Sorting atomic items

Chapter 5

Distribution based sorting paradigms

The distribution-based sorting

```
QuickSort (S ,i, j)
If (i<j) {
        r = pick the position of a "good pivot"
        Swap S[r] with S[i];
        p = Partition (S, i, j);
        QuickSort (S, i, p-1);
        QuickSort (S, p+1, j)
}
```

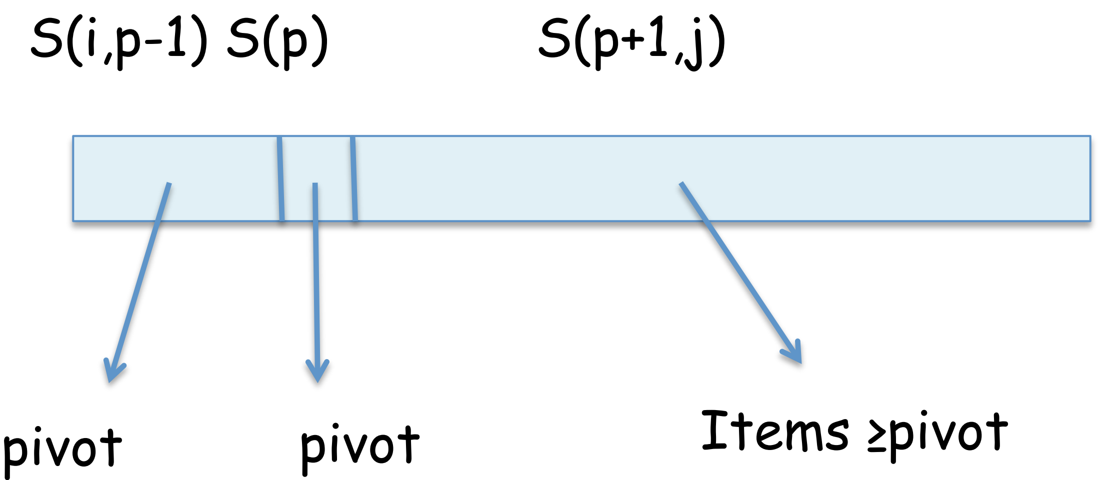
Based on Divide&Conquer. Combine step is not present.

Divide step : Procedure Partition

QuickSort is in place alg.

The distribution-based sorting

Partition divides the array in 3 parts:



Partition takes O(n)

If the two sub-arrays are balanced at each level of the recursion

$T(n)=2T(n/2)+O(n) = O(n \log n)$ as MergeSort

To study the worst case, we look at the position of q that maximize the time

$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + O(n)$ where q range from 0 to n-1

The distribution-based sorting

Guess: $T(n) \leq cn^2$

$$T(n) = \max_{0 \leq q \leq n-1} (cq^2 + c(n-q-1)^2) + O(n) = c \max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) + O(n)$$

Gives the maximum when $q=0$ or $q=n-1$:

$$(q^2 + (n-q-1)^2) \leq (n-1)^2 = n^2-2n-1$$

$T(n) \leq c(n^2-2n-1) + O(n) \leq cn^2$ worst case

QuickSort
Expected running time

- Sequence S(1,n); Rank Z(1, n) : Zi is the i-th smallest element;
- $p_{i,j}$ is the probability that a comparison Zi : Zj occurs during an execution of QiuickSort;
- The expected total number is: $E = \sum_{i=1}^{n} \sum_{j>i} p_{i,j}$

Remarks:

1. In Partition two items are compared if one of them is a pivot.
2. If two items go in different sub-arrays they will never be compared in the future.

Expected running time

If j=i+1 the elements are compared for sure: there not exist an element that, being the pivot, can put them in separate sub-arrays as pivot. $p_{i,i+1}=1$

If j>i+1 consider the set of elements A = {Z_i, Z_{i+1},..., Z_j) if as pivot is selected an element not in A all elements remain in the same partition and Zi and Zj are not compared.

if Zi or Zj are selected as pivot, Zi and Zj are compared

If Zk is selected with k ≠i,j A is split into 2 sub-arrays and Zi and Zj are not compared.

So $p_{i,j}=2/(j-i+1)$ (when j=i+1 $p_{i,j}=1$)

