Analysis of Algorithms

Linear Selection and Linear Sorting

If more than one question appears correct, choose the more specific answer, unless otherwise instructed.

Concept: *linear selection*

1. Suppose you wish to find both the minimum and maximum values in an array of n values, n being odd. Consider this algorithm:

   (i) Make a pass through the array and find the minimum

   (ii) Make a pass through the array and find the maximum

   (iii) Report the minimum and the maximum at termination

   What is the minimum number of key comparisons that this algorithm needs to perform?

   (A) $2n-1$
   (B) $2n-2$
   (C) $2n+2$
   (D) $2n+1$
   (E) $2n$

2. Suppose you wish to find both the minimum and maximum values in an array A of n values, n being odd. Consider this algorithm:

   (i) Set the maximum and minimum to the first value in the array

   (ii) Make a pass through the array `(i = 1; i < n; i += 2)`

   (iii) Compare A[i] with A[i+1]

   (iv) Update the minimum if the smaller of the two is less

   (v) Update the maximum if the larger of the two is greater

   (vi) Report the minimum and the maximum at loop termination

   What is the minimum number of key comparisons that this algorithm needs to perform?

   (A) $2n-3$
   (B) $3n-1$
   (C) $3\frac{n-1}{2}+2$
   (D) $3n-3$
   (E) $3n$
   (F) $3\frac{n-1}{2}$

3. Consider running the linear selection algorithm on an array of n unique elements. What is a tight lower bound on the number of elements less than the median of medians? Assume the median of medians is found with groups of five and that there are an odd number of groups.

   (A) $3\frac{n}{5}+2$
   (B) $5\frac{n}{10}+2$
   (C) $\frac{3n+5}{10}$
   (D) $3\frac{n}{5}+3$
   (E) $2\frac{n}{5}+3$
   (F) $3\frac{n}{10}+3$
   (G) $5\frac{n}{10}+3$
   (H) $2\frac{n}{5}+2$

4. Consider running the linear selection algorithm on an array with n unique elements. What is a tight lower bound on the number of elements less than the median of medians? Assume the median of medians is found with groups of three and that there are an odd number of groups.

   (A) $3\frac{n}{6}+1$
   (B) $2\frac{n}{3}+1$
   (C) $2\frac{n}{6}+1$
   (D) $\frac{n}{3}+1$
   (E) $\frac{n}{3}$
   (F) $2\frac{n}{6}+2$
   (G) $2\frac{n}{3}+2$
   (H) $3\frac{n}{6}+2$

5. Consider running the linear selection algorithm on an array with n unique elements. What is a tight lower bound on the number of elements less than the median of medians? Assume the median of medians is found with groups of seven and that there are an odd number of groups.

   (A) $3\frac{n}{7} + 2$
   (B) $11\frac{n}{14} + 3$
   (C) $3\frac{n}{14} + 2$
   (D) $11\frac{n}{14} + 2$
   (E) $\frac{2n+7}{7}$
   (F) $\frac{2n+28}{14}$
   (G) $3\frac{n}{14} + 3$
   (H) $3\frac{n}{7} + 3$

6. Consider running the linear selection algorithm on an array with $n = 7^k$ unique elements. What is a reasonable recurrence equation to describe the running time of the algorithm? Assume the median of medians is found with groups of seven.

   (A) $T(n) = T(\frac{n}{7}) + T(\frac{10n}{14}) + \theta(n)$
   (B) $T(n) = T(\frac{n}{7}) + T(\frac{4n}{14}) + \theta(n)$
   (C) $T(n) = T(\frac{n}{7}) + T(\frac{4n}{7}) + \theta(n)$
   (D) $T(n) = T(\frac{n}{7}) + T(\frac{6n}{7}) + \theta(n)$
   (E) $T(n) = T(\frac{n}{7}) + T(\frac{11n}{14}) + \theta(n)$
   (F) $T(n) = T(\frac{n}{7}) + T(\frac{5n}{7}) + \theta(n)$

7. Consider running the linear selection algorithm on an array with $n = 3^k$ unique elements. What is a reasonable recurrence equation to describe the running time of the algorithm? Assume the median of medians is found with groups of three.

