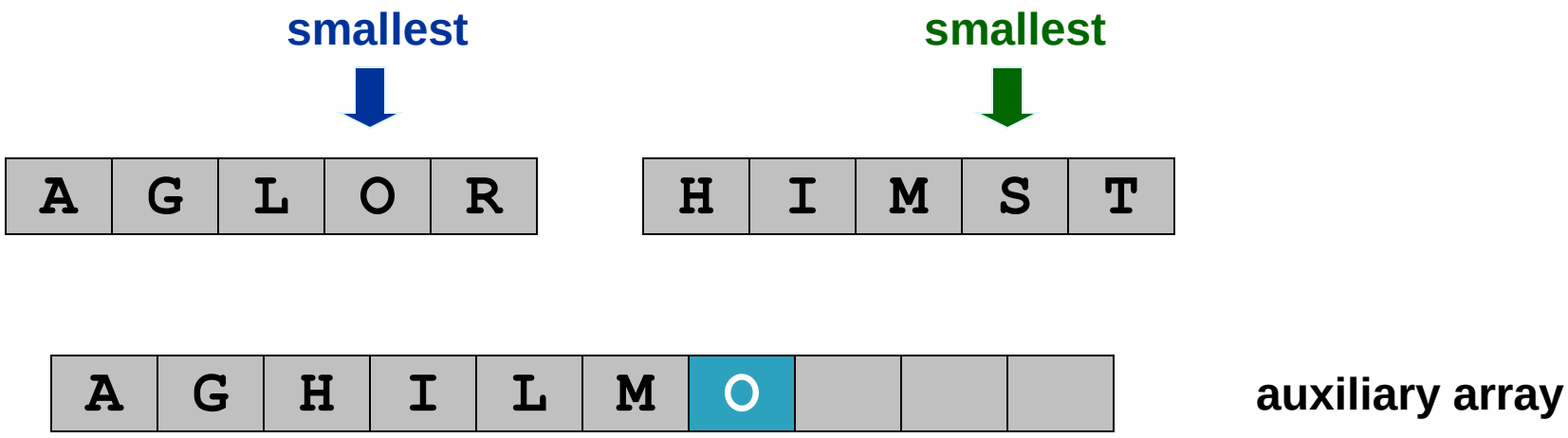
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



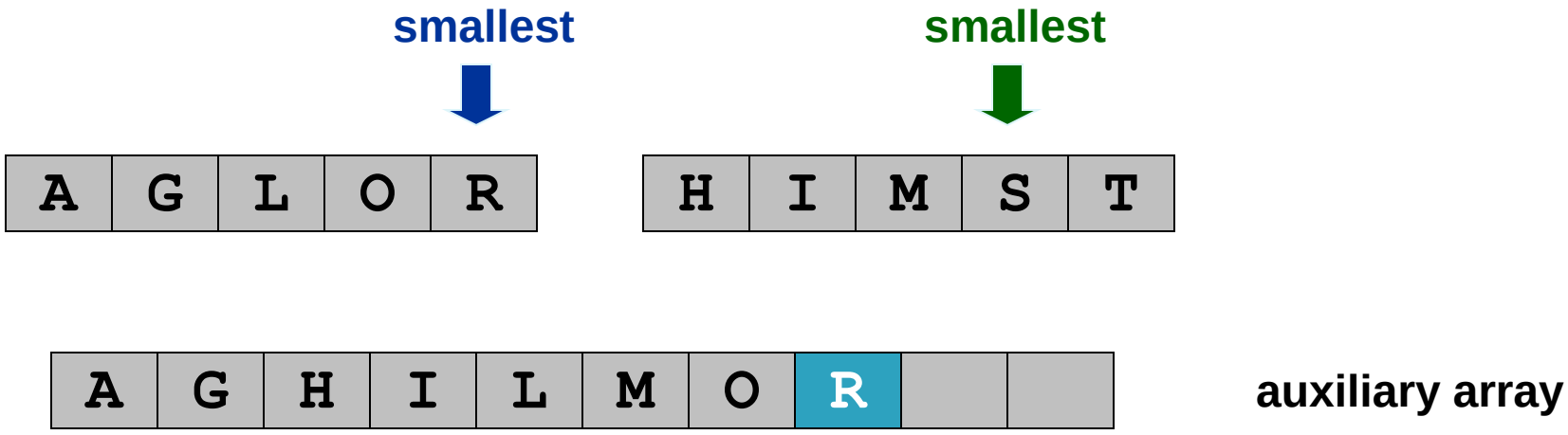
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



