

Solutions to CS/MCS 401 Exercise Set #7 (Fall 2007)

Exercise 8.1-1 The minimum depth of a leaf node is $n-1$, as every comparison sorting algorithm requires at least $n-1$ comparisons, even in the best case. Suppose the sorted order for an array a of size n is $a[i_1], a[i_2], a[i_3], \dots, a[i_{n-1}], a[i_n]$.

The sorting algorithm must compare $a[i_1]$ with $a[i_2]$; otherwise it has no way to distinguish the order

$$a[i_1], a[i_2], a[i_3], \dots, a[i_{n-1}], a[i_n]$$

from

$$a[i_2], a[i_1], a[i_3], \dots, a[i_{n-1}], a[i_n],$$

since every comparison other than that of $a[i_1]$ with $a[i_2]$ turns out the same in both cases.

Likewise, it must compare $a[i_2]$ with $a[i_3]$, ..., $a[i_{n-1}]$ with $a[i_n]$.

Exercise 8.1-3 Suppose a comparison sorting algorithm runs in linear time from some fraction $\delta(n)$ of its inputs. This means that there exists a constant C (not depending on n) such that, for all n sufficiently large, the algorithm performs at most Cn comparisons for $\delta(n)n!$ of its $n!$ inputs. In the decision tree, there must be at least $\delta(n)n!$ leaves at depth Cn or less. But we know that the number of leaves at depth Cn or less is bounded by 2^{Cn} . So $\delta(n)n! \leq 2^{Cn}$, or $\delta(n) \leq 2^{Cn}/n!$. Approximating $n!$ by Stirling's formula gives

$$\delta(n) \leq 2^{Cn}/n! \leq 2^{Cn}/((n/e)^n \text{sqrt}(2\pi n)) = (2^C e/n)^n / \text{sqrt}(2\pi n).$$

Exercise 8.1-3 asks specifically about the case $\delta(n) = 1/2$, $\delta(n) = 1/n$, and $\delta(n) = 1/2^n$.

In none of these cases is $\delta(n) \leq (2^C e/n)^n / \text{sqrt}(2\pi n)$ for some constant C and all n sufficiently large. If $\delta(n) = 1/2^n$, then $\delta(n)/((2^C e/n)^n / \text{sqrt}(2\pi n)) = (n/2^{1+C} e)^n \text{sqrt}(2\pi n)$ approaches ∞ as n approaches ∞ , since $n/2^{1+C} e > 1$ for all n sufficiently large. So a comparison sorting algorithm cannot run in linear time even for $1/2^n$ of its inputs.

Exercise K

i	j,p											r		
		36	83	75	48	14	71	64	22	91	69	58	88	72

	i,p	j											r		
			36	83	75	48	14	71	64	22	91	69	58	88	72

	i,p		j										r			
				36	83	75	48	14	71	64	22	91	69	58	88	72

	i,p			j									r	r			
					36	83	75	48	14	71	64	22	91	69	58	88	72

		p	i				j							r			
					36	48	75	83	14	71	64	22	91	69	58	88	72

		p		i			j							r			
					36	48	14	83	75	71	64	22	91	69	58	88	72

		p			i			j						r				
						36	48	14	71	75	83	64	22	91	69	58	88	72

		p				i			j					r					
							36	48	14	71	64	83	75	22	91	69	58	88	72

		p					i			j				r						
								36	48	14	71	64	22	75	83	91	69	58	88	72

		p						i			j			r							
									36	48	14	71	64	22	83	75	91	69	58	88	72

		p							i			j		r								
										36	48	14	71	64	22	69	75	91	83	58	88	72

		p								i			j	r									
											36	48	14	71	64	22	69	58	91	83	75	88	72

		p									i			j,r										
												36	48	14	71	64	22	69	58	91	83	75	88	72

		p										i			j,r										
													36	48	14	71	64	22	69	58	72	83	75	88	91

Exercise L Elements that are shaded will be exchanged in the next step.

left												right
36	83	75	48	14	71	64	22	91	69	58	88	72

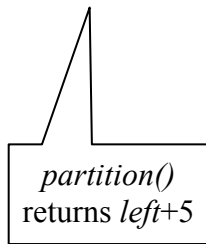
left	lo											right
64	83	75	48	14	71	36	22	91	69	58	88	72
										hi ← hi ← hi		

left		lo										right
64	58	75	48	14	71	36	22	91	69	83	88	72
										hi ← hi ← hi		

left					lo → lo → lo		hi					right
64	58	22	48	14	71	36	75	91	69	83	88	72

left						hi	lo					right
64	58	22	48	14	36	71	75	91	69	83	88	72

left												right
36	58	22	48	14	64	71	75	91	69	83	88	72



partition()
 returns *left+5*