Part 1.1: Tracing Sorting Algorithms by Hand

Input arrays:

- random = [C, D, Y, X, A, Z, B];
- sorted = [A, B, C, D, X, Y, Z];
- reversed = [Z, Y, X, D, C, B, A].

Insertion Sort

pseudocode:

```
INSERTION-SORT(A)
for j=2 to A.length
    key=A[j]
    i=j-1
    while i>0 and A[i]<key
        A[i+1]=A[i]
        i-=1
    A[i+1]=key
```

ramdom case

red for insertion element, blue for already sorted sub-array, green for unsorted sub-array.

iteration	array
iteration 0	[C, D, Y, X, A, Z, B]
iteration 1	[C, D, Y, X, A, Z, B]
iteration 2	[C, D, Y, X, A, Z, B]
iteration 3	[C, D, X, Y, A, Z, B]
iteration 4	[A, C, D, X, Y, Z, B]
iteration 5	[A, C, D, X, Y, Z, B]

iteration	array
iteration 6	[A, B, C, D, X, Y, Z,]

sorted case

red for insertion element, blue for already sorted sub-array, green for unsorted sub-array.

iteration	array
iteration 0	[A, B, C, D, X, Y, Z]
iteration 1	[A, B, C, D, X, Y, Z]
iteration 2	[A, B, C, D, X, Y, Z]
iteration 3	[A, B, C, D, X, Y, Z]
iteration 4	[A, B, C, D, X, Y, Z]
iteration 5	[A, B, C, D, X, Y, Z]
iteration 6	[A, B, C, D, X, Y, Z]

reversed case

red for insertion element, blue for already sorted sub-array, green for unsorted sub-array.

iteration	array
iteration 0	[Z, Y, X, D, C, B, A]
iteration 1	[Y, Z, X, D, C, B, A]
iteration 2	[X, Y, Z, D, C, B, A]
iteration 3	[D, X, Y, Z, C, B, A]
iteration 4	[C, D, X, Y, Z, B, A]
iteration 5	[B, C, D, X, Y, Z, A]
iteration 6	[A, B, C, D, X, Y, Z]