   (A) $T(n) = T(\frac{n}{3}) + T(\frac{n}{6}) + \theta(n)$
   (B) $T(n) = T(\frac{n}{3}) + T(\frac{5n}{6}) + \theta(n)$
   (C) $T(n) = T(\frac{n}{3}) + T(\frac{n}{2}) + \theta(n)$
   (D) $T(n) = T(\frac{n}{3}) + T(\frac{2n}{5}) + \theta(n)$
   (E) $T(n) = T(\frac{n}{3}) + T(\frac{n}{3}) + \theta(n)$
   (F) $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + \theta(n)$

8. Consider running the linear selection algorithm on an array with $n = 5^k$ unique elements. What is a reasonable recurrence equation to describe the running time of the algorithm? Assume the median of medians is found with groups of five.

   (A) $T(n) = T(\frac{n}{5}) + T(\frac{2n}{5}) + \theta(n)$
   (B) $T(n) = T(\frac{n}{5}) + T(\frac{n}{2}) + \theta(n)$
   (C) $T(n) = T(\frac{n}{5}) + T(\frac{9n}{10}) + \theta(n)$
   (D) $T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + \theta(n)$
   (E) $T(n) = T(\frac{n}{5}) + T(\frac{3n}{5}) + \theta(n)$
   (F) $T(n) = T(\frac{n}{5}) + T(\frac{4n}{5}) + \theta(n)$

9. **T** or **F**: If the linear selection algorithm uses groups of three to find the median of medians, the asymptotic run time is still $\theta(n)$.

10. **T** or **F**: If the linear selection algorithm uses groups of seven to find the median of medians, the asymptotic run time is still $\theta(n)$.

Concept: *decision trees*

11. In proving a tight lower bound for a class of algorithms, one tries to establish, over all possible algorithms:

    (A) the best possible worst case
    (B) the worst possible best case
    (C) the best possible best case
    (D) the worst possible worst case

12. In an efficient decision tree for comparison sorts of n numbers, what is the smallest possible depth of a leaf, in the best case? Assume the root is at depth 0.

    (A) $n - 1$
    (B) approximately $\log n$
    (C) approximately $n \log n$
    (D) $n!$
    (E) n
    (F) 1

13. In an efficient decision tree for comparison sorts of n numbers, what is the largest possible depth of a leaf, in the worst case? Assume the root is at depth 0.

    (A) approximately $\log n$
    (B) n
    (C) $n - 1$
    (D) 1
    (E) approximately $n \log n$
    (F) $n!$

14. In an efficient decision tree for comparison sorts of n numbers, what does the shortest path from the root to a leaf represent?

    (A) the best case behavior of the sort
    (B) the worst case behavior of the sort
    (C) the average case behavior of the sort
    (D) nothing, the longest path is what's important

15. An efficient decision tree for comparison sorts of n numbers has how many leaves?

    (A) $n \log n$
    (B) $n!$
    (C) $\log n$
    (D) 2^n

16. Deriving a tight lower time bound for comparison sorts, based upon an efficient decision tree, yields:

    (A) $\Omega(n)$
    (B) $\Omega(n \log n)$
    (C) unbounded
    (D) $\omega(n \log n)$
    (E) $O(n \log n)$
    (F) $O(n)$

Concept: *linear sorting fundamentals*

17. Stability in a sort means:

    (A) the sort can work on any kind of number (integers, reals, etc.)
    (B) the asymptotic run time does not vary upon the input
    (C) the order of ties in the output reflects the input

18. The best case behavior for insertion sort is:

    (A) linear
    (B) quadratic
    (C) exponential
    (D) log linear

19. **T** or **F**: A linear-time sort does not compare entire keys with one another.

20. **T** or **F**: A linear-time sort must always compare entire keys with one another.

Concept: *linear sorting – counting*

For counting sort, assume array A holds the data to be sorted, array B holds the sorted data, and array C holds the counts. The index i is used to sweep arrays A and B and the index j is used to sweep array C.

