HW#1

1515655 컴퓨터과학과 임소희

1. MINMAX3(I,j,min,max) { // i=starting index, j=last index

// min,max = output for considering range

/\*I, j are integers s.t. 1<=i<=j<=n \*/

If (i==j) { min 🡨 A(i); /\*한 개의 element인 경우, min,max 모두 같은 수가 됨 \*/

Max 🡨 A(i); } // A(index) = list 내 index에 해당하는 실제 값

else if (i==j-1) { if A(i) < A(j) {min 🡨 A(i); max 🡨 A(j); }

else { min 🡨 A(j); max 🡨 A(i); } } /\*2개의 element 인 경우\*/

else { mid1 🡨 ; // : 버림

mid2 🡨 ;

call MINMAX3 (i,mid1,fmin,fmax); //first part

call MINMAX3 (mid1+1,mid2,smin,smax); //second part

call MINMAX3 (mid2+1,j,tmin,tmax); //third part

min 🡨 MIN(fmin,smin,tmin); //COMBINE 작업

max 🡨 MAX(fmin,smin,tmin); //COMBINE 작업 } }

Recurrence relation:

T(n) = 0 if n=1 // 비교회수 0이므로

1. if n=2 // 한번만 비교하면 됨
2. if n=3 // min,max 함수(횟수 : 2+2)만 부르면 됨

3\*└T)┘ + +4 if n>2 /\* 세 그룹으로 나눠 minmax 수행, min,max 비교 +4

세 수 중 min, max 선택 할 때, 두 번 비교하게 됨

ex)a,b,c 🡪 a,b 비교 후 큰 수 🡪 그 수와 c 비교 후 max\*/

Complexity analysis:

∵ Let n= T(n) = 3T)+4

= 3(3T)+4)+4 = T)+3\*4+4

= 3(3(3T)+4)+4)+4 = T)+\*4+3\*4+4

.

.

.

