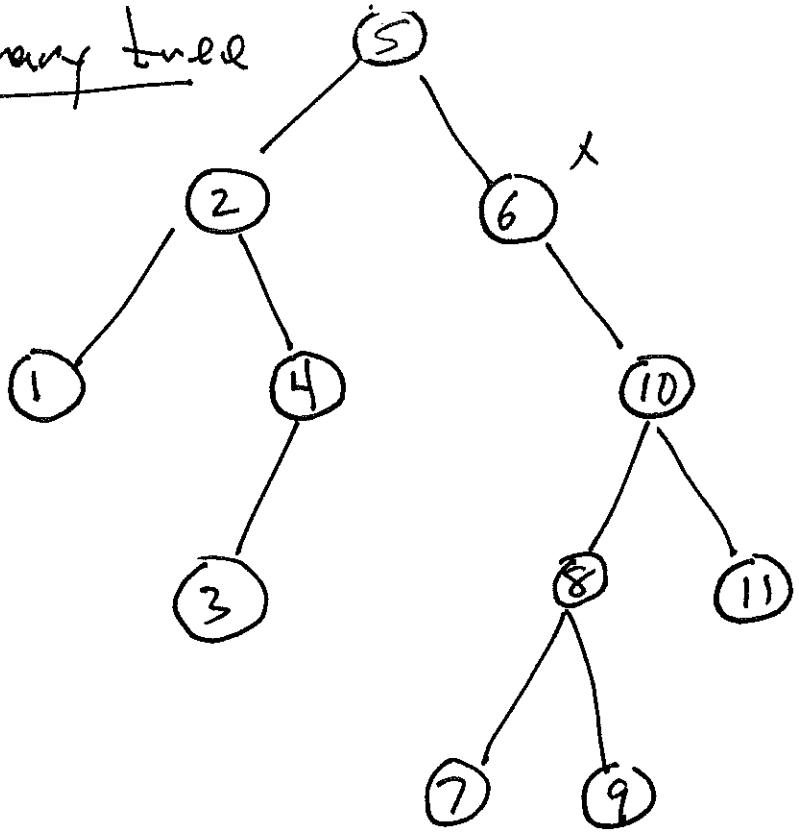
CMPS 12B 2-25-11

Prior-in-Post order tree-walks:

Ex. a Binary tree



BST Property:

Values in left subtree of x ≤ Value at x

Value at x ≤ Values in right subtree of x

- In order tree walk:
  - left subtree
  - this node
  - right subtree

Ex 1 2 3 4 5 6 7 8 9 10 11

- Pre order tree-walk
  - this node
  - left subtree
  - right subtree

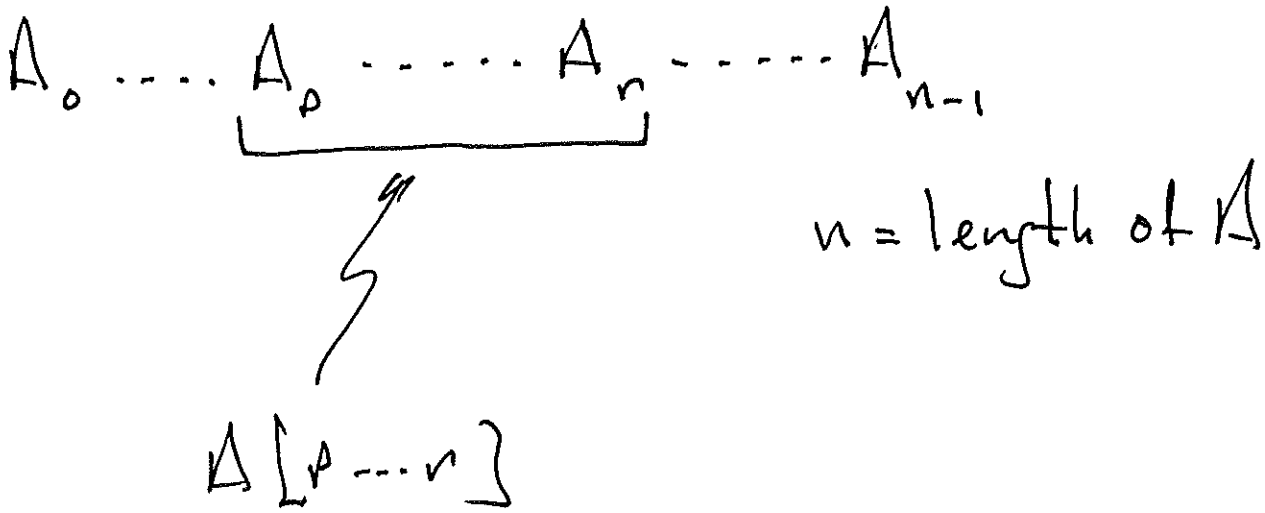
Ex. 5 2 1 4 3 6 10 8 7 9 11

- Post order tree walk
  - left subtree
  - right subtree
  - this node

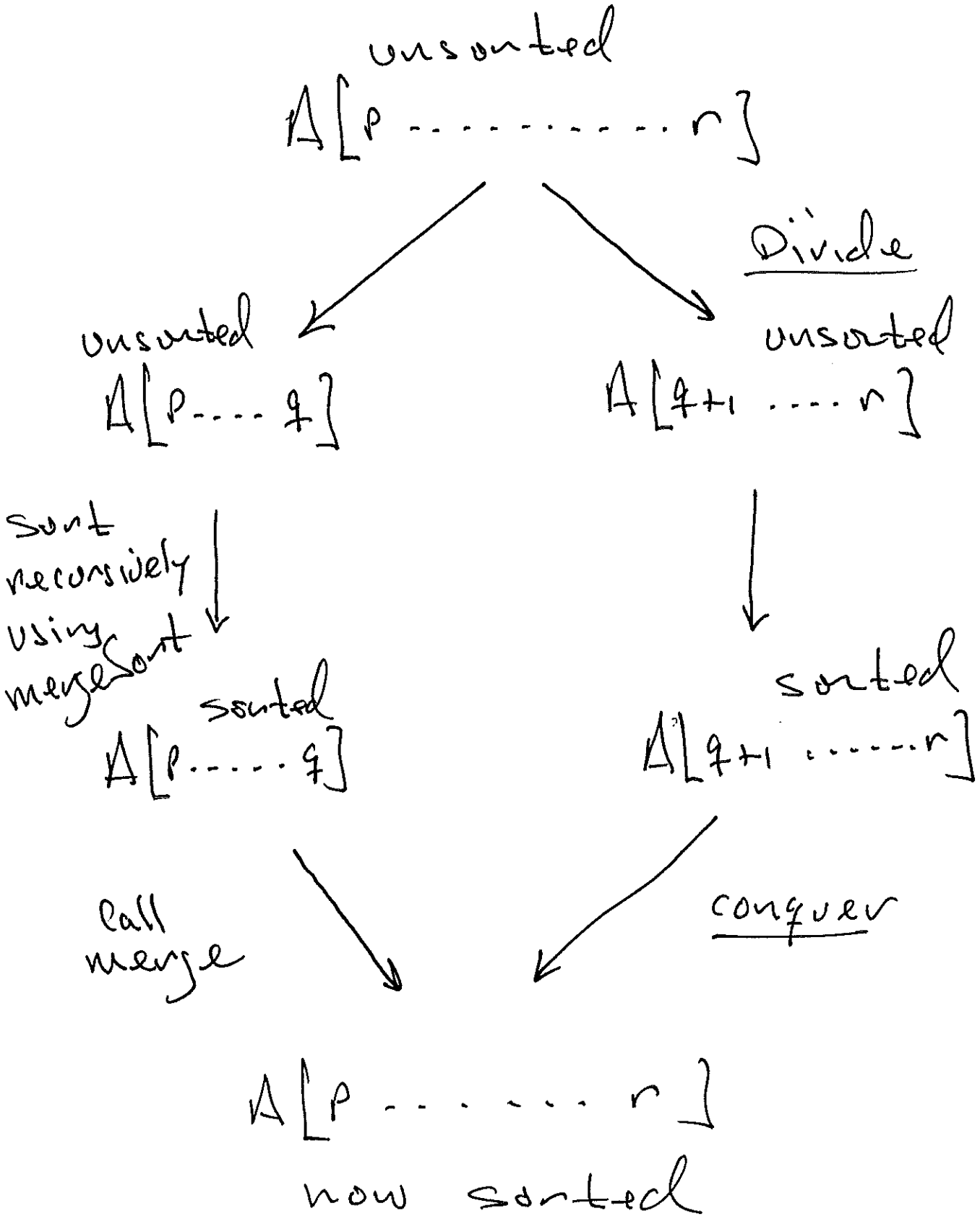
Ex. 1 3 4 2 7 9 8 11 10 6 5

MergeSort: sorts an array of integers in increasing order.

Notation $A[p \ldots r]$ subarray of A from index p to r

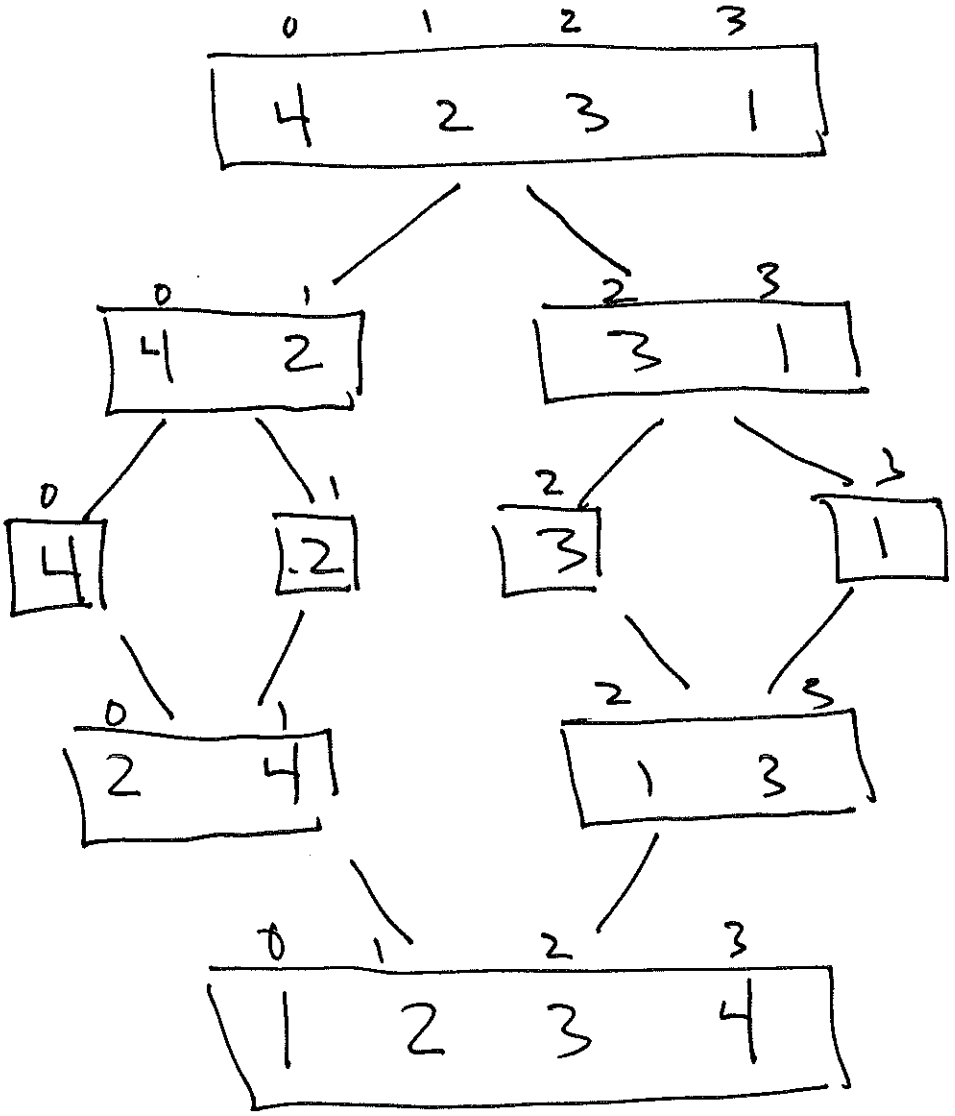


Start with

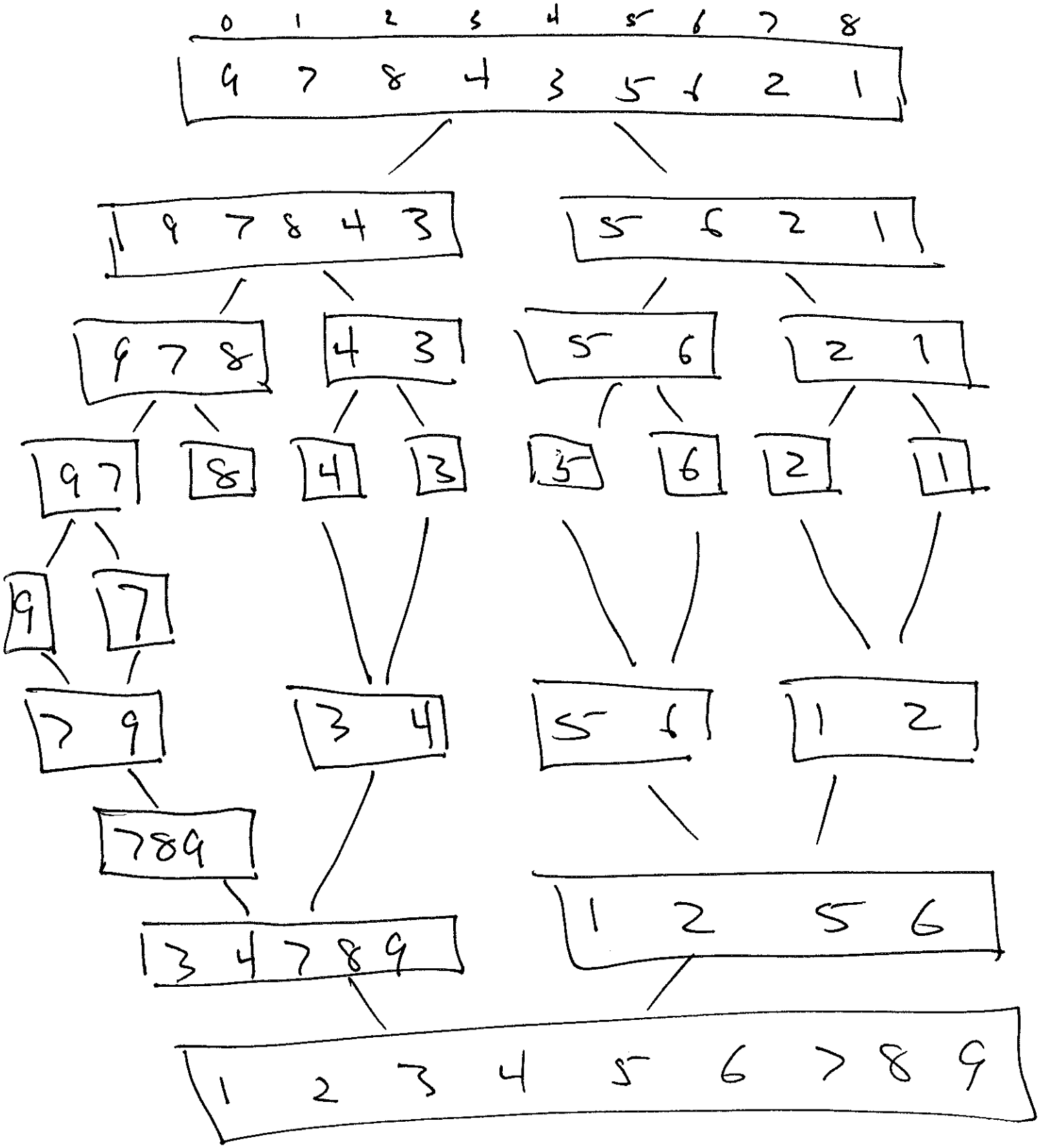

unsorted
A[p ..... r]
Divide
unsorted
A[p .... q]
unsorted
A[q+1 .... r]
sort recursively using mergesort
sorted
A[p .... q]
sorted
A[q+1 ..... r]
call merge
conquer
A[p ..... r]
now sorted

```
Public static void mergeSort(int[]A, int p, int r)
{
    if (p < r)
    {
        int q = (p+r)/2;
        mergeSort(A, p, q);
        mergeSort(A, q+1, r);
        merge (A, p, q, r);
    }
}
```

Trace: $n = 4$



Trace: n = 9


0 1 2 3 4 5 6 7 8
9 7 8 4 3 5 6 2 1
9 7 8 4 3
5 6 2 1
9 7 8
4 3
5 6
2 1
9 7
8
4
3
5
6
2
1
9
7
7 9
3 4
5 6
1 2
7 8 9
1 2 5 6
1 2 3 4 5 6 7 8 9

How to merge ?

- let n_1 = length $A[p \ldots q] = q - p + 1$
- let n_2 = length $A[q+1 \ldots r] = r - q$
- copy $A[p \ldots q]$ into $L[0 \ldots (n_1 - 1)]$
- copy $A[q+1 \ldots r]$ into $R[0 \ldots (n_2 - 1)]$
- Place L & R back into

$A[p \ldots r]$

- to do this:

Compare L_i to R_j

smaller goes into A_k