21. Counting sort is:

    (A) stable if lower indexed elements from the input array are transferred to higher indexed elements in the output array.
    (B) stable if higher indexed elements from the input array are transferred to higher indexed elements in the output array.
    (C) always stable
    (D) always unstable

22. Suppose you are to sort n numbers using counting sort. What size should the C array be?

    (A) equal to n
    (B) greater than n
    (C) less than n
    (D) there's not enough information

23. Consider using a counting sort to sort the input array `[2,5,0,3,2,0,3,3]`. After the first phase, when `C[i]` holds the number of elements equal to i, the array C looks like:

    (A) `[2,0,5,2,3,0,3,3]`
    (B) `[2,2,4,6,7,8]`
    (C) `[2,2,4,7,7,8]`
    (D) `[0,0,2,2,3,3,3,5]`
    (E) `[2,0,2,3,0,1]`

24. Consider using a counting sort to sort the input array `[2,5,0,3,2,0,3,3]` with auxiliary array C. After the second phase, when `C[i]` holds the number of elements less than or equal to i, the array C looks like:

    (A) `[2,0,5,2,3,0,3,3]`
    (B) `[2,2,4,6,7,8]`
    (C) `[2,2,4,7,7,8]`
    (D) `[2,0,2,3,0,1]`
    (E) `[0,0,2,2,3,3,3,5]`

25. Consider using a stable counting sort to sort the input array `[2,5,0,3,2,0,3,3]` with destination array B. At start of phase three, using a right to left placement, the first element to be placed in B is:

    (A) 1, at index 0
    (B) 2, at index 3
    (C) 5, at index 7
    (D) 4, at index 5
    (E) 0, at index 1
    (F) 3, at index 6

26. Let n be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and k is the maximum number in the collection. The time complexity of counting sort is:

    (A) $\Theta(n+k)$
    (B) $\Theta(nk)$
    (C) $\Theta(n^k)$
    (D) $\Theta(n \log k)$

27. Let n be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and k is the maximum number in the collection. If $k = o(n)$, then the time complexity of counting sort is:

    (A) $\Theta(k)$
    (B) $\Theta(n \log k)$
    (C) $\Theta(n^2)$
    (D) $\Theta(n)$
    (E) $\Theta(k^2)$
    (F) $\Theta(k \log n)$

28. **T** or **F**: Suppose the lower bound of some numbers to be sorted is not zero. Counting sort can still be used without changing its fundamental time bounds.

29. **T** or **F**: Suppose the lower bound of some numbers to be sorted, `LB`, is not zero. Counting sort can still be used without changing its fundamental time bounds, if, among other changes, the statement `C[A[i]] += 1` is changed to `C[A[i]-LB] += 1`.

30. **T** or **F**: Suppose the lower bound of some numbers to be sorted, `LB`, is not zero. Counting sort can still be used without changing its fundamental time bounds, if, among other changes, the statement `B[C[A[i]]-1] = A[i]` is changed to `B[C[A[i-LB]]-1] = A[i]`.

31. **T** or **F**: Counting sort can be used to sort n decimal numbers uniformly distributed over the range of zero to n^5 in linear time.

32. Consider using counting sort to sort n numbers uniformly distributed over the range of zero to n^k. The asymptotic complexity of the sort will be

    (A) $\theta(k)$
    (B) $\theta(k \log n)$
    (C) $\theta(n^k + n)$
    (D) $\theta(n+k)$
    (E) $\theta(n \log k)$
    (F) $\theta(n)$

Concept: *linear sorting – radix*

33. Suppose the recurrence equation $T(p, q) = T(p, q-1) + \theta(p)$ is used to describe radix sort. Considering only the sorting of the numbers on a particular digit, which sort is consistent with that equation?

    (A) radix sort
    (B) counting sort
    (C) mergesort sort
    (D) selection sort

34. Suppose the recurrence equation $T(p, q) = T(p, q-1) + \theta(p \log p)$ is used to describe radix sort. Considering only the sorting of the numbers on a particular digit, which sort is consistent with that equation?

    (A) selection sort
    (B) radix sort
    (C) counting sort
    (D) mergesort sort

35. Suppose the recurrence equation $T(p, q) = T(p, q-1) + \theta(p^2)$ is used to describe radix sort. Considering only the sorting of the numbers on a particular digit, which sort is consistent with that equation?

    (A) counting sort
    (B) selection sort
    (C) mergesort sort
    (D) radix sort

36. Suppose the recurrence equation $T(p, q) = T(p, q-1) + \theta(p)$ is used to describe radix sort. Which variable in the equation refers to the number of digits?

    (A) q
    (B) d
    (C) p
    (D) n

37. Consider using radix sort for sorting the following numbers:

    ```
    558
    354
    318
    622
    ```

    After the first pass, the order of the numbers is:

    (A) 622, 558, 354, 318
    (B) 318, 354, 558, 622
    (C) 622, 354, 558, 318
    (D) 622, 558, 318, 354
    (E) 622, 354, 318, 558
    (F) 354, 318, 558, 622

38. Let n be the count of numbers in a collection of positive, base$_{10}$ numbers. Let m be the number of digits in the largest number in the collection. Suppose the auxiliary sort works in $\Theta(n)$ time. Then radix sorting takes time:

    (A) $\Theta(m \log n)$
    (B) $\Theta(nm)$
    (C) $\Theta(n \log n)$
    (D) $\Theta(n \log m)$

39. Let n be the count of numbers in a collection of positive, base$_{10}$ numbers. Let m be the number of digits in the largest number in the collection. Suppose the auxiliary sort works in $\Theta(n \log n)$ time. Then radix sorting takes time:

    (A) $\Theta(mn \log n)$
    (B) $\Theta(n \log m)$
    (C) $\Theta(n \log(m + n))$
    (D) $\Theta(n \log(mn))$

40. **T** or **F**: Suppose during one pass of radix sort, there is a tie between two numbers. Since they are considered the same number, it does not matter if those two numbers swap positions.

41. Consider the sort used for each pass of radix sort. Must the auxiliary sort be stable?

    (A) Yes, because swapping ties can undo the work of previous passes
    (B) No, because radix sort works even if the auxilliary sort is unstable.
    (C) Yes, because swapping ties can undo the work of future passes
    (D) No, because there can be no ties in radix sort.

42. **T** or **F**: Radix sort can be used to sort n decimal numbers uniformly distributed over the range of zero to n^5 in linear time.

Concept: *linear sorting – bucket*

43. **T** or **F**: If bucket sort uses equally-sized buckets, then the distribution of the incoming numbers need not be uniform for the sort to run in expected linear time.

44. Consider using bucket sort to sort n numbers evenly distributed over the range 0 to m. Roughly, how many buckets should you use?

    (A) n
    (B) m
    (C) $\frac{m}{n}$
    (D) $\frac{n}{m}$

45. Consider using bucket sort to sort n numbers evenly distributed over the range 0 to m. Roughly, what size bucket should you use?

    (A) m
    (B) $\frac{n}{m}$
    (C) $\frac{m}{n}$
    (D) n

46. Consider using bucket sort to sort n non-negative numbers evenly distributed over the range 0 to m. The range of the first bucket should be:

    (A) 0 to $\frac{m}{n}$
    (B) 0 to $\frac{n}{m}$
    (C) $\frac{n}{m}$ to $\frac{2n}{m}$
    (D) $\frac{m}{n}$ to $2\frac{m}{n}$

47. Consider using bucket sort to sort n non-negative numbers evenly distributed over the range 0 to m. The average count of numbers in a bucket is:

    (A) $\frac{m}{n}$
    (B) 1
    (C) $\frac{n}{m}$
    (D) 0

48. Consider a bucket sort that uses insertion sort to sort the individual buckets. Suppose you could bound the maximum count of numbers in a bucket to a constant C. Then the asymptotic running time to sort one bucket would be:

    (A) $\Theta(n)$, because insertion sort is linear in the best case
    (B) $\Theta(1)$
    (C) $\Theta(n \log n)$, because insertion sort is log-linear in the average case
    (D) $\Theta(n^2)$, because insertion sort is quadratic in the worst case

49. Consider using bucket sort to sort n non-negative numbers evenly distributed over the range 0 to m. The *expected* running time is:

    (A) log-linear
    (B) constant
    (C) linear
    (D) quadratic

50. Consider using bucket sort to sort n non-negative numbers in the range 0 to m with no a priori knowledge of the distribution of the numbers. Using insertion sort as the auxilliary sort, the worst case running time is:

    (A) constant
    (B) cubic
    (C) quadratic
    (D) linear

51. **T** or **F**: Bucket sort can be used to sort n decimal numbers uniformly distributed over the range of zero to n^5 in linear time.