QuickSort
Expected running time

The expected total number is:

$$E = \sum_{i=1}^{n} \sum_{j>i} p_{i,j} = \sum_{i=1}^{n} \sum_{j>i} p_{i,j} = 2/(j-i+1) = \sum_{i=1}^{n} \sum_{k=1}^{n-i} 2/(k+1) \leq$$

$$2 \sum_{i=1}^{n} \sum_{k=1}^{n} 1/k$$

since $\sum_{k=1}^{n} 1/k = \ln n + O(1)$ hence

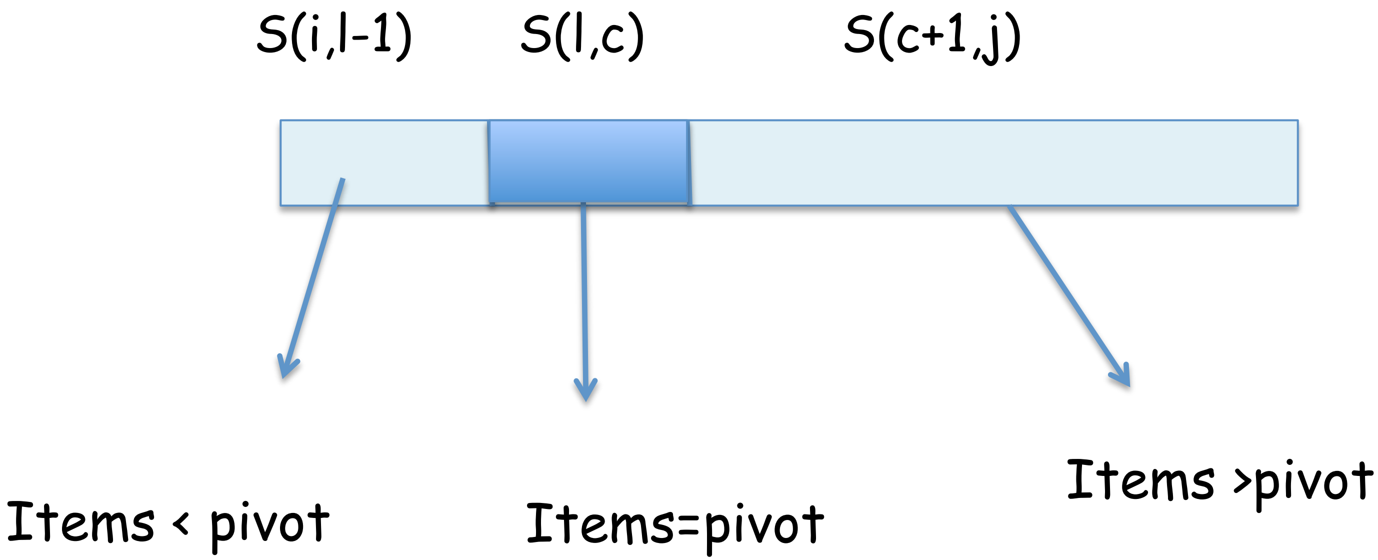
$E = O(n\log n)$

In average quicksort takes at most 1.45 nlogn operations

3-ways Partition

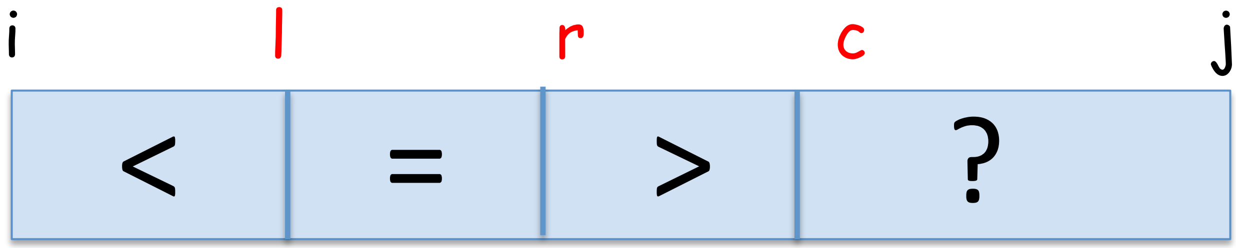
In procedure Partition of QuickSort elements equal to the pivot are arbitrarily distributed among the 2 partitions.

In 3-ways Partition we have:



3-ways Partition takes O(n). The central part can be discarded in the Recursion.

3-ways partition



Variable l indicates the last item < than the pivot

Variable r indicates the fist item > to the pivot

Variable c indicates the nextitem to be considered.

If S[c]>pivot c=c+1

If S[c]=pivot exchange S[c] and S[r] ; c=c+1; r=r+1;