= T)+(++… +3+1)\*4 // n=

= T(3)+()\*4 = +2() // T(3)=4

= +-2 = 2n-2

∴ T(n) = 2n-2

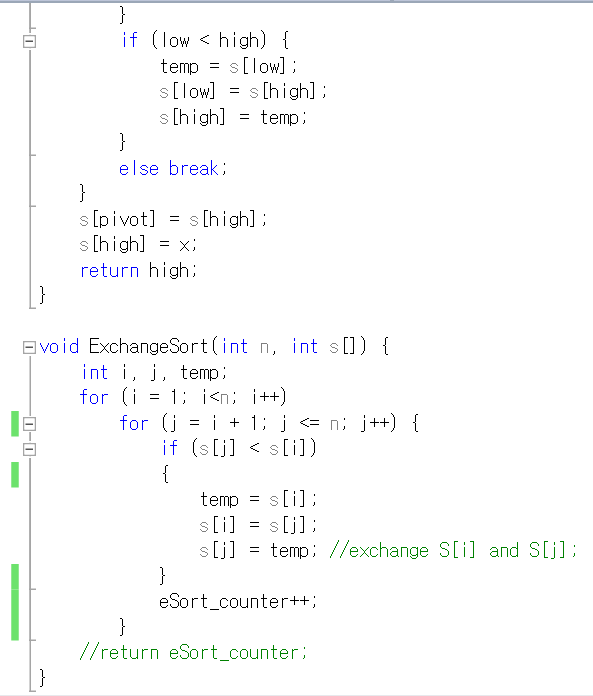
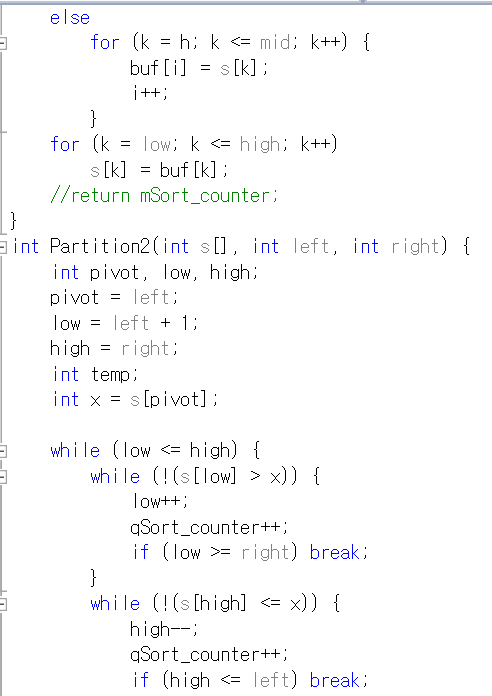
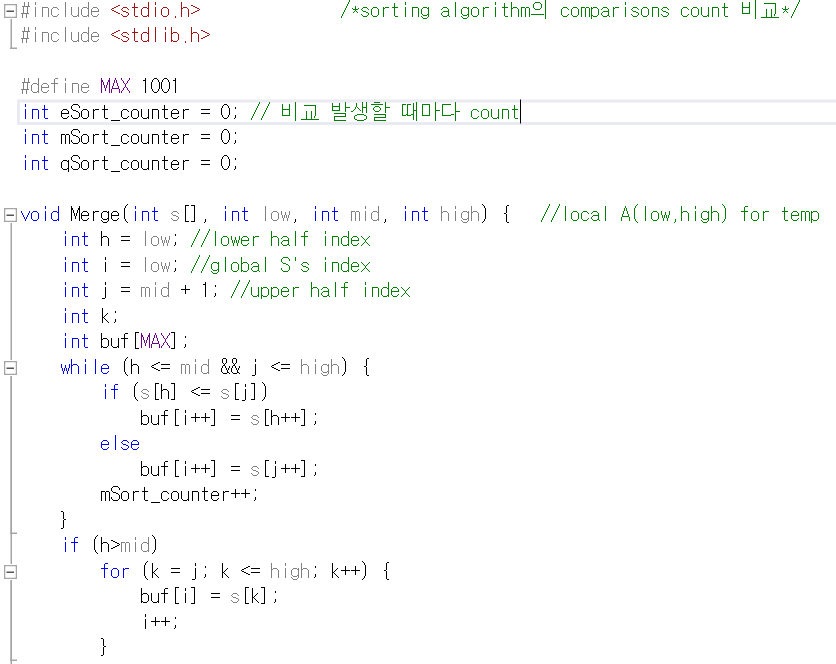
Closed form:

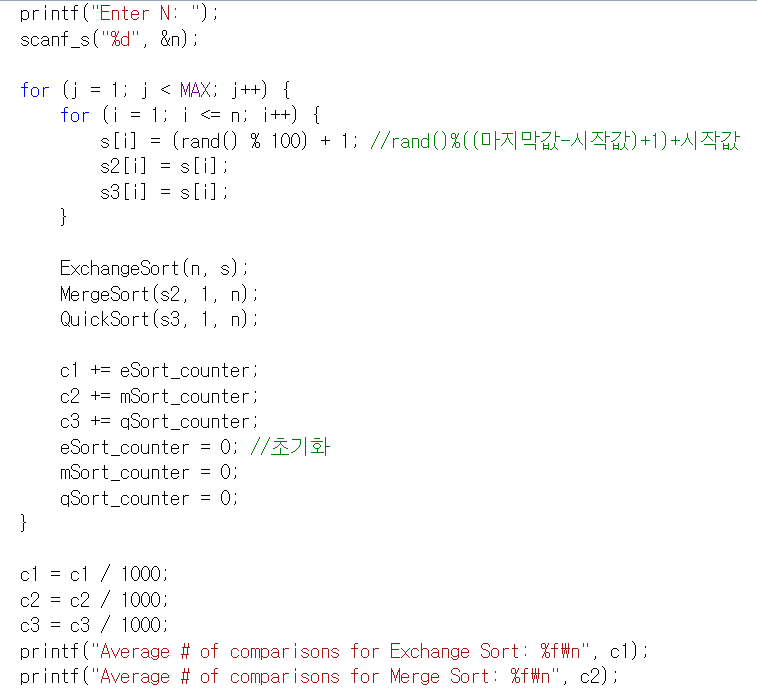
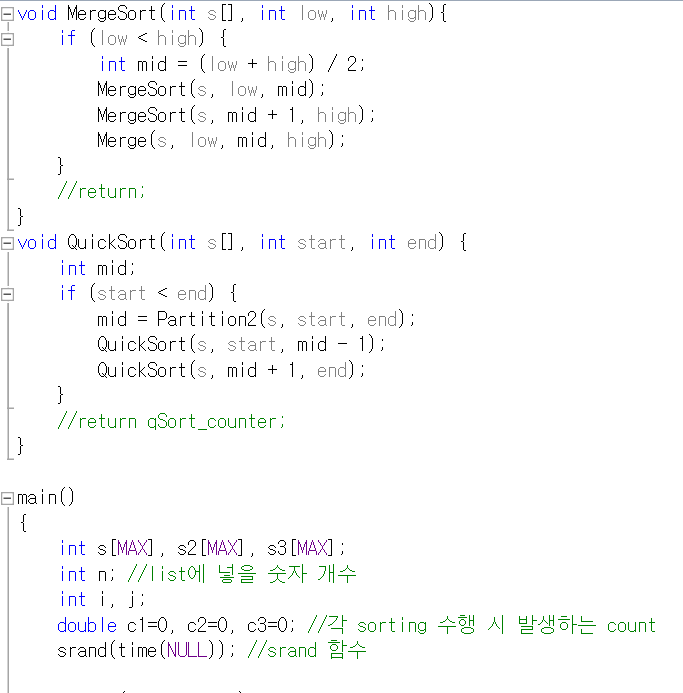
T(n) = n-1 best-case : Ω(n)

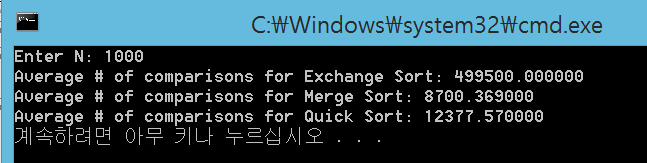
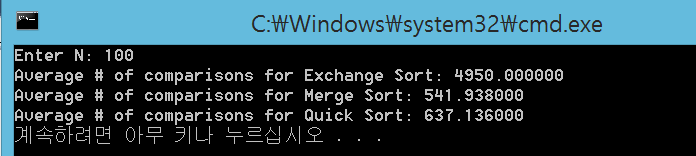
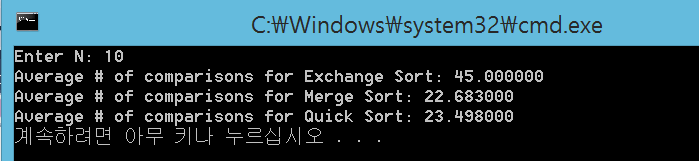
Average and worst-case : O(n)

∴ Θ(n)

1. sorting algorithms의 average numbers of comparisions 비교 (1000 test sets)







작은 수를 넣을 때는 comparison count가 비슷하나(merge<=quick)

수가 커질수록 차이가 커진다.

Merge🡪Quick🡪=Exchange sort 순으로 comparison count가 커진다.

(comparison 큰 순서 : merge<=quick<exchange)

아래의 표는 sort algorithms의 time complexity를 비교한 것이다.

|  |  |
| --- | --- |
| Sort algorithms | Time complexity |
| Exchange | Θ() 🡪 O() |
| Merge | O() |
| Quick | Worst case : O()  Average case : O() |

Time complexity는 대부분 worst case로 구할 수 있다(Quick의 경우 average case 고려).

Time complexiy의 속도를 비교하면

(exchange)🡪 (quick)🡪= (merge)순으로 빨라진다.

(속도 빠른 순서 : (exchange) < (quick) <= (merge))

따라서, 앞의 코드 실행 결과 comparison count와 비교했을 때와 속도 순위의 상관관계를 유추 할 수 있다.

즉, Comparison이 적을수록 속도도 빠른 것이다.