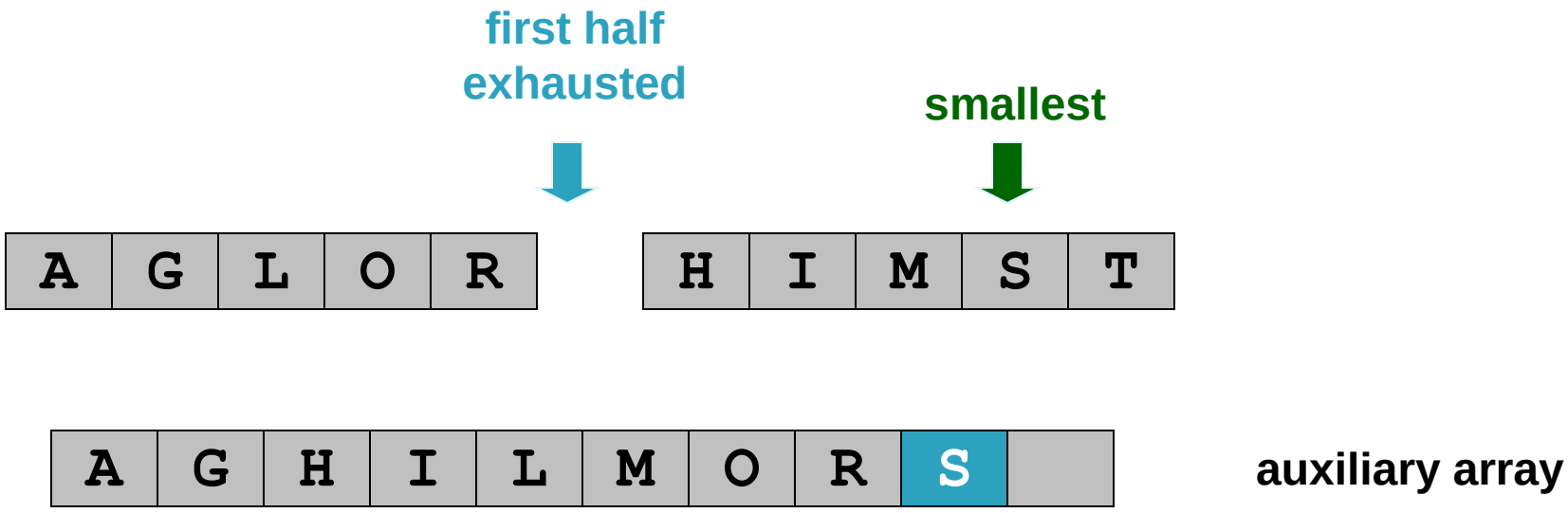
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



