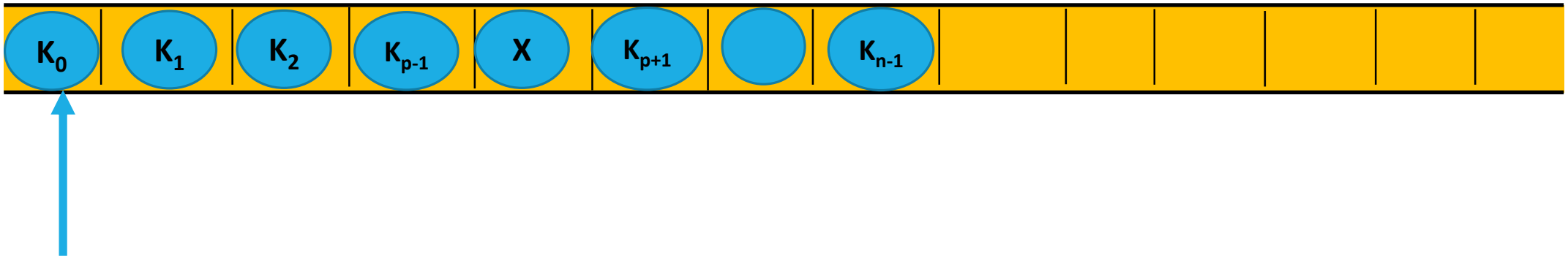
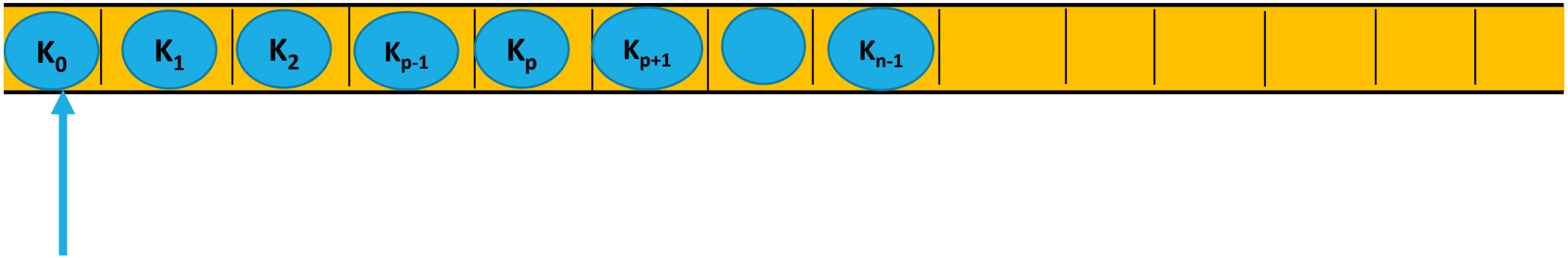
2.4 顺序表查找

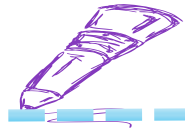
$(k_0, k_1, \cdots, k_{p-1}, X, k_{p+1} \cdots, k_{n-1})$





