

Algorithm
Autumn 2010, Midterm Exam, November 11
9:20-12:05am

1. (20 %) Please describe briefly the following sorting algorithms along with their time complexities. Which of them are stable sorting algorithms? Which of them are in-place sorting algorithms?

- a. Counting sort
- b. Radix sort
- c. Bucket sort
- d. Merge sort
- e. Insertion sort

2. (5 %) Find the error in the following proof that all horses are the same color.

CLAIM: In any set of h horses, all horses are the same color.

PROOF: By induction on h .

Basis: For $h = 1$. In any set containing just one horse, all horses clearly are the same color.

Induction step: For $k > 1$ assume that the claim is true for $h = k$ and prove that it is true for $h = k + 1$. Take any set H of $k + 1$ horses. We show that all the horses in this set are the same color. Remove one horse from this set to obtain the set H_1 with just k horses. By the induction hypothesis, all the horses in H_1 are the same color. Now replace the removed horse and remove a different one to obtain the set H_2 . By the same argument, all the horses in H_2 are the same color. Therefore all the horses in H must be the same color, and the proof is complete.

3. (20 %)

- a. Rank $2^{\lg n}$, n^2 , $n!$, $n^{\lg \lg n}$, $n \lg n$ by order of growth.
- b. Please show that if $f(n) = a_m n^m + \dots + a_1 n + a_0$, then $f(n) = \Theta(n^m)$.
- c. Describe the three types of arguments in mathematical proofs.
- d. Show that $n^3 + 3n = \Theta(n^3)$.

4. (15 %) Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(n/4) + T(3n/4) + 5n$. Use the substitution method to verify your answer.

5. (10 %) Use the master method to give tight asymptotic bounds for the following recurrences.

- a. $T(n) = 9T(n/3) + n^2$.
- b. $T(n) = 9T(n/3) + n^3$.
- c. $T(n) = 9T(n/3) + n^4$.

6. (15 %) Analysis the time complexity and prove the correctness of BUILD-MAX-HEAP procedure. (Hint: At the start of each iteration of the for loop of lines 2-3, each node $i+1, i+2, \dots, n$ is the root of a max-heap. For an n -element heap, height is $\lfloor \lg n \rfloor$ and at most $\lceil n / 2^{h+1} \rceil$ nodes of any height h .)

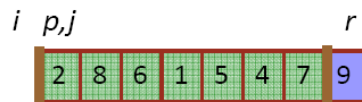
MAX-HEAPIFY(A, i)

1. $\ell \leftarrow \text{LEFT}(i)$
2. $r \leftarrow \text{RIGHT}(i)$
3. **if** $\ell \leq \text{heap-size}[A]$ and $A[\ell] > A[i]$
4. **then** $\text{largest} \leftarrow \ell$
5. **else** $\text{largest} \leftarrow i$
6. **if** $r \leq \text{heap-size}[A]$ and $A[r] > A[\text{largest}]$
7. **then** $\text{largest} \leftarrow r$
8. **if** $\text{largest} \neq i$
9. **then** exchange $A[i] \leftrightarrow A[\text{largest}]$
10. MAX-HEAPIFY ($A, \text{largest}$)

BUILD-MAX-HEAP(A)

1. $\text{heap-size}[A] \leftarrow \text{length}[A]$
2. **for** $i \leftarrow \lfloor \text{length}[A]/2 \rfloor$ **downto** 1
3. **do** MAX-HEAPIFY(A, i)

7. (15 %) Analysis the time complexity of the RANDOMIZED-QUICKSORT algorithm. Illustrate the operation of PARTITION on the following array.



1. RANDOMIZED-QUICKSORT(A, p, r)
2. **if** $p < r$
3. **then** $q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r)$
4. RANDOMIZED-QUICKSORT($A, p, q-1$)
5. RANDOMIZED-QUICKSORT($A, q+1, r$)

RANDOMIZED-PARTITION(A, p, r)

1. $i \leftarrow \text{RANDOM}(p, r)$
2. exchange $A[r] \leftrightarrow A[i]$
3. **return** PARTITION(A, p, r)

PARTITION(A, p, r)

1. $x \leftarrow A[r]$
2. $i \leftarrow p - 1$
3. **for** $j \leftarrow p$ **to** $r - 1$
4. **do if** $A[j] \leq x$
5. **then** $i \leftarrow i + 1$
6. exchange $A[i] \leftrightarrow A[j]$
7. exchange $A[i+1] \leftrightarrow A[r]$
8. **return** $i + 1$

意圖不軌者
將受最嚴厲的懲罰