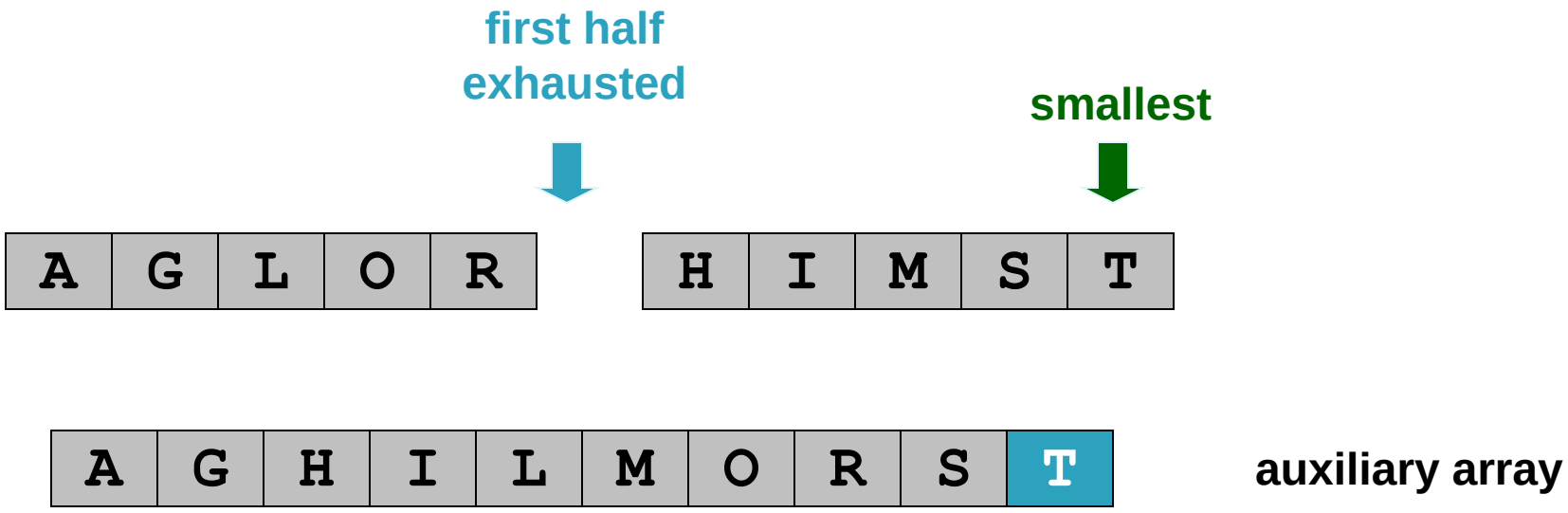
Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



Merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



Implementing Mergesort

mergesort (see Sedgewick Program 8.3)

```
Item aux[MAXN];

void mergesort(Item a[], int left, int right) {
   int mid = (right + left) / 2;
   if (right <= left)
      return;
   mergesort(a, left, mid);
   mergesort(a, mid + 1, right);
   merge(a, left, mid, right);
}
```

Implementing Merge (Idea 0)

```
mergeAB(Item c[], Item a[], int N, Item b[], int M )
  { int i, j, k;
    for (i = 0, j = 0, k = 0; k < N+M; k++)
      {
        if (i == N) { c[k] = b[j++]; continue; }
        if (j == M) { c[k] = a[i++]; continue; }
        c[k] = (less(a[i], b[j])) ? a[i++] : b[j++];
      }
  }
```

Implementing Mergesort

merge (see Sedgewick Program 8.2)

```
void merge(Item a[], int left, int mid, int right) {
  int i, j, k;

  for (i = mid+1; i > left; i--)
     aux[i-1] = a[i-1];
  for (j = mid; j < right; j++)
     aux[right+mid-j] = a[j+1];

  for (k = left; k <= right; k++)
     if (ITEMless(aux[i], aux[j]))
        a[k] = aux[i++];
     else
        a[k] = aux[j--];
}
```

copy to temporary array

merge two sorted sequences

Mergesort Demo

- Mergesort The auxilliary array used in the merging operation is shown to the right of the array a[], going from (N+1, 1) to (2N, 2N).
- The demo is a dynamic representation of the algorithm in action, sorting an array a containing a permutation of the integers 1 through N. For each i, the array element a[i] is depicted as a black dot plotted at position (i, a[i]). Thus, the end result of each sort is a diagonal of black dots going from (1, 1) at the bottom left to (N, N) at the top right. Each time an element is moved, a green dot is left at its old position. Thus the moving black dots give a dynamic representation of the progress of the sort and the green dots give a history of the data-movement cost.