Merge Sort

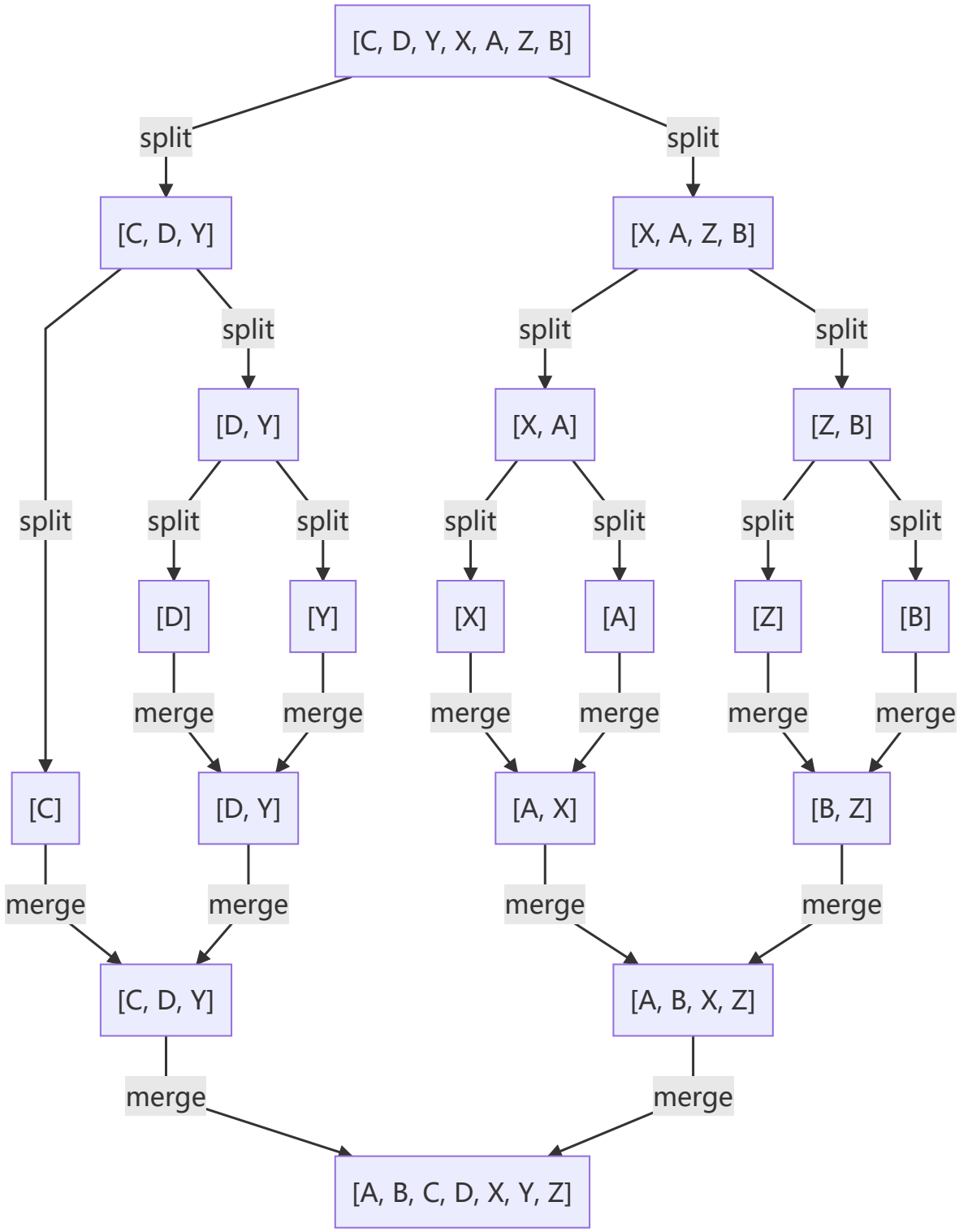
pseudocode:

```
1  MergeSort(A,begin,end) // [begin,end)
```

```
if end-begin==1
    return A
mid=begin+(end-begin)/2
MergeSort(A,begin,mid)
MergeSort(A,mid,end)
Merge(A,begin,end,mid)

Merge(A,begin,end,mid) // [begin,mid) and [mid,end)
n1=mid-begin
n2=end-mid
let L[1…n1] and R[1…n2] be a new array
for i=1 to n1
    L[i]=A[begin+i]
for i=1 to n2
    R[i]=A[mid+i]
i=1
j=1
for k=begin to end
    if L[i]<=R[j]
        A[k]=L[i]
        i+=1
    else
        A[k]=R[j]
        j+=1
```