If S[c] <pivot l=l+1: exchange S[c] and S[l]; S[c] and S[r]; r=r+1; c=c+1

3-ways partition

l r c

5 2 9 12 12 12 20 18 13 15 17 19 12 8

l r c

5 2 9 12 12 12 20 18 13 15 17 19 12 8 no exchange

l r c

5 2 9 12 12 12 12 18 13 15 17 18 20 8 1 exchange 12:20

5 2 9 8 12 12 12 18 13 15 17 18 20 12 2 exchanges 8:12 and

5 2 9 8 12 12 12 18 13 15 17 19 20 12 and 12:18

5 2 9 8 12 12 12 12 13 15 17 18 20 18

l r c

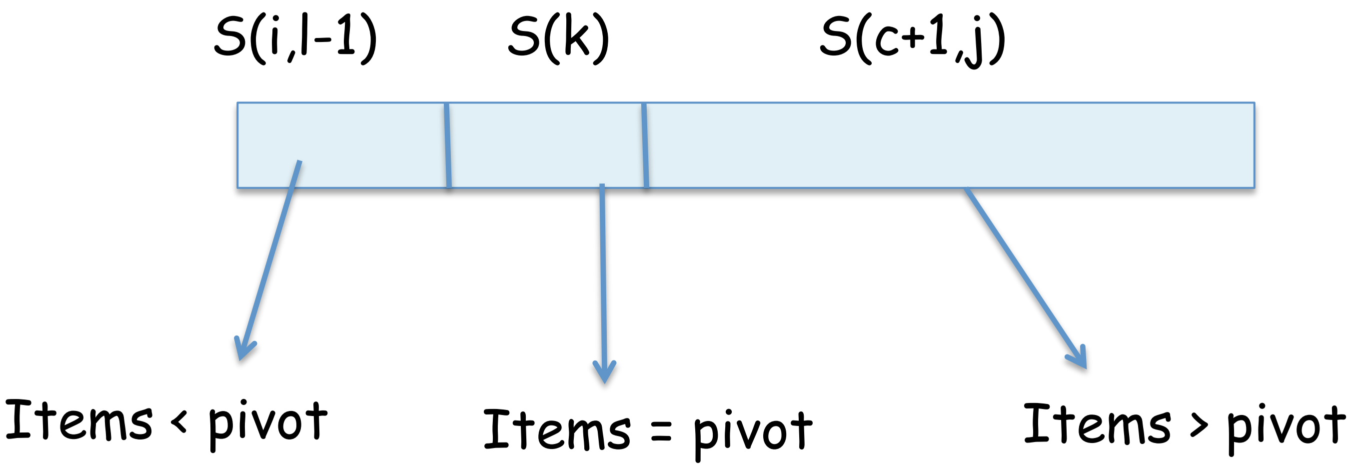
3-ways Partition(S, i,j)

```
P = S[i]; l = i; r = i + 1;
for (c = r; c ≤ j; c++) do
        if (S[c] == P) then
                swap S[c] with S[r];
                r++;
        else if (S[c] < P) then
                swap S[c] with S[l];
                swap S[c] with S[r];
                r++; l++;
        end if
endfor
return < l, r-1 >
```

Recursion on S(i, l-1) and S(r, j)

Modify QuickSort to select the k-th item

Idea: select an item at random S[r] and call Partition.
Let k the position of the pivot.



If k is in the range of the items equal to the pivot return : S[r] is the k-th item.

If k is in the range the items less than the pivot: Recurse on S(i,l-1) and k.

If k is in the range the items greater than the pivot: Recurse on S(c+1, j) and k-c.

RandSelect

Algorithm 5.5 Selecting the k-th ranked item: $\textsc{RandSelect}(S, k)$

r = pick a random item from S;

$S_< =$ items of S which are smaller than $S[r]$;

$S_> =$ items of S which are larger than $S[r]$;

$n_< = |S_<|$;

$n_= = |S| - (|S_<| + |S_>|)$;

if $(k \le n_<)$ **then**

return $\textsc{RandSelect}(S_<, k)$;

else if $(k \le (n_< + n_=))$ **then**

return $S[r]$;

else

return $\textsc{RandSelect}(S_>, k - n_< - n_=)$;

end if

Expected running time

$T(n) = T(n-1) + O(n) = O(n^2)$ Worst case time

$O(n)$ Average time RAM model

$O(n/B)$ I/O's for the disk model

"good selection" a partition where $n_<$ and $n_>$ are not larger than $2/3n$. Positions of the pivot for a good selection: the blue

Probability to have a good selection is 1/3. Let T_a the average time:

$$T_a(n) \leq O(n) + 1/3\, T_a((2/3)n) + 2/3\, T_a(n)$$ subtract $T_a(n)$

$$1/3\, T_a(n) \leq O(n) + 1/3\, T_a((2/3)n)$$ multiply by 3

$$T_a(n) \leq O(n) + T_a((2/3)n)$$

Expected running time

$$T_a(n) \le O(n) + T_a(2/3n)$$

It can be computed with Master Th. (or by substitution)

$$T_a(n) \le O(n)$$

RandSelect is very efficient in average!

2-level model:

$$T_a(n) \le O(n/B) + T_a(2/3n) = O(n/B)$$

Since the procedure partition can be executed in the 2-level model with a single pass over the input items.

Use RandSelect to improve QuickSort

- Instead of 1 pivot, select at random 2s+1 pivots.
- Select the median pivot among the 2s+1

- s=1 select 3 pivot and with 2 comparisons select the median.
- s>1 : sort the items and select the median O(slogs)
- select the median (k =s/2) by RandSelect O(s) average.

- Select as pivot the median item of the whole array k=n/2
- Select a pivot that generates 2 a balanced partition, the 2 parts are fractions of n: αn and $(1-\alpha)n$ with $\alpha<0.5$. Apparently meaningless, is good for parallel CPU.