顺序表查找算法

算法2-9

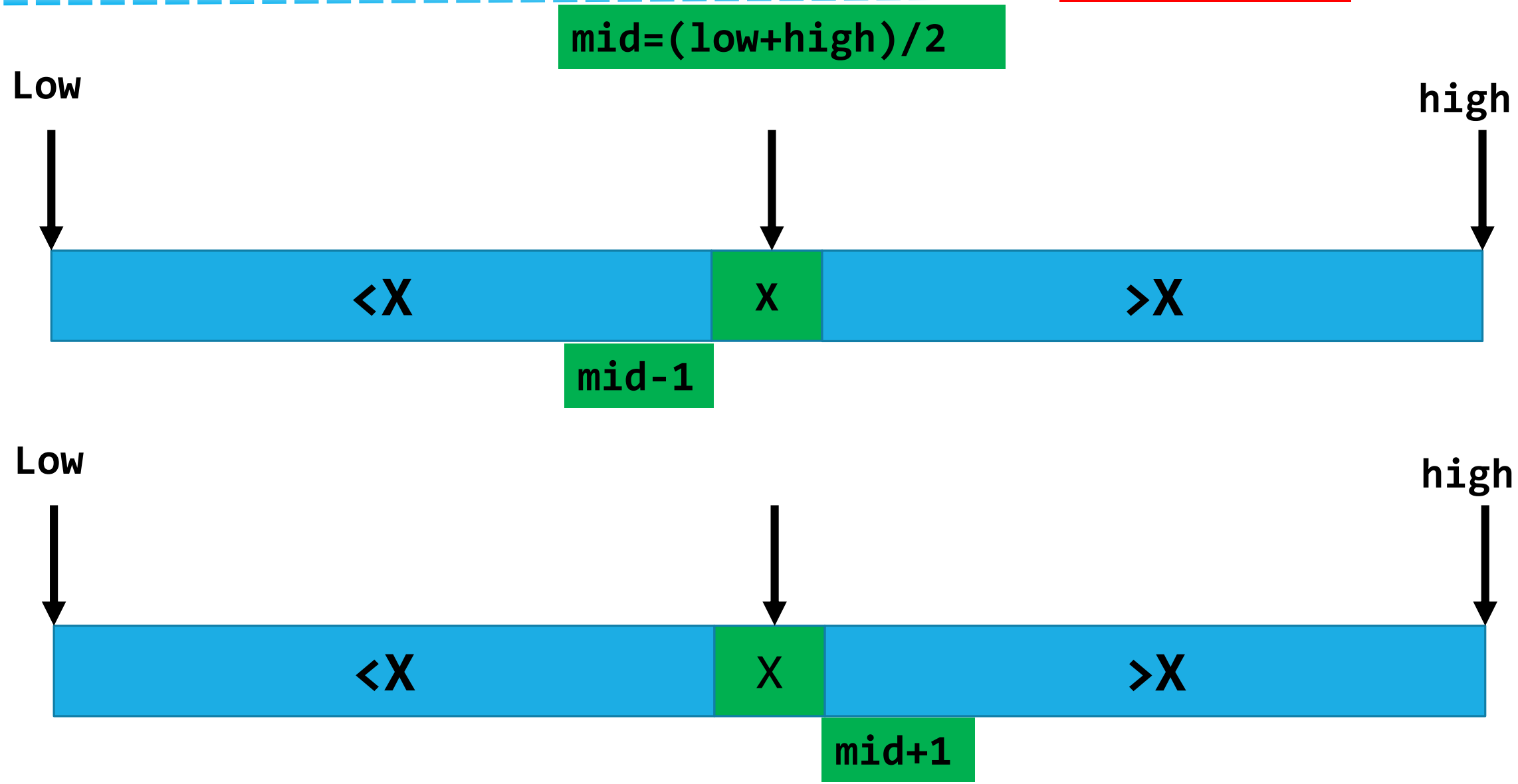


```
1  int LocateIndex_seq(SeqList slist, int x)//查找值为x的元素，返回元素所在下标
2  {
3          int q;
4          for(q=0;q<slist->n;q++){
5                  if (slist->elem[q] == x)//查找成功，返回对应的下标
6                          return q;
7          }
8          return -1;//查找失败，返回-1
9  }
```

算法时间复杂度O（n）

二分查找

减而治之



2.4.2 二分查找算法

查找30的过程

	low						mid						high
Step1	05	10	25	27	30	35	**45**	49	50	52	55	60	70
						因为 45>30，进入到前半区间							

	low		mid			high							
Step2	05	10	**25**	27	30	35	45	49	50	52	55	60	70
		因为 25<30，进入后半区间											

				low	mid	high							
Step3	05	10	25	27	**30**	35	45	49	50	52	55	60	70
				30==30，查找成功									

2.4.2 二分查找算法

查找48的过程

	low						mid						high
Step1	05	10	25	27	30	35	**45**	49	50	52	55	60	70
						因为 45<48，进入后半区间							

								low		mid			high
Step2	05	10	25	27	30	35	45	49	50	**52**	55	60	70
									因为 52>48，进入前半区间				

							low=mid high						
Step3	05	10	25	27	30	35	45	**49**	50	52	55	60	70
							因为 49>48，进入前半区间						
							high Low=mid						
Step4	05	10	25	27	30	35	45	49	50	52	55	60	70
							因为 low>high，查找失败						

顺序表查找算法

二分查找（检索）
算法2-10

```
int binsearch(SeqList slist, int key, int *pos)
{
        int index = 1;   int mid;
        int low = 0;   int high = slist->n - 1;
        while (low <= high)
        {
                mid = (low + high) / 2;
                if (slist->elem[mid] == key) {
                        *pos = mid;
                        printf("找到,共进行%d次比较\n", index);
                        printf("要找的数据%d在位置%d上\n", key, mid);
                        return 1;
                }
                else if (slist->elem[mid] > key)
                        high = mid - 1;
                else    low = mid + 1;
                index++;
        }
        *pos = low;
        printf("没有找到,共进行%d次比较\n", index-1);
        printf("可将此数插入到位置%d上\n", *pos);
        return -1;
}
```