ramdom case

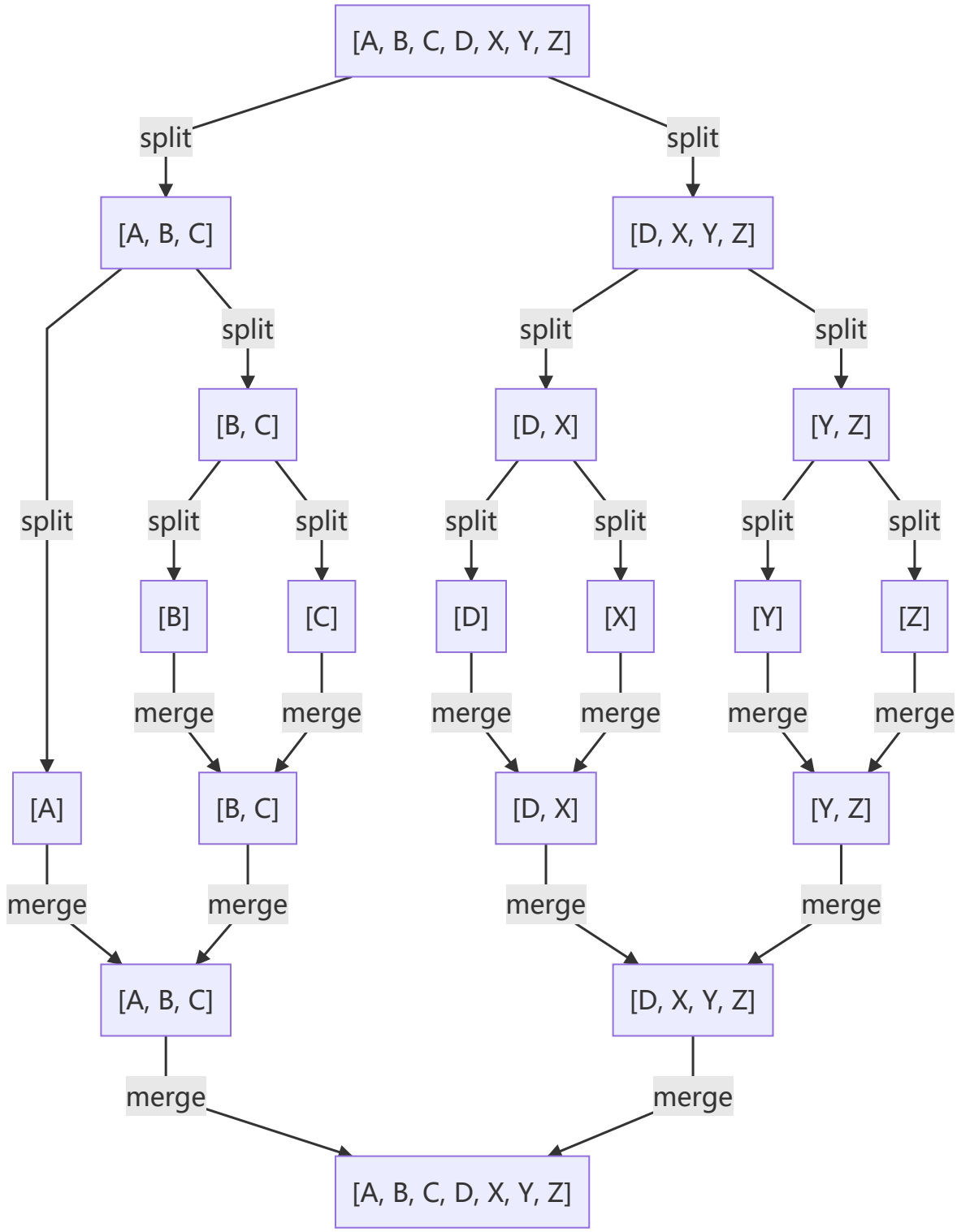


merge [C, D, Y] and [A, B, X, Z]

iteration	i	j	merged_array
0	0	0	[]
1	0	1	[A]
2	0	2	[A, B]
3	1	2	[A, B, C]

iteration	i	j	merged_array
4	2	2	[A, B, C, D]
5	2	3	[A, B, C, D, X]
6	3	3	[A, B, C, D, X, Y]
7	3	4	[A, B, C, D, X, Y, Z]

sorted case

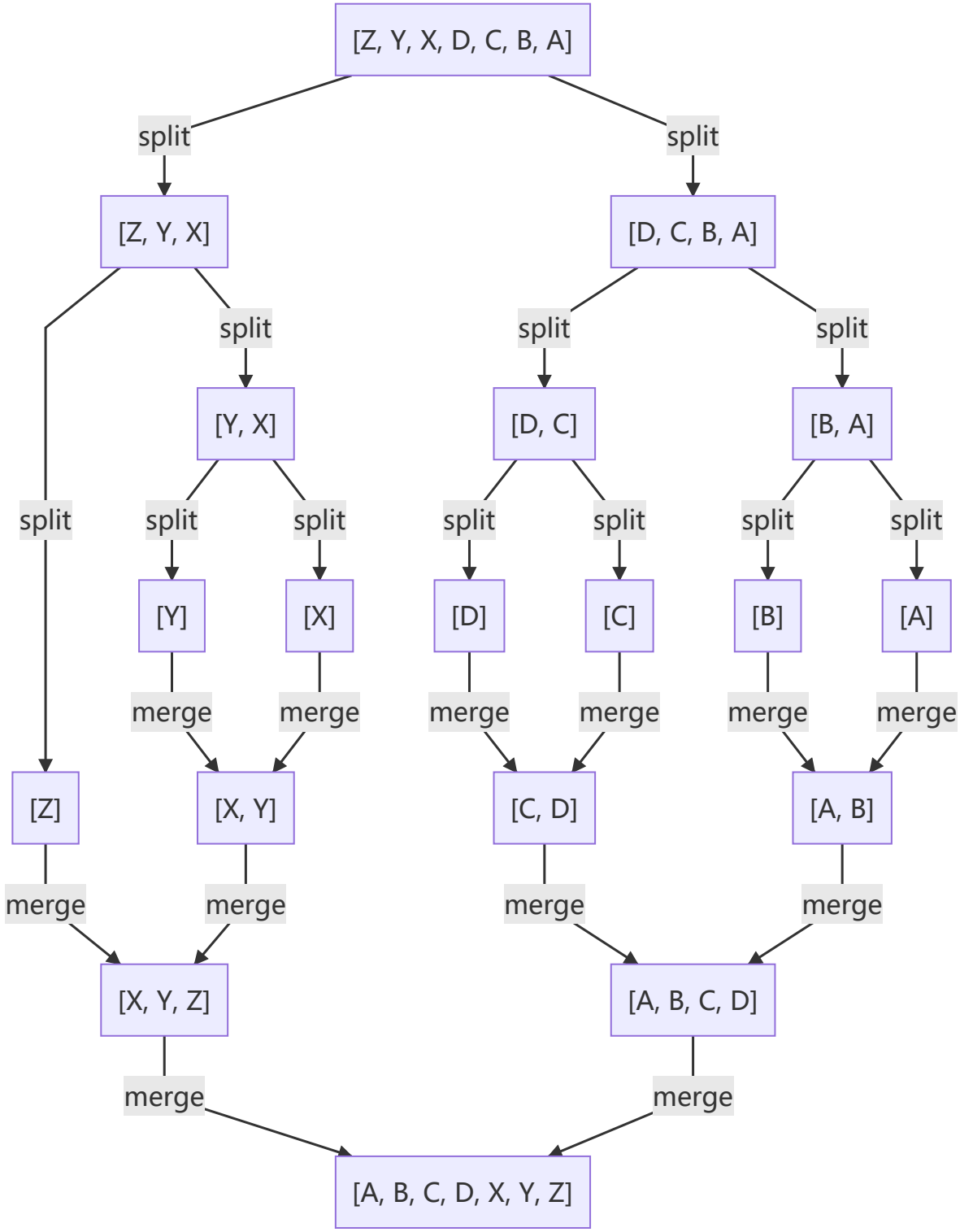


merge [D, X] and [Y, Z]

iteration	i	j	merged_array
0	0	0	[]
1	1	0	[D]
2	2	0	[D, X]
3	2	1	[D, X, Y]

iteration	i	j	merged_array
4	2	2	[D, X, Y, Z]

reversed case



merge [C, D] and [A, B]

iteration	i	j	merged_array
0	0	0	[]
1	0	1	[A]
2	0	2	[A, B]
3	1	2	[A, B, C]
4	2	2	[A, B, C, D]

TimSort

code:

```
public void TimSort(int[] arr, int start, int end, int param) {
        int i=0;
        while(i<end-start){
            if(i+param<=end){
                InsertionSort(arr, i, i+param);
            } else {
                InsertionSort(arr, i, end);
            }
            i+=param;
        }
        i=param;
        while(i<end-start){
            if(i+param<=end){
                merge(arr, start, i, i+param);
            } else {
                merge(arr, start, i, end);
            }
            i+=param;
        }
    }

```

random case

different colors for different run, blue for already merge

run size=2

iteration	array
run 1	[C, D, Y, X, A, Z, B]
run 2	[C, D, Y, X, A, Z, B]
run 3	[C, D, Y, X, A, Z, B]

iteration	array
run 4	[C, D, Y, X, A, Z, B]
merge 1	[C, D, X, Y, A, Z, B]
merge 2	[A, C, D, X, Y, Z, B]
merge 3	[A, B, C, D, X, Y, Z]

sorted case

different colors for different run, blue for already merge

run size=2

iteration	array
run 1	[A, B, C, D, X, Y, Z]
run 2	[A, B, C, D, X, Y, Z]
run 3	[A, B, C, D, X, Y, Z]
run 4	[A, B, C, D, X, Y, Z]
merge 1	[A, B, C, D, X, Y, Z]
merge 2	[A, C, D, X, Y, Z, B]
merge 3	[A, B, C, D, X, Y, Z]

reversed case

different colors for different run, blue for already merge

run size=2

iteration	array
run 1	[Z, Y, X, D, C, B, A]
run 2	[Z, Y, X, D, C, B, A]
run 3	[Z, Y, X, D, C, B, A]
run 4	[Z, Y, X, D, C, B, A]
merge 1	[D, X, Y, Z, C, B, A]

iteration	array
merge 2	[B, C, D, X, Y, Z, A]
merge 3	[A, B, C, D, X